SHORT-TERM FORECAST TECHNIQUES FOR ENERGY MANAGEMENT SYSTEMS IN MICROGRID APPLICATIONS

Benson Herman Mbuya

A Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy in Sustainable Energy Science and Engineering of the Nelson Mandela African Institution of Science and Technology

Arusha, Tanzania

July, 2023

ABSTRACT

In the 2015 Paris Agreement, 195 countries adopted a global climate agreement to limit the global average temperature rise to less than 2°C. Achieving the set targets involves increasing energy efficiency and embracing cleaner energy solutions. Although advances in computing and Internet of Things (IoT) technologies have been made, there is limited scientific research work in this arena that tackles the challenges of implementing low-cost IoT-based Energy Management System (EMS) with energy forecast and user engagement for adoption by a layman both in off-grid or microgrid tied to a weak grid.

This study proposes an EMS approach for short-term forecast and monitoring for hybrid microgrids in emerging countries. This is done by addressing typical submodules of EMS namely: load forecast, blackout forecast, and energy monitoring module. A short-term load forecast model framework consisting of a hybrid feature selection and prediction model was developed. Prediction error performance evaluation of the developed model was done by varying input predictors and using the principal subset features to perform supervised training of 20 different conventional prediction models and their hybrid variants. The proposed principal k-features subset union approach registered low error performance values than standard feature selection methods when it was used with the 'linear Support Vector Machine (SVM)' prediction model for load forecast. The hybrid regression model formed from a fusion of the best 2 models ('linearSVM' and 'cubicSVM') showed improved prediction performance than the individual regression models with a reduction in Mean Absolute Error (MAE) by 5.4%.

In the case of the EMS blackout prediction aspect, a hybrid Adaptive Similar Day (ASD) and Random Forest (RF) model for short-term power outage prediction was proposed that predicted accurately almost half of the blackouts (49.16%), thereby performing slightly better than the stand-alone RF (32.23%), and ASD (46.57%) models. Additionally, a low-cost EMS smart meter was developed to realize the implemented energy forecast and offer user engagement through monitoring and control of the microgrid towards the goal of increasing energy efficiency.

DECLARATION

I, Benson Herman Mbuya, do hereby declare to the Senate of The Nelson Mandela African Institution of Science and Technology that this dissertation titled "*Short-Term Forecast Techniques for Energy Management Systems in Microgrid Applications*" is my original work and that it has neither been submitted nor being concurrently submitted for degree award in any other institution.

21/07/2023

Benson Mbuya — Date

The above declaration is confirmed by:

21/07/2023

Dr. Thomas Kivevele — Date

Firmato digitalmente
da:MARCO MERLO
Organizzazione:
POLITECNICO DI
MILANO/80057930150

21/07/2023

Prof. Marco Merlo — Date

COPYRIGHT

CERTIFICATION

The undersigned certify that they have read and hereby recommend for acceptance by the Nelson Mandela African Institution of Science and Technology a dissertation entitled "*Short-Term Forecast Techniques for Energy Management Systems in Microgrid Applications*" in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Sustainable Energy Science and Engineering of the Nelson Mandela African Institution of Science and Technology, Arusha Tanzania.

Dr. Thomas Kivevele — Date: 21/07/2023

Firmato digitalmente da:MARCO MERLO
Organizzazione: POLITECNICO DI MILANO/80057930150

Prof. Marco Merlo — Date: 21/07/2023

ACKNOWLEDGMENTS

I am foremost grateful to the almighty God, Jehovah, for the gift of life, and the prospect He has given mankind of studying his awe-inspiring works forever (Ecclesiastes 3:11). I especially value the significant and constructive contribution of my supervisor Dr. Thomas Kivevele, his guidance and encouragement helped me a great deal. I am also indebted to Prof. Marco Merlo for his self-sacrificing spirit in sharing his expertise and accommodating me in his tight schedule which greatly enhanced the quality of my research.

I am grateful to Politecnico di Milano for sponsoring my PhD. studies under the project "Program capacity" building in Tanzania contract No. PA/015/2013/2013, and for being involved in the energy4growing project which saw a deployment of a smart microgrid at Ngarenanyuki secondary school. I thank Prof. Emanuela Colombo for all the assistance she provided in my PhD journey and for spearheading collaborations between PoliMi and Arusha Technical College. I also appreciate all the help I received from Dr. Claudio Brivio, Dr. Matteo Moncecchi, and Dr. Aleksandar Dimovski. I acknowledge the support I received from Ngarenanyuki secondary school's staff especially the school's former headmaster Mr. James Somi in data collection.

I acknowledge the Dean School of Materials, Energy, Water and Environmental Sciences (MEWES) Prof. Kelvin Mtei for being approachable and helpful. I also acknowledge the kind assistance from academic officer Mr. Haji Chomba, and all insightful inputs from MEWES staff during graduate seminars. I also acknowledge The Arusha Technical College Administration and all the colleagues in the Electrical department for unwavering support in adapting to various challenges that came along at various stages of my research. My gratitude also goes to Kilimanjaro International Institute of Telecommunication, Electronics and Computers (KIITEC) and Powergen for their assistance in data collection and/or otherwise. Special thanks goes to Rafiki Electronics Techcentrall Arusha for providing the necessary hardware components for the research and also aiding in data collection.

Last but not least I am grateful to my family and relatives for the immeasurable moral support that proved instrumental in the fulfilment of this research. The list is by no means exhaustive, I remain indebted to all whose names I haven't mentioned and who assisted me in one way or another towards completing this research.

DEDICATION

This work is dedicated to my lovely wife Jane Mollel Mbuya who has been my anchor throughout my PhD research marathon.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS AND SYMBOLS

AC	Alternating Current
ADC	Analog to Digital Converter
Ah	Ampere hour
AI	Artificial Intelligence
AMI	Advanced metering infrastructure (AMI).
ANN	Artificial Neural Network
API	Application Peripheral Interface
APP	Application Program – a software program
AR	Autoregressive models
ARIMA	autoregressive-integrated moving average model
ARIMAX	Autoregressive-integrated moving average model with exogenous inputs
ARMA	Autoregressive moving average
ASD	Adaptive Similar Day algorithm
BEMS	Building Energy Management System
BF	Blackout Forecast
BMS	Battery Management System
CNN	Convolutional neural networks
CPP	Critical Peak Pricing
CSS	Cascading Style Sheets
DC	Direct Current
DER	Distributed Energy Resources
DG	Diesel Generator
DL	Deep learning
DR	Demand Response
DSM	Demand Side Management
EMS	Energy Management System
ES	Exponential smoothing
FS	Feature Selection
GA	Genetic algorithm
GDP	Gross Domestic Product
GM	Grey model
GPIO	General-Purpose Input/Output pin

GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HAN	Home Area Network
HMI	Human machine interface
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
HVAC	Heating, ventilation, and air conditioning
IoT	Internet of Things
JSON	Javascript Object Notation
k-NN	k-nearest neighbour algorithm
kW	Kilo watt
LF	Load Forecast
LiFeSO4	Lithium iron phosphate
LSTM	Long short-term memory neural networks
LTLF	Long-term load forecast (LTLF)
MAE	Mean absolute error
MAPE	Mean absolute percentage error (MAPE)
MCU	Microcontroller Unit
ML	Machine Learning
MLP	Multiple layer perceptron
MOSFET	Metal–oxide–semiconductor field-effect transistor
MQTT	Message Queuing Telemetry Transport
MSE	Mean square error (MSE)
MTLF	Medium-term load forecast
NN	Neural Network
PIR	Passive Infrared sensor
PLC	Programmable Logic Controller
PV	Photovoltaic
PWM	Pulse-width-modulation
RES	Renewable Energy Sources
RF	Random forest algorithm
RMSE	Root mean square error (RMSE)
RNN	Recursive neural networks
ROC-AUC	Receiver operating characteristic area under the curve
RTP	Real Time Pricing

SCADA	Supervisory Control and Data Acquisition
SHEMS	Smart Home Energy Management System
SOC	state-of-charge
SQL	Structured Query Language
SSA	sub-Saharan Africa
STLF	Short-term load forecast
SVM	Support Vector Machines
SVR	Support vector regression
TANESCO	Tanzania Electric Supply Company Limited
ToU	Time of use tariffs
VSTLF	Very short-term load forecast

CHAPTER ONE

INTRODUCTION

1.1 Background of the Problem

The fast-rising world population has become a cause for concern in energy availability and resources as electricity utility companies try to keep up the pace and catch up with consumers' energy demand. More than ever before, the demand for more responsible and prudent energy use is crucial; especially in light of worsening climate change effects partly linked to heavy energy generation through fossil fuels. To this effect, emerging countries stand to benefit from the deployment of Energy Management Systems (EMS) and pick up the pace with initiatives similar to high-income countries towards Smart Grids. The EMS can go a long way in increasing the energy efficiency of microgrids as well as help utilities and consumers save money by optimizing operations of the invested energy sources to their full potential and lifespan. Continuous, cheap, and reliable electric energy is considered a vital impetus for economic development (Soliman & Al-Kandari, 2010).

Although electricity is a necessity in today's world, many countries in the developing world have limited access to it. This is especially so in Africa. Taking Tanzania as an example: In the 2012 census, only 17% of households had access to electricity 70% of which were urban residences and only 5.3% of rural households had access to electricity (National Bureau of Statistics, 2013). The geography, and cost required for the national grid to reach these villages deter any fast attempts to remedy the situation. A more promising solution has been to promote rural electrification through renewable energy (IEA, 2014). Despite their numerous advantages, renewable energy grids are often affected by intermittency. In most cases, these can be resolved by installing storage systems (Dinh *et al.,* 2020).

Energy Management Systems (EMS) is expected to be one way of increasing energy availability while minimizing generation costs, and environmental hazards (Balatsky *et al.,* 2015). The EMS is a system that leverages soft computing to monitor and enhance energy generation and utilization (Serna-Suarez *et al.,* 2015). The EMS toolkit usually includes the following important organs: A forecasting module that may involve a Load Forecast (LF) and Blackout Forecast (BF), a reporting mechanism or energy dashboard, and an energy storage management module (Zafar *et al.,* 2020). Microgrids require substantial financial investments which stand to benefit from a cost-effective EMS to be long-lasting. However, studies on EMS for emerging countries are limited.

Considering LF, an important EMS module; it has remained relevant since the inception of the electric grid. Failure to perform LF may cause substantial financial losses to the electric utility operators as well as dismay customers. The LF is a problem of determining future electricity consumption from current and past time series consumption data taken at a sampling rate of either seconds, minutes, hours, days, months, or even years; depending on the forecast horizon – whether it's short-term, medium, or long-term. Load demand forecasting is vital for power system operations such as unit commitment, maintenance and expansion planning, electricity market operations including ancillary services, and spinning (rapid) reserves (Ghiasi *et al.*, 2018). Accurate LF provide useful information on required resources and materials such as fuels required to operate the generating plants as well as other resources ensure uninterrupted and profitable generation and distribution (Anwar *et al.,* 2018). In most cases, the target of LF is to minimize the total cost of power generation while meeting electricity load demand. Most electric grid networks are aging with the lifespan being extended; accurate load forecasts become crucial (Hong & Fan, 2016). Due to environmental concerns, policymakers are advocating the penetration of more Renewable Energy Sources (RES) into the grid, therefore, adding to the need for accurate load forecasts.

Despite modern power systems being more complex on account of RES and electric vehicles, advances in computing and the large-scale rollout of smart meters and sensors in high-income countries resulted in the collection of vast amounts of grid-related data required by Artificial Intelligence (Ai) and Machine learning forecast algorithms (Abera & Khedkar, 2020; Oprea & Bara, 2019a; Zhou & Brown, 2017). Electric power systems in emerging countries, specifically, in sub-Saharan Africa (SSA) are lousy and stand to benefit from load forecasts since the installed generation capacity is typically small to meet growing demand. Although load forecast is critical to grid operations as already mentioned, there are few studies in the literature on load forecast and blackout forecast in emerging countries especially African countries.

Most emerging countries in the SSA suffer from frequent blackouts, most times than not these blackouts result from load shedding by the utility grid operators. When there is a shortage of generation power some customers get disconnected from the grid. Blackouts may also be due to natural disasters or failure of some equipment in the grid due to poor maintenance. Blackout forecasts help utility companies to take measures in advance of containing an imminent blackout. A blackout forecast can act as an early warning to end users to mitigate the effects of a power outage including making arrangements for alternative backup power and shifting loads (activities) to the time when power is available. A blackout forecast may assist in battery management strategies such as ensuring batteries have enough charge prior to the blackout.

Therefore, a useful EMS will study consumer energy usage habits/patterns to make future predictions (load forecast) whereas the desire to know electric power availability or supplier generation patterns results in a blackout forecast. In a way, LF and BF are two sides of the same coin addressing power demand and availability. The work by Gou and Wu (2008a), addressed prediction of only cascading blackout and the inherent challenges. Kogo *et al.* (2014), Performed a demand-side blackout forecast using only a similar days approach for persistently scheduled blackouts in emerging countries. In the study by Nateghi *et al.* (2014a), the authors conducted a blackout forecast focusing only on hurricane-induced blackouts in a mature grid. In the same vein, the authors Papic and Ciniglio (2014), conducted a study focused on the prediction and prevention of cascading blackouts using a mature grid case study. Majority of the works on blackout focus on the analysis of cascading blackout causes, aftermath, and prevention (Alkar *et al.,* 2019; Bo *et al.,* 2015; Papic *et al.,* 2018; Rahman *et al.,* 2016; Sforna & Delfanti, 2006).

An indispensable element of EMS is reporting via an energy dashboard or Graphical User Interface (GUI) either to the operator or consumer. Smart meters have been instrumental in this aspect; they record consumers' power usage at many intervals per day and can be accessed remotely, this being a sharp contrast to the classical meters which were read manually say once a month. Thanks to IoT, data from smart meters can be displayed to end users thus raising awareness and promoting energy efficiency. Modern lifestyle has seen smartphones go from luxury goods to life necessities with people becoming inseparable from their internet-connected smartphones. Smartphones have also received wide adoption even in emerging countries. This, in turn, is creating an opportunity to engage more users in an EMS scheme aimed at increasing energy efficiency through raising awareness of energy consumers and promoting responsible energy usage. In effect, several researchers have attempted to integrate a combination of IoT devices as part of Demand Side Management (DSM) and smart meters for monitoring and control applications in EMS. The positive effect of energy usage monitoring and reporting should not be undervalued or easily dismissed, as it shapes habits; habit drives consumption. Energy savings due to EMS energy dashboards and smart meter compounds.

As it is also common practice to mix renewable energy and non-renewable energy, which requires an EMS to optimize and efficiently manage the amount of energy generated and stored (Serna-Suarez *et al.,* 2015). There is always room for improvements in load forecast and blackout forecast algorithms. There is a specific direction for potential improvements in the spread of errors, interpretability of errors; enhancing the simplicity of the forecasting process by reducing the requirement of resources (data, hardware, and labor) this will be attractive from the business vantage point (Hong & Dickey, 2015). This research aims at developing an EMS

for hybrid microgrid energy systems based on multiple energy sources: renewables and non-renewable, storage, and user loads. The focus will be on LF, BF, and user engagement aspects of the EMS.

Granted, there may be no single panacea EMS for rural and low-income countries scenarios due to the uniqueness of various regions; this dissertation attempts to develop an EMS that takes into account that low-income regions may not have required financial resources for large renewable systems and that if these renewable power systems are installed, they may be connected to unstable grids plagued at times by frequent power outages. So far, there have been few works on LF and BF for low-income countries, especially in the SSA region. The research presented in this dissertation aims at filling the above.

1.2 Statement of the Problem

The majority of people in rural areas lack the technical expertise to handle and care for the installed sophisticated and expensive microgrids, thus, prompting EMS inclusion in hybrid microgrids for efficient and optimum energy management which also eliminate human error. On top of that, if RES systems are opted, they are usually small-sized solar-based with Lead-acid battery technology (which has less energy density than Lithium battery) due to financial limitations. Therefore, an EMS system becomes necessary to maximize the installed system's throughput. Submodules of EMS may include load forecast, blackout forecast, and energy monitoring module. There is no perfect short-term load forecaster (Hong & Fan, 2016). Research shows that there is room for improvements in the accuracy of forecasting models, which are also influenced by the geographical location of microgrids (Martínez-Álvarez *et al.,* 2015; Su *et al.,* 2017).

In recent years, there has been numerous advancement and discovery of new bio-inspired AI-based forecasting algorithms which have been found to have better performance than statistical-based models (Alkhathami, 2015; Shi & Li, 2017). Ensembled or hybrid models are the current hotspot of forecasting research since they result in higher accuracy and performance (Takiyar & Singh, 2015; Khann *et al.,* 2016). In the open literature, there have been few or no works on EMS load forecasting models studies for developing countries with hybrid islanded microgrids or microgrids connected to unstable main grids such as is the case in Tanzania (Ainah & Folly, 2015; Mir *et al.,* 2020; Williams, 2017).

Weak grids of low-income countries are plagued by more frequent blackouts than mature grids of high-income countries, however, there are few works on the matter, in fact, works on blackout forecast are fewer than works on load forecast. There is a research gap in demand-

side blackout studies. Microgrid investments initiatives may consist of relatively complex and expensive components, and if coupled with load forecast, blackout forecast, and battery management the sophistication increases significantly, therefore, it is necessary to put in place an EMS energy monitoring and dashboard that gently interacts with a layman despite the sophisticated algorithms running behind the scenes, one that can in effect act as a microgrid management coach, trainer or digital assistant. In turn, this can contribute to the realization of the full potential and lifespan of the installed microgrid apparatus consequently averting the grid's premature collapse. This gap beckon to be bridged.

1.3 Rationale of the Study

Electricity is a driver of society's prosperity and livelihood. However, it's a costly affair to operate reliably quality electricity. A good analogy to the electricity business industry is the restaurant business, whereby just the right amount of food needs to be produced to satisfy incoming customers. Underproduction of food results in loss of revenue, while overproduction is problematic, resulting in more investment in refrigeration units. Likewise, for profitability, it's essential to maintain an equilibrium between electric energy generation and demand. Therefore, load forecasts and blackout forecasts have to be made in advance by EMS to mitigate losses. Load forecasts are vital for electric power planning, which may also involve: managing generation capacity, scheduling, peak reduction, and market evaluation. Electric energy forecast is a moving target, due to consumers evolving lifestyles, technological advancement, shift towards electric vehicles, population growth, and climate change. All these factors continue to influence electric energy forecast while incentives and room for improving forecast accuracy exist.

Some regions in SSA have utilities operating in a black box manner, with limited or no grid information available to the end user. Electric energy consumers in emerging countries are usually forced to have electricity backup systems since they are affected by frequent power outages. The outcome of the blackout forecast could be used in the implementation of a battery management system strategy that ensures an adequate state of charge (SoC) when a blackout is imminent and allows the battery to fully discharge when a blackout occurrence is unlikely. Consequently, money could be saved if batteries are not oversized, and the SoC is optimized with respect to blackout forecast output. The blackout forecast in this study could also be useful to the utility for load shedding and demand side management applications.

In the literature, studies on blackouts have not been fully exhausted. There are limited works on blackout forecast studies for emerging countries' scenarios. Emerging countries' power grids

give an interesting research study focus because they are weak, still evolving, and are typically characterized by frequent disturbances as opposed to mature grids of developed countries. Moreover, due to scattered settlements in emerging countries like Tanzania, it is not feasible yet for the national main power grid to reach all communities, therefore, most emerging countries embark on rural electrification programs which employ hybrid microgrids that in time get connected to the main power grid. Consequently, microgrids that connect to the main grid have to contend with disturbances in the main grid. This work attempts to address and bridge this gap with a blackout forecast model using a case study from Arusha-Tanzania. This work studies blackout forecast from the customer's perspective, at the end user's side (premise) of a weak grid.

There is a Swahili proverb that may be translated as follows: *"Assets or resources with no accounting, vanish without a trace"*. Electric energy is a precious resource that must be monitored and utilized wisely and frugally out of environmental concerns. Some emerging countries including Tanzania have launched and promoted rural electrification campaigns, in some areas these involve having hybrid microgrids with different sources of energy including renewables. This necessitates incorporating energy forecast systems in the microgrid EMS. Low-income countries such as Tanzania did not follow the same path in adopting telecommunication infrastructure, it leapfrogged into mobile telephones, for the most part skipping landlines interconnectivity. Similarly, as microchips and sensors continue to decrease in cost, researchers must conduct case studies in low-income countries on low-cost EMS smart-meter technologies in order to get the ground truth and the right solutions that work in low-income countries setting. This study strives towards that aim; Thus, this study is not only important to the scientific community but also the electric power consumers and distributors in the emerging countries' weak grids as well as stakeholders in mature grids.

1.4 Research Objectives

1.4.1 Main Objective

To develop EMS with electric energy forecasting and energy monitoring, for use in microgrids.

1.4.2 Specific Objectives

(i) The identification of machine learning techniques suitable for EMS in hybrid microgrids.

(ii) To develop a short-term load forecast model for use in emerging countries' microgrids.

(iii) To develop a short-term blackout forecast model for microgrids connected to weak grids.

(iv) To develop and implement a practical low-cost EMS system with energy monitoring and user engagement.

1.5 Research Questions

(i) Given a hybrid microgrid with renewable sources, what is the optimal load forecast model required to better manage resources if the microgrid is connected to a weak grid?

(ii) Given a hybrid microgrid with renewable sources, what is the optimal blackout model required to better manage resources if the microgrid is connected to a weak grid?

(iii) How can a short-term blackout forecast model for microgrids be connected to weak grids?

(iv) How can a practical low-cost EMS system with energy monitoring and user engagement be developed and implemented?

1.6 Significance of the Study

The main contributions of this research are as follows:

(i) A load forecast model development framework is presented. The load forecast model given is based on K-Features mean score, union, and refined exhaustive search feature selection approaches. This work proposes the combination of more than one feature selection model to increase performance at the prediction stage of the load forecast procedure. Thereafter, applying a grid search algorithm on a set of prediction models followed by a fusion of two best performing models based on an error metric such as MAE, to form a hybrid model with better performance than the two constituting individual models. The hybrid prediction model is formed from the elementwise maximum (mean or minimum) forecast instances of two regression models. Quality load forecasts save costs in microgrid operations.

(ii) A power outage short-term prediction model is proposed. The proposed blackout model is a hybrid based on the random forest (RF) model and adaptive similar day model (ASD) that performs prediction from the consumer vantage point instead of the utility standpoint. The blackout short-term prediction challenge was solved as a regression

problem as well as a classification problem. The applicability of the model was tested using a real case study.

(iii) Development of a low-cost EMS smart meter and energy dashboard for promoting energy use awareness as well as user engagement. The prototype is implemented using open-source IoT components and technologies to provide energy monitoring and control, visualization, and reporting suitable for low-income countries' scenarios. The applicability of the prototype was demonstrated in a real case study.

1.7 Delineation of the Study

This study aimed at investigating energy forecast and monitoring modules of EMS in emerging countries' scenarios. Electric load forecast methods and power blackout forecast methods are discussed using two case studies from Arusha. Additionally, an experimental low-cost smart meter and energy dashboard are given. The study uses data from Ngarenanyuki secondary school Microgrid based in Arusha-Tanzania which is composed of pico hydropower, PV-battery system with a backup diesel generator to validate the developed EMS model. The school is also connected to the national grid (TANESCO) which is unstable. The study also uses data taken from a pico-PV-battery grid-tied system at a pilot site in Levolosi ward also in Arusha. Therefore, this research develops EMS suitable for both off-grid and grid-tied operating conditions of emerging countries scenarios.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Recent years have seen a continued rise in energy crisis driven by industrialization and consumerism which in turn have contributed to the worsening global warming and climate change. One way to mitigate the energy crisis and protect the environment has been by increasing energy efficiency, lowering the use of fossil energy sources, encouraging renewable energy sources (RES), and energy management systems (EMS). This chapter highlights machine learning approaches for load forecast, blackout forecast, and viable approaches and considerations in implementing EMS in emerging countries.

This chapter presents a review of EMS concepts relevant to the implementation of microgrids in low-income countries and leans specifically towards EMS deployed on the demand side of the grid rather than the utility's side. To this end, first EMS Demand Side Management (DSM) is reviewed to introduce various strategies used to manage energy from the consumer's side. Secondly, selected machine learning approaches adopted by researchers for load forecast and blackout forecast are given. Lastly, smart meter and user interaction technologies for EMS are reviewed.

2.2 Energy Management Systems Overview

Energy management systems is computerized energy dashboard used as a portal to monitor, manage, and track electrical energy consumption in microgrids. The EMS help grid operators lower operation costs, make a more informed decision, detect anomalies, and also aid in planning. The EMS continues to attract the attention of more research in pursuit of increased energy efficiency and optimization. The EMS are vital elements in *Smart Grids* since consumers are increasingly becoming active consumers (prosumers). In the traditional electric grid, consumers are oblivious to their wasteful energy usage patterns, this issue is even worse in emerging countries. However, smart grids through the use of technologies like Internet of Things (IoT) can help raise consumers' energy consumption awareness and boost energy efficiency. Smart Grids are characterized with two-way power flow as well as two-way signal communication infrastructure. It's difficult to control or optimize that which hasn't been measured or monitored. Therefore, there is a need to study appropriate technologies for use in EMS in emerging countries and rural areas. Load and blackout forecasts are among the key organs of EMS. Research works on load forecast and blackout forecast are few, especially for

weak grids in emerging countries. The EMS is an umbrella term that may incorporate a combination of the following specialized functions or variations: Demand Side Management; Battery Management System (BMS); Building Energy Management System (BEMS); Smart Home Energy Management System (SHEMS).

The DSM involves controlling the consumer's energy use habit or quantity by the utility company, in other words, all actions taken at the customer's premise to lower energy use or increase energy efficiency. Traditionally utility companies aim to reduce peak hour customer loads to save cost and avoid activating fossil fuel-based generation sources to meet peak hour demand. To achieve these various DSM schemes are employed such as: (a) Demand Response (DR) through financial incentives (Time of use tariffs - ToU) of having cheaper energy tariffs during off-peak periods (valley filling), thus using less energy during peak periods (peak clipping); (b) encouraging the use of energy-efficient appliances effectively reducing energy used to perform the same task; (x) Distributed Energy Resources (DER). The preceding DSM approaches are typically facilitated by monitoring energy use through EMS. Figure 1 shows the main aspects involved in DSM.

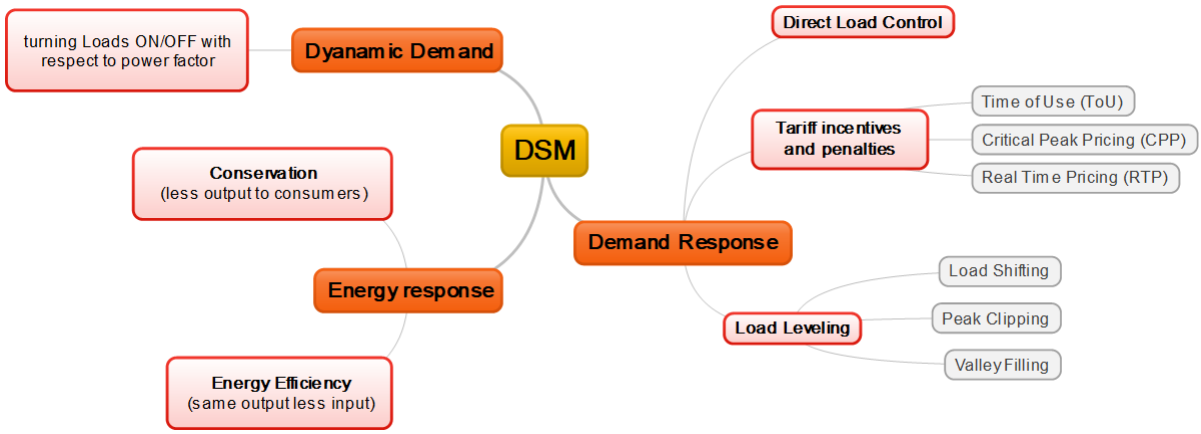


Figure 1: Demand Side Management components

Electric loads may be grouped into two categories namely deferrable (controllable) loads or non-deferrable (non-controllable) loads; depending on the application they can also be household (residential) loads or commercial loads (Afzalan & Jazizadeh, 2020; Moradzadeh *et al.*, 2021). At times in microgrids, there may not be enough power to sustain all the loads forcing some loads to be disconnected – this is called *load shedding*. Although load shedding is practiced by utility companies and results in blackouts, the concept can still be applied in smaller microgrids serving households, buildings, or communities by disconnecting loads based on a priority rule (Alahmed & Al-Muhaini, 2020). Under this mode of handling power

shortage, critical loads are given priority over non-critical loads. Therefore, all the loads are assigned priority levels to help manage the loads during instances when power isn't enough to be allocated to all the loads. Fuzzy logic can be used in decision-making regarding which loads to be shed with respect to available power level thresholds (Laghari *et al.,* 2018). Fuzzy logic follows generalized Boolean logic with ranges of values. It is useful in applications where rough estimates are acceptable. Inputs are simply assigned to a neighbourhood of values without needing to be precise. For example, under fuzzy logic, the pressure of a system could have five levels, namely very low, low, medium, high, and very high. One of the important inputs in EMS for buildings, homes, or institutions is occupancy detection, which then aids in load control decisions (Yang *et al.,* 2016; Zhao *et al.,* 2015; Zou *et al.,* 2017).

Ngarenanyuki microgrid (one of the sites for this study) has already been described in the literature: One study was done comparing AC and DC bus configurations, control strategies, reliability, and efficiency before the implementation of the architecture in Fig. 6 (Carmeli *et al.,* 2014). Another study reported an overview of the typical configuration and economic models of batteries in the off-grid system and describes the application of batteries in the Ngarenanyuki school grid experimental project (Mandelli *et al.,* 2015). Carmeli *et al.* (2015) performed an analysis of the school's actual power supply system prior to deployment of the architecture in Fig. 6 as well as a simulation of operations and dynamics of the architecture. Mandelli *et al.* (2015) examined the school's consumption pattern, the simulation of the electro-mechanical operation, and the power flow from generators and loads. Nyari *et al.* (2017) studied the consumption of electrical appliances in stand-by and active operation states. A Matlab-based stochastic procedure that allows to generate load profiles of microgrids in small communities was developed by Mandelli *et al.* (2017), and validated with Ngarenanyuki microgrid data. Mauri *et al.* (2016), developed a neural-fuzzy EMS for the Ngarenanyuki school grid. Ngarenanyuki microgrid started as an off-grid system up until it was connected to the main grid in late 2016.

2.3 Machine Learning Approaches for Load Forecast and Blackout Forecast

In accordance with Moore's law, computational power has seen an exponential increase, along with-it propelling advancements in Artificial Intelligence (AI) and Machine Learning (ML). AI is a much broader term with ML as one of its subsets. The ML is data-driven. More researchers continue embarking on the quest of refining accuracy for the elusive load forecast and blackout forecast due to their stochastic nature. This is partly because factors such as consumers' lifestyle, population growth, economy, weather, and climate don't remain static. Thanks also to smart meters and IoT platforms, it is becoming easier to obtain the vast amount

of unprecedented data (big data) needed to train ML algorithms. However, effective management of the power grid is a complex and costly affair demanding adequate electricity supply, despite the presence of intermittent RES in the generation mix. This calls for quality load forecast and blackout forecast models for proper load demand planning and generation scheduling. It can be argued that ML is an art involving heuristics. A forecasting model that works well in a given dataset, won't perform well in all types of datasets or scenarios.

2.3.1 Load forecast

No gold standard exists for classifying load forecast techniques (Hong & Fan, 2016). However, an attempt could be made to classify Load Forecast (LF) techniques into three broad groups, namely correlation (engineering method), extrapolation (data-driven), and a combination of both (Nti *et al.,* 2020). Correlation techniques involve performing LF with respect to factors related to the economy and demography such as population, HVAC, weather, employment, business, building permit, and building structure. Whereas, extrapolation techniques involve performing LF estimation based on historical data (time-series) trend analysis (Nti *et al.,* 2020). It is also common for researchers to group LF methodologies into classical statistical or mathematical models, AI/machine learning models, and hybrid models (Hong & Fan, 2016; Vivas *et al.,* 2020). If only historical time-series data is used for LF then it is referred to as univariate, otherwise, if other exogenous data such as weather data are included in LF then it becomes multivariate (Mir *et al.,* 2020; Vivas *et al.,* 2020). Depending on the target application LF can be performed in different intervals or horizons. According to forecasting intervals, LF is typically grouped into four groups namely, long-term load forecast (LTLF), medium-term load forecast (MTLF), short-term load forecast (STLF), and very short-term load forecast (VSTLF) (Hong & Fan, 2016; Mamun *et al.,* 2020). The VSTLF spans a few seconds to a few minutes and is used for distribution schedule and generation forecasting applications. The STLF covers forecasts from a few minutes up to 1 day ahead for spinning reserves allocation and maintenance schedule. Deregulation of electricity distribution and penetration of RES has caused an increase in STLF since daily market prices are affected by RES (Gasparin *et al.,* 2022). The MTLF usually covers a few days to a few months for seasonal LF planning whereas, LTLF covers 1 year to a decade for generation capacity investment growth planning. Generally, forecasting error is dependent on the horizon; the shorter the horizon the shorter the forecast error (Vivas *et al.,* 2020).

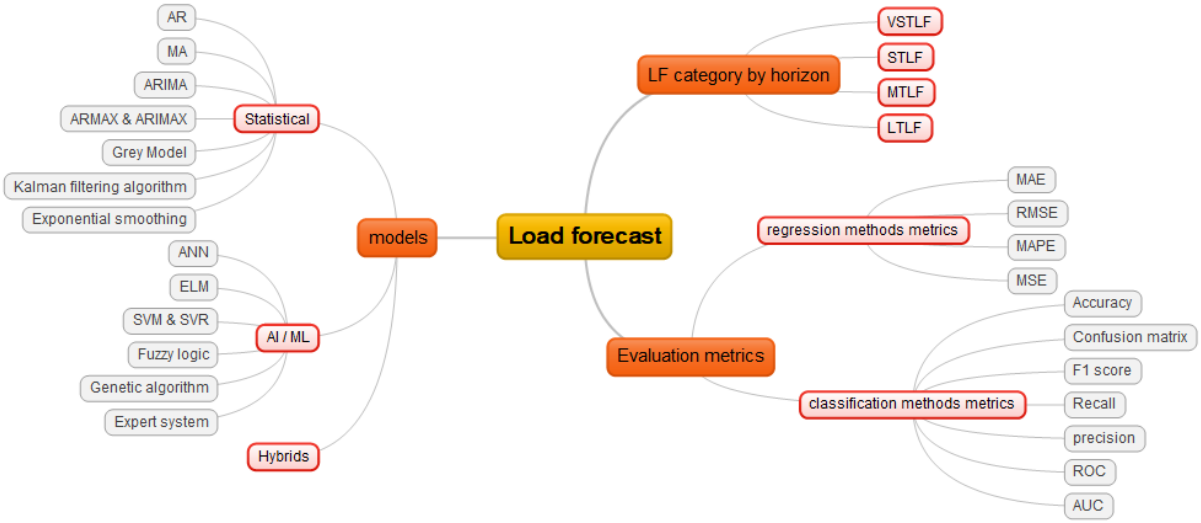


Figure 2: Load Forecast algorithms categories

2.3.2 Load and Blackout Forecast Model Development Process Overview

Collected data for load forecast and blackout forecast, initially needs to be pre-processed. Pre-processing ensures that input data is noise-free and in the correct format. This may involve data cleaning through the detection and removal of outliers, data exploration, feature selection, and data structure transformation. Data exploration involves visualising data distribution to gain useful insights into the nature of the prediction problem in question. Input data may have many input variables (features), which may not all be needed to develop an ML model. Since not all input variables influence the prediction ability of the model while others are redundant in their effect on model prediction. Thus, the need for feature selection. A number of techniques have been reported in the literature for feature selection (Eseye *et al.*, 2019; Pourdaryaei *et al.*, 2021; Subbiah & Chinnappan, 2020).

The next stage is to split the dataset into a training set (70% of data is used), a validation set (15% of the dataset), and a test set (15% of the dataset). The splitting ratio could be different depending on the application. However, reserving some of the data for validation is important to avoid the overfitting (bias) phenomena and ensure unbiased generalization of the model (Srivastava *et al.*, 2014; Ying, 2019). The test set is used to gauge the trained model's accuracy. At this stage inputs from the test, set are fed into the trained ML model to give out predicted outputs that are finally compared with actual output data from the same test set used. The difference (error index) between the predicted data and actual (observed/expected) test data is computed to decide against retraining, accepting, or discarding the trained model. There is a number of performance evaluation metrics used by most researchers in the literature for assessing the error index of trained regression models. The topmost error index evaluation metrics for regression models are mean absolute error (MAE), mean absolute percentage error

(MAPE), mean square error (MSE), and root mean square error (RMSE). Error metrics for classifiers are confusion matrix, accuracy score, precision score, F1 score, recall, and receiver operating characteristic area under the curve (ROC-AUC) (Keszocze *et al.*, 2018; Singla *et al.*, 2021; Singleton & Grindrod, 2021).

Machine learning can either be considered supervised learning or unsupervised learning. This stems from the way the algorithm works during training. Under supervised learning, a model is supplied with both input predictors as well as the corresponding outputs (targets/labels). Unsupervised learning algorithms don't require labels or targets. Clustering algorithms such as the K-means algorithm belong to this category. Generally, most electricity load and blackout forecast problems in ML can be solved as either a regression problem or a classification problem (Oprea & Bara, 2019b; Madrid & Antonio, 2021; Nystrup *et al.*, 2021). Regression algorithms use models that extrapolate ordered continuous output vector (sequence data) from the input data, whereas classification algorithm map input variables to unordered discrete categorical output (response/label). Figure 3 shows a typical machine learning workflow from literature (Bildirici & Özaksoy, 2016; Kim *et al.*, 2020; Mamun *et al.*, 2020; Román-Portabales *et al.*, 2021).

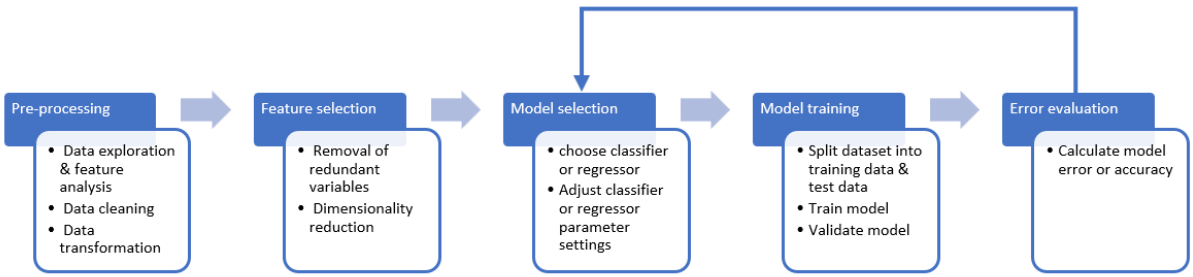


Figure 3: Typical machine learning workflow

2.3.3 Load Forecast Models

Principles of LF can also be applied to blackout forecasts as well as other utilities such as water, and gas (Hong & Fan, 2016). The ML models have gained much attention in the past few decades partly due to advancements in computation power and to optimize the aging electricity grids. This section will give an overview of only some of the algorithms most relevant to LF, there has been an increase in hybrid ML models in the literature. Usually, in ML, the goal is to come up with a relation (model/expression/equation) that maps inputs to corresponding outputs (targets). According to Mamun *et al.* (2020), widely used STLF algorithms are namely Expert Systems, Time-Series-based forecast, Support Vector Machines (SVM), Artificial Neural Networks (ANN), Regression-Based Approach, Similar Day Approach, and Fuzzy logic.

Linear regression models employ a linear relation between a set of inputs to corresponding outputs. The goal of the algorithm during model development (training) is to compute coefficients and intercept the linear expression governing linear regression (Dudek, 2016). Some researchers opt to use multiple linear regression because it can handle the many variables in LF (Dhaval & Deshpande, 2020). Logistic regression is based on the logistic (sigmoid) function whereby the output is a probability bound between 0 and 1. It is suited for binary classification challenges (Fuks & Salazar, 2008; Saxena *et al.,* 2019; Wang *et al.,* 2018).

Classical statistical LF techniques include grey model (GM), Kalman filtering, and exponential smoothing (ES). Classical models also include the Box-Jenkins models namely autoregressive models (AR), autoregressive moving average (ARMA), autoregressive-integrated moving average model (ARIMA), and autoregressive-integrated moving average model with exogenous inputs (ARIMAX) (GeP *et al.,* 2008; Hammad *et al.,* 2020). These were the first breed of models to be employed during the early days of LF. The main weakness of the aforementioned classical LF techniques is that they assume the observed system to be linear. ML models such as artificial neural networks (ANN) or simply neural networks (NN), support vector machines (SVM), and decision trees have gained much attention because they are more suited for non-linear challenges like LF. In Mbuli *et al.* (2020), authors performed an overview of decomposition LF; these methods decompose time series data into 3 components: Trend and seasonality, deterministic components, and stochastic irregular components part of time series data. Although decomposition methods are less accurate than other ML models, they can sometimes be preferred since they are easy to model, understand, and use (Mbuli *et al.,* 2020).

The ANN is bio-inspired by the way the human brain learns, and attempts to mimic brain neuron operations in solving computational problems (Aslam *et al.,* 2021). The ANN is typically comprised of three layers: an output layer, a hidden layer, and an input layer. The NN layers have activation functions, adjustable biases, and weights (Aslam *et al.,* 2021). Earlier versions of NN used in LF were multiple layer perceptron (MLP) which were essentially feedforwarding NN. Later recursive neural networks (RNN) which are more sophisticated than MLP were adopted for LF (Gasparin *et al.,* 2022). Long short-term memory (LSTM) neural networks are derivatives of RNN which have gained more popularity in LF research due to their good performance (Elahe *et al.,* 2021). Other types of NN also widely used for LF are the convolutional neural networks (CNN).

Deep learning (DL) is a subfield of ML which uses deep neural network models. The term ‘deep’ alludes to the fact that they have many layers of non-linear hidden neurons and a very large output layer that enables these models to adapt to learning new patterns (Bourdeau *et al.,*

2019). The DL models are also gaining the interest of LF researchers (Elahe *et al.,* 2021; Gasparin *et al.,* 2022). The DL models require larger datasets and more computational power than shallow learning models, NN models with one hidden layer (Aslam *et al.,* 2021). The ANN has been combined with other models to form hybrids for LF, for example, ANN has been combined with a genetic algorithm (GA) resulting in ANN-GA. In this case, GA provides selection, crossing, and mutation operations which can be used to optimize the ANN parameters such as weights and bias (Islam *et al.,* 2014). In Pai and Hong. (2005), GA was used to determine free parameters for the SVM used in LF.

The k-nearest neighbour (k-NN) is a commonly used technique for clustering and classification. It can be used to group similar patterns in electricity load time series data (Bourdeau *et al.,* 2019). Ensemble models are formed by weighing the accuracy of two or more models and combining them to form a model that gives optimal performance. Lopez-Martin *et al.* (2021), compared a number of ensemble and deep learning models in STLF using data spanning 3 years from the Spanish utility. Decision trees are a class of ML algorithms that use tree-like structures to split features in a dataset through a cost function from the root to the leaves. At each node of the tree, a condition is checked between binary or categorical values until arriving at the end decision leaf nodes which represent the target output (Breiman *et al.,* 2017). Random forest (RF) is a supervised ML algorithm made by growing several decision trees on the samples and taking their majority vote or average for the case of classification and regression, respectively (Breiman, 2001). Support vector machines (SVM) are commonly used for solving nonlinear problems, they were first introduced by Cortes and Vapnik (1995). They can give good performance even for relatively small datasets (Bourdeau *et al.,* 2019). An SVM specific for regression problems is the support vector regression (SVR) method (Ribeiro *et al.,* 2022).

Zhang *et al.* (2021), performed a review of ML in building load prediction while considering different algorithms, their applications, and data. They also suggested a standardized framework for building load prediction. Mir *et al.* (2020), gave a review of LF focusing on select low- and middle-income countries over mostly work employing LTLF, followed STLF, and fewer works on MTLF. The role of LF input data such as GDP, population, weather, and load data over different horizons was highlighted. Román-Portabales *et al.* (2021), carried out a review of ANN-based ML for LF whereby the LSTM model was found to offer good performance in most STLF papers considered.

The work by Mamun *et al.* (2020), gives an extensive overview of the mechanisms, merit, and disadvantages of mainstream (ANN and SVM) LF techniques. Contemporary hybrid LF

algorithms were described including SVM and GA, SVM and firefly algorithm (SVM-FA), SVM and fruit fly optimization algorithm (SVM-FOA), SVM and particle swarm optimization (PSO), SVM and harmony search (SVM-HS), SVM and artificial bee colony (SVM-ABC), SVM and simulated annealing (SVM-SA); ANN-FOA, ANN-GA, ANN-FA, ANN, and clustering technique (ANN-CT), ANN and neural fuzzy inference system (ANN-NFIS), ANN and artificial immune system (ANN-AIS), ANN and wavelet transform (ANN-WT). Future research trend appears to be headed for hybrids of 3 or more algorithms (Mamun *et al.*, 2020).

In the work by Nti *et al.* (2020), the authors performed a review of LF models by load type – commercial, residential, and combined; the forecast model – conventional, AI, and hybrids; evaluation metrics; and forecast horizon. The authors found that 90% of the studies they reviewed used AI methods; 10% were statistical methods; there were few studies on LF in African countries; residential LF had received little attention.

Table 1 shows a comparison of widely used load forecast models. The ARIMA models can be useful for load forecasting when the data exhibits certain stationary patterns, but they may struggle with nonlinear relationships and complex seasonal patterns. The ANNs are widely used for load forecasting due to their ability to capture complex nonlinear relationships — capturing intricate patterns in load data but may require more computational resources and data pre-processing compared to SVM. The RF can handle nonlinear relationships, variable interactions, and seasonal patterns in the data. The RF are known for their robustness and can handle large amounts of data. The LSTM can effectively model sequential data, making them suitable for time series forecasting tasks. The LSTMs can capture long-term dependencies and handle varying sequence lengths, which can be beneficial for load forecasting. They often outperform traditional methods like SVM and ARIMA in scenarios where temporal dynamics and historical patterns play a significant role. Given a dataset, a systematic search for a suitable algorithm needs to be conducted. A proposed approach for identifying a suitable algorithm for STLF is discussed later in Chapter 3.

Table 1: Comparison of selected key energy forecasting algorithms from the literature

Algorithm	Characteristics	Suitability for energy forecasting	Dataset type and size	Selected references
AR (Autoregressive)	Uses past values of the variable to forecast future values.	Suitable for capturing short-term dependencies and trends in energy data.	Time-series data, and small datasets	Baharudin and Kamel (2008)
Moving Average (MA)	Uses the average of past values to forecast future values.	Suitable for smoothing out short-term fluctuations in energy data.	Time-series data, and small datasets	(Ogunjuyigbe *et al.* (2021), Yin *et al.* (2022)
Autoregressive Integrated Moving Average (ARIMA)	Combines autoregressive and moving average components to handle trends and seasonality.	Suitable for capturing both short-term and long-term dependencies and seasonal patterns in energy data.	Time-series data, and mall to large datasets	Kapoor and Sharma (2018)
Autoregressive Moving Average with Exogenous Variables (ARMAX)	Extends ARIMA by incorporating exogenous variables in the model.	Suitable for incorporating external factors, such as weather or economic indicators, into energy forecasting.	Time-series data with exogenous variables, and small to large datasets.	Li *et al.* (2014)
Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX)	Similar to ARMAX but includes an integration component for non-stationary data.	Suitable for handling non-stationary energy data while considering exogenous variables.	Time-series data with exogenous variables, and small to large datasets.	Sheng and Jia (2020)

Algorithm	Characteristics	Suitability for energy forecasting	Dataset type and size	Selected references
Grey Model	Uses a small number of data points for forecasting and relies on data transformation.	Suitable for medium-term and long-term energy forecasting with limited historical data available.	Time-series data, and small datasets	Zhang and Wu (2021), Zhao *et al.* (2022)
Kalman Filtering Algorithm	Recursive estimation algorithm that updates predictions based on new data.	Suitable for real-time energy forecasting and tracking dynamic changes in energy data.	Time-series data, and small to large datasets.	Sharma *et al.* (2020) Takeda *et al.* (2016), Zhang *et al.* (2021)
Exponential Smoothing	Gives more weight to recent observations and smooths out fluctuations.	Suitable for short-term energy forecasting, particularly when the data has no clear trend or seasonality.	Time-series data, and small datasets	Ji *et al.* (2012)
Random Forest	Ensemble learning method that combines multiple decision trees.	Suitable for capturing complex relationships and interactions among variables in energy forecasting.	Time-series data with exogenous variables, and small to large datasets.	Dudek (2022), Subbiah and Chinnappan (2022)
Expert Systems	Rule-based approach, incorporates domain knowledge.	Suitable for incorporating expert knowledge and rules specific to energy forecasting.	Any type of relevant data, small to large datasets	Lahouar and Ben Hadj Slama (2015)
Similar Day Approach	Compares the current day with historical days having similar patterns.	Suitable for short-term energy forecasting based on similar historical patterns.	Time-series data, and small datasets	Dudek (2015a, 2015b), Huang *et al.* (2019)

Algorithm	Characteristics	Suitability for energy forecasting	Dataset type and size	Selected references
Neural Network	Utilizes complex mathematical models and learns patterns from training data.	Suitable for capturing nonlinear relationships and complex dependencies in energy data.	Any type of relevant data, and small to large datasets	Agana *et al.* (2018), Akhil-Srinivas *et al.* (2021)
Extreme Learning Machine (ELM)	Randomly assigns input weights and solves a linear system to obtain output weights.	Suitable for fast training and accurate forecasting in energy data.	Any type of relevant data, and small to large datasets	Cheng *et al.* (2013), Bin *et al.* (2012), Kunqiao and Jiandong (2021)
Support Vector Machine (SVM)	Uses statistical algorithms to find optimal decision boundaries.	Suitable for both short-term and long-term energy forecasting with good generalization capabilities.	Any type of relevant data, and Small to large datasets	Amin and Hoque, (2019), Khan *et al.* (2018)
Fuzzy Logic	Deals with uncertainty and imprecision in data through linguistic variables and rules.	Suitable for handling vague or incomplete data and incorporating expert knowledge.	Any type of relevant data, and small to large datasets.	Anoop and Kanchana (2018), Mukhopadhyay *et al.* (2018)
Genetic Algorithms	Evolutionary optimization algorithm inspired by natural selection and genetics.	Suitable for optimizing model parameters or feature selection in energy forecasting.	Any type of relevant data, and small to large datasets.	Chaturvedi (2008)

2.3.4 Blackout forecast

Access to electricity remains an issue yet to be resolved in sub-Saharan Africa (SSA), where 600 million people don't have access to electricity, this being nearly half of the population (IEA, 2014; International Energy Agency, 2019). Reliable and quality access to electricity boost livelihood and drives forward the economy (Falentina & Resosudarmo, 2019). According to a 2019 survey conducted by the Tanzanian rural energy agency (REA), it revealed that only 37.7% percent of people on the mainland had a connection to electricity (REA & NBS, 2020). This is in part due to expensive connection costs along with low use of electricity and poverty among countryside communities (Abdullah & Markandya, 2012). Taking Tanzania as an example, even areas with access to electricity at times experience power outages, brownouts, and voltage surges. Many times, electricity demand exceeds supply in the SSA region, therefore load shedding or rolling blackouts – power rationing, becomes essential to prevent the electric grid from failing. However, end-users prefer power cut notifications in advance so that they can plan to mitigate power outage effects (Nkosi & Dikgang, 2018). These notifications usually reach a very small percentage of users, especially if the outage will cover a relatively small neighbourhood. Rolling blackouts hit most, residential sector or poorer neighbourhoods than the industrial sector due to both economic and political factors (Aidoo & Briggs, 2019; Pollet *et al.,* 2015).

Studies investigating blackouts are not new. Blackouts are caused by multifaceted interactions of many factors such as serious line components failure, negligence in handling vegetation along transmission and distribution lines, wildlife, extreme weather, man-made error, and other natural calamities (Gou & Wu, 2008b). The problem of blackout prediction can be attempted either by short-term (real-time) prediction or long-term prediction. Nateghi *et al.* (2014a, 2014b), developed a hurricane-induced blackout prediction algorithm based on a random forest algorithm. Hurricane-induced power outage prediction models are complex and involve large expensive datasets. The dataset may comprise the following input variables: electric grid data, data on the pre-storm situation of soil moisture, drought, land use data, geographical measures, wind data, and so forth. Gou and Wu (2008), classified blackout causes as either deterministic or probabilistic. The study also investigated control strategies with an emphasis on islanding control strategies. The study by Alkar *et al.* (2019), reported that frequent power outages were found to be due to the inefficiency of the power plant to match loads, malfunction of protection equipment in the transmission lines, poor ability of Supervisory Control and Data Acquisition (SCADA), division of the main power grid to microgrids which are more prone to oscillations, delayed equipment maintenance and population increase. The study by Rahman *et al.* (2016), used data of large-scale power outages globally to explore blackout causes, perform a risk

analysis, and fault analysis. In Bo *et al.* (2015), worldwide blackout incidents were analysed, and their causes, mitigation, and restoration measures were investigated. Blackout causes, protection issues, blackout prevention, and blackout restoration have been investigated by some studies (Makarov *et al.*, 2005; Sforna & Delfanti, 2006). In the work of Mei *et al.* (2009), two indices for quantitative blackout risk evaluation were developed.

Cheng *et al.* (2017), combined other outage factors such as real-time electric grid system operation data, weather forecast, and geographical data to predict blackout components in the current electric grid system operation condition. They also developed a load behaviour forecast model under power outage circumstances using an expert fuzzy system. Kogo *et al.* (2014), proposed 3 heuristics to forecast the start time of the next 24 hrs irregular scheduled power cuts namely: Start time of power-cut based prediction (SBP), frequency-based prediction (FBP), and a hybrid of SBP and FBP. Papic and Ciniglio (2014), proposed a framework for supporting planners and operators in evaluating multiple outages that lead to cascading outages. Papic *et al.* (2018), identified two indices for power outage reliability namely: average frequency and average duration of sustained automatic outages. The indices could be employed in forecast-based planning, maintenance, and operation activities. According to the study, the foremost causes of outages were found to be: Weather (rain, snow, ice storms, wind, dust, etc), equipment failure, and wildfire.

The 26th UN climate change conference of the parties (COP26) attests to the fact that climate change is a burning issue globally. Extreme weather events are on the rise, unfortunately, they also affect electricity generation and supply infrastructure reliability, sometimes resulting in curtailments and in some cases rolling blackouts (Chandramowli & Felder, 2014; Chen *et al.*, 2021; Zachariadis & Hadjinicolaou, 2014). Thus, blackouts should continue to be studied. In the literature, studies on blackouts have not been fully exhausted. There have been few or no works on blackout forecast studies for emerging countries' scenarios. Emerging countries' power grids give an interesting research study focus because they are weaker, still evolving, and are typically characterised by frequent disturbances as opposed to mature grids of developed countries. Moreover, due to scattered settlements in emerging countries like Tanzania, it is not feasible for the national main power grid to reach all communities, therefore, most emerging countries embark on rural electrification programs which employ hybrid microgrids that in time get connected to the main power grid. Consequently, microgrids that connect to the main grid have to contend with disturbances in the main grid.

2.4 Smart meter, data logging, and user engagement considerations for EMS in emerging countries

The primary source of data in smart grids is the advanced metering infrastructure (AMI). The AMI involves the deployment of a number of smart meters on the consumer side of the grid. The goal is to save costs and increase energy efficiency. Smart meters usually are characterised by the following features: the ability to measure energy consumption at least in 15 minutes intervals; provide ToU information to users; ability to disconnect/reconnect consumers remotely; control some appliances over Home Area Network (HAN); provide a web portal or dashboard for historical consumption data; provide on-demand energy readings. This is in sharp contrast to the past analogue electric meters which were read manually once a month. In the case of Tanzania, the national electric utility company TANESCO currently provides consumers with prepaid electric meters, although this is likely to change in the future since the trend globally is moving towards rolling smart meters and upgrading to Smart Grid. Evidently, smart meters are important to realize DSM operations in a microgrid. Inevitably, EMS tasks such as load forecasting rely on historical data collected by smart meters into a database.

Some low-cost IoT electronic devices and technologies which could be suitable for hybrid microgrids in emerging countries scenarios include the following: Arduino MCU modules (Uno, Mega, etc), NodeMCU; ESP32; Raspberry pi (2 / 3 / 4); Xbee; Zigbee; LoRA; GSM modules; SD card modules; power measurement modules. Table 2 highlights the preceding common IoT devices. Since one of the objectives of this study was to develop a low-cost EMS smart meter and based on literature survey comparison made in Table 2, Arduino Mega microcontroller was chosen for use in this study because it is relatively cheaper than other options like Raspberry pi by a factor of 10. It also had the needed features such as sufficient GPIO pins, PWM pins, enough ADC pins for interfacing sensors, and serial communication capability. However, it lacked internet connectivity, hence, NodeMCU microcontroller being cheaper than ESP32, it was also included in this study in order to add IoT functionality to the developed low-cost smart meter while facilitating remote monitoring operations. The PV-battery systems are usually one of the sought-after options for power backup systems or off-grid systems in emerging countries as the price for such systems is becoming affordable (Nguyen *et al.*, 2018). The most common battery technology adopted for these systems is lead-acid batteries, although Lithium-ion batteries are increasingly becoming attractive as they possess twice more energy density and lifespan (Dufo-López *et al.*, 2021; Podder & Khan, 2016). Smart meters can also incorporate a pulse-width-modulation (PWM) based battery management system (BMS) to monitor and control battery state-of-charge (SOC) via charging/discharging cycles (Chang, 2014; Qin & Du, 1994).

In the work by Karthick *et al.* (2021), they designed and implemented a low-cost smart meter that: monitored electric energy consumption via the PZEM-004T module; monitored power quality issues (voltage swag, swell, and transient) using the SVM algorithm running on Raspberry pi device; DSM load scheduling of primary and secondary loads via NodeMCU-based smart plugs; Blynk mobile app was used for user engagement (Blynk IoT Platform: For Businesses and Developers, n.d.). In the work by Iqbal and Manzoor (2020), the authors developed supervisory control and data acquisition (SCADA) system that included: A PZEM-004T module for energy measurements; ACS712 hall effect current sensor to measure grid-tied PV-battery current; a voltage divider circuit to convey PV voltage to ESP32 analog to digital converter (ADC); loads controllable by relay interfaced to ESP32 module; Ubidots platform as an EMS dashboard (IoT Platform | Internet of Things | Ubidots, n.d.).

Table 2: Comparison of selected low-cost smart meter implementation from literature

Technology / Device	EMS Role	Merit	Limitation(s)	Selected applications EMS related references
Arduino uno / nano / mega	Direct load control, BMS via PWM.	Low cost, low energy footprint.	Low-cost Arduino boards (e.g uno, mega) lack wireless connectivity, storage for datalogging, computation power for ML.	Batista *et al.* (2013), Kondaveeti *et al.* (2021), Pawar and Vittal (2019)
NodeMCU	Direct load control, BMS via PWM, remote monitoring and control of loads, interface with analog sensors.	Low cost, low energy footprint, wireless and internet accessible.	Single analog input pin, lack computation power for ML.	Karthick *et al.* (2021)
ESP32	Direct load control, BMS via PWM, remote monitoring and control of loads, interface with analog sensors, occupancy detection.	Low cost, low energy footprint, wireless and internet accessible.	lack computation power for ML.	Hijawi *et al.* (2020)
Raspberry pi	Direct load control, BMS via PWM, remote monitoring and control of loads, interface with analog sensors, and onboard Machine learning.	Have computation power for ML	Lack onboard analog input pins, need external ADC converter, higher energy consumption than Arduino boards, nodemcu, and esp32 boards.	Arumuga *et al.* (2017), Benyezza *et al.* (2021) Ferdoush and Li (2014)

Technology / device	EMS Role	Merit	Limitation(s)	Selected applications EMS related references
	Wireless connectivity of electric loads and sensors			
Zigbee® / Xbee®	Wireless connectivity of electric loads and sensors.	Low power consumption and medium coverage (not more than 300 m)	Lacks PWM support which is important in controlling some loads.	Karami *et al.* (2018), Kyi and Taparugssanagorn (2020), Sisavath and Yu (2021)
LoRA	Wireless connectivity of electric loads and sensors.	Low power consumption and wide coverage (about 5 km to 15 km).	Not suitable for real-time data applications since packets can be sent every few minutes (approximately 5 minutes). Supports only low data rates of up to 27 Kbps.	Kanakaraja *et al.* (2021)
GSM modules	Wireless RF connectivity of electric loads and sensors	Ability to send and receive SMS, ability to make calls	Rely on mobile networks which may have poor signal or network problems. Data charges for sending SMS and calls	Sibiya *et al.* (2021)
Building occupancy sensors (PIR modules, reed switch etc)	Appliance and lighting control	Ability to detect the presence of an occupant in the building	As a tripwire, it may remain activated even when the user forgot the window or door open. It may not detect when a person has left the building or room.	Demir *et al.* (2017), Pocero *et al.* (2017)

(i) Energy dashboard options

The IoT platform is a cloud computing software that provides an interface between IoT sensors and applications. These platforms offer a combination of the following services: Application development, visualization, data management, device control, and monitoring (Ray, 2016). The IoT platforms can simplify the deployment of low-cost human-machine interface (HMI) or EMS that could be used in emerging countries' scenarios because of the growing smartphone penetration and increasing internet access (Mbanaso *et al.,* 2015). The IoT sensors typically have low power consumption and low data bandwidth, hence are suitable for low-cost microgrid EMS applications in emerging countries. Developing countries have the lowest electric energy consumption per capita (International Energy Agency, 2019, 2014). Rolling out more low-cost EMS will go a long way in increasing energy efficiency and raising awareness. Studies show energy efficiency increase when users get feedback on their energy consumption (Chen *et al.,* 2014).

Chatbots

A chatbot is a software program that simulates human-like conversations with users via text messages. With the advancement in AI, technology chatbots are gaining more users and are being used to add value to various platforms and fields such as healthcare, finance, and energy sector (Adamopoulou & Moussiades, 2020; Kern *et al.,* 2022; Mogaji *et al.,* 2021). Since smartphones have high penetration globally including in developing countries. Therefore, chatbots could be leveraged to engage with electricity consumers and aid in energy monitoring and control applications. Chatbots could be created on top of already established instant messaging applications such as Facebook Messenger, WhatsApp, and Telegram (Suresan *et al.,* 2021).

Google Firebase

Has also attracted the attention of IoT developers. It offers cloud storage, server-side functions, and ML services. In a work by Hashmi *et al.* (2021), an EMS employing Firebase and Node-RED for DSM was proposed. Their work lacked an ML forecaster module. Node-RED was used to provide a Graphical User Interface (GUI) to the end user and as a Message Queuing Telemetry Transport (MQTT) broker. The MQTT is a lightweight IoT standard messaging protocol ideal for devices with limited network bandwidth. The MQTT protocol is made up of an MQTT message broker which acts as a server and a number of MQTT clients. In their work, EMS data from voltage and current sensors were sent in Javascript Object Notation (JSON) format. The MQQT broker relayed received data to an energy dashboard web portal

via HTTPS protocol. Although MQTT is the preferred protocol for data transfer due to its smaller data packet size, a developer may opt for HTTPS because it allows the transmission of lengthy headers and messages for human readability (Wukkadada *et al.*, 2018).

Google Sheets

Can also be employed as part of the EMS dashboard for data logging and visualization. Google Application Script is a cloud-based javascript platform that can integrate with other Google ecosystem Apps, and can therefore be a useful tool in developing a low-cost EMS (Moise *et al.*, 2020). Niculescu *et al.* (2021), integrated NodeMCU with BLYNK APP and Google Spreadsheet for indoor air quality monitoring. Their application only performed monitoring without any control or ML-based operations. Other tools that may be used in developing a cost-effective EMS that could be suitable for developing countries include the following: Heroku for ML; Git and Github for code version control and maintenance; IFTTT; and many more.

2.5 Main Takeaways and Potential Gaps

Load forecast is still a scientific research hotspot because of its economic implication – since it's vital in minimizing energy wastage for grid operators. The trend in load forecast studies appears to gravitate towards ANN with aggregation of ML models. A large body of hybrid models that have been developed mostly comprises two models blended. What will happen if more than two models are combined/stacked? Does performance degrade as more models are stacked? There is a need of compiling a set of microgrid dataset benchmarks upon which new works may be weighed against. Most forecast models developed were tailor-made for the specific grid. New insights may be found from testing already established models against different microgrid datasets including small, medium, and large datasets, datasets from weak grids, and mature grids datasets. Power outage works examined mostly conduct a study on the causes and prevention of blackout, however, works on blackout forecast are relatively few. This is still uncharted territory, more so, with respect to utility operator's-side forecasts as well as demand-side blackout forecasts.

Most smart meter studies examined failed to incorporate AI-based energy forecast modules. Furthermore, there is still room for research works on the role of social media platforms on users' energy behavioural change using IoT by conducting case studies in communities and households. Research on energy management through hybrid IoT wireless mesh networks is still open for more advancements. It is necessary to incorporate IoT technologies for LF and BF so as to effectively relay forecast information to the end user and enable proactive DSM and BMS actions. The main limitation in implementing IoT in low-cost EMS, is that, access to

internet is a requirement. The quality or signal strength of internet connection may aid or impair performance of the low-cost EMS smart meter. If access to internet is an issue, then use of GSM modules in the low-cost EMS smart meter may be a viable alternative.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Case Study Area

This study was conducted based on data from two locations: Ngarenanyuki village and Levolosi ward both in Arusha Tanzania. There are a number of microgrids already deployed in different parts of Tanzania and elsewhere, this study was conducted at Ngarenanyuki secondary school as part of the Energy4Growing (E4G) project which had a vision of seeing Ngarenanyuki school as a microgrid research laboratory. Later when the installed E4G switchboards went out of service in 2018 due to technical faults, a smaller implementation of a microgrid was deployed in an office building in Levolosi ward to continue with the study. It was more convenient to carry out data collection from the Levolosi microgrid as it was easily accessible, in the vicinity of Arusha Technical College, and closer to NMAIST; Ngarenanyuki village is remote. The E4G project was carried out by a research team of the Department of Energy of Politecnico di Milano which aimed to develop and implement a hybrid micro-grid in Ngarenanyuki school to improve the local power supply service of the school. The author was part of the E4G team after receiving a PhD scholarship as part of the fruits of the E4G project. The author was involved in the testing phase of the innovative converter and control switchboards at MCM PoliMi spin-off laboratory (in Milan, Italy) and was later also involved in the installation and monitoring of the switchboards that ensued (at Ngarenanyuki school in Arusha, Tanzania).

Figure 4: Energy sources at Ngarenanyuki school before deployment of E4G project: (a). a 3.2 kW run-off-river micro-hydropower (MHP) plant, Banki type, (b) Inverter-based PV DC backup systems of 940 W coupled with 1400 Ah/12V battery bank, (c) 5 kW diesel generator, (d) non-operational wind turbine, (e) Inverter

Before the deployment of the E4G converter and control switchboards, the school had a mix of distributed generation sources (Fig. 4). The main power source of the school was a 3.2 kW run-off-river micro-hydropower (MHP) plant, Banki type. Excess hydropower was wasted as heat in the energy dampers of the hydro turbine generator instead of being stored in the battery bank. The water stream for the turbine was shared and managed by the local farmers causing intermittency in the hydropower output. As farmers would divert water away from the MHP to their farms; this was years later resolved with a configuration that allowed farmers to divert water after it had gone through the MHP plant thus not interfering with MHP plant output. Variations in the MHP output then remained due to rainy seasons.

Before the deployment of the E4G converter and control switchboards, the school was also equipped with different additional backup systems: (a) Inverter-based PV DC backup systems of 940 W coupled with a 1400 Ah/12V battery bank, (b) A manually operated 5 kW diesel generator was used only when hydropower or PV-inverter-battery power was insufficient to run the school's printer, photocopy machine, or during special celebrations/events, etc. The

generator was for emergency use only to cut down costs from buying fuel for the generator (Mandelli *et al.*, 2015).

Before designing the E4G switchboards, the E4G research team conducted an assessment of the energy consumption of the school by installing an energy meter to measure and record the load power flows of the school from June to September 2014. The resulting consumption pattern is shown by the Box plot in Fig. 5. Median consumption is represented by the line halfway on each box, whereas, the whiskers represent outliers (rare load extremes). The load curve varied greatly from one day to the next (Carmeli *et al.*, 2014; Carmeli *et al.*, 2015).

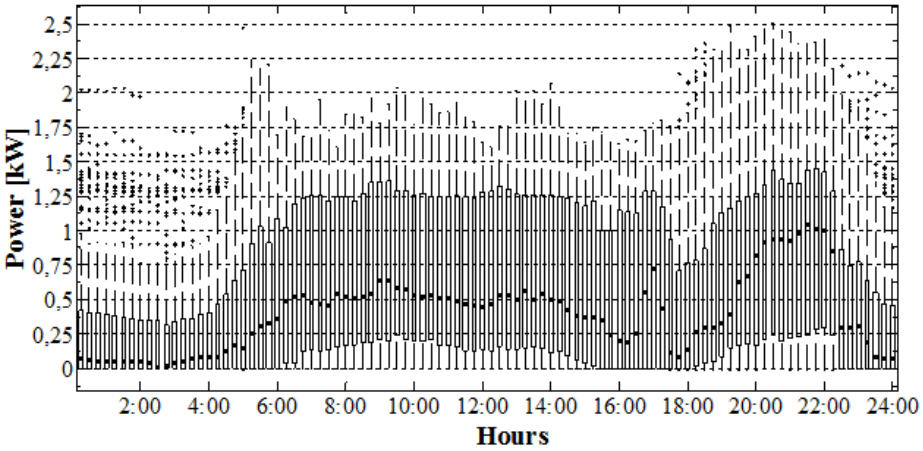


Figure 5: Box plot of daily load before upgrading Ngarenanyuki school microgrid

Under the E4G project, the Politecnico di Milano Department of Energy collaborated with EKOENERGY (Home - EKOenergy, n.d.) and SunEdison to implement at Ngarenanyuki Secondary School an innovative converter and control switchboards. These were designed to manage the school's 10 kW hybrid micro-grid comprising: a run-of-river hydropower system (3 kW), backup generator (5 kW), PV-inverter, and battery storage (Carmeli *et al.*, 2015). Apart from installing the switchboards, the project involved upgrading the school's PV by 3 kW, and battery bank by 30x202 Ah/12V maintenance free sealed lead-acid batteries. At this point the old 1400 Ah/12V flooded lead-acid battery bank was already near the end of its lifespan therefore it was phased-out and replaced with the new maintenance free sealed lead-acid batteries.

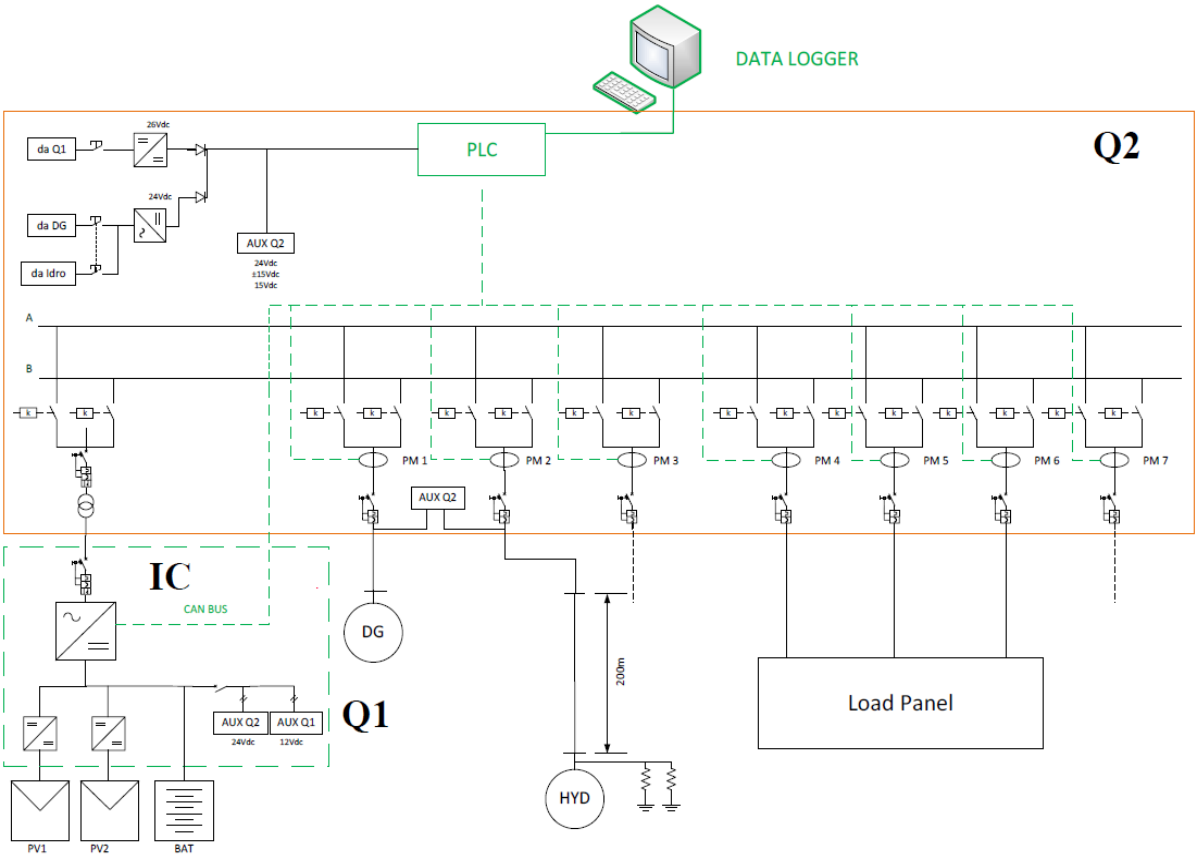


Figure 6: The study area, Ngarenanyuki microgrid architecture

The deployed architecture (Fig. 6) of the microgrid supports four power sources. The PV-Inverter, Diesel Generator, and hydro turbine are now connected to the system, while spare input power is reserved for future use. The architecture integrates mixed power sources available in the school allowing to compensate for the limits of each one (Carmeli *et al.*, 2014; Mandelli *et al.*, 2015). Moreover, in contrast to the previous single bus bar architecture, the new architecture relies on a double bus bar, inspired by the classical configuration adopted for big power plants, to allow greater flexibility in the energy management of the school. This configuration also allows connecting the E4G micro-grid in parallel with other systems (i.e., in the future, the national grid) and was expected to support maximum peak power loads up to 20 kW in the future.

The system could operate in Manual/Automatic modes. Manual mode allows the operator to connect or disconnect loads as well as energy sources. Automatic mode automatically connects/disconnects sources and loads based on their priority index using the Programmable Logic Controller (PLC). When operated in automatic mode, the system uses excess energy from the hydro turbine to charge the batteries and reduce wastage of energy as heat on the dump loads connected to the hydro turbine to aid in energy balance, thus, the batteries are charged by both the PV panels and the excess energy from the hydro turbine. The PLC also serves as a data logger set to a 1-second sampling rate. In Fig. 6, Q1 represents the inverter

control (IC) board while Q2 denotes the PLC switchboard (Carmeli *et al.,* 2014; Mandelli *et al.,* 2015). A detailed description of the microgrid, from the design phase to the deployment and operation is available on the Energy4Growing research team's Facebook page (https://web.facebook.com/energy4growing2014/). All the data and important milestones of the project implementation can be freely accessed on the Facebook page. The Facebook page also facilitated easy sharing of the project approach to people from scientific and non-scientific backgrounds.

The connection of Ngarenanyuki to the main grid availed the opportunity for more energy-intensive activities using the main grid whereas low-power activities such as lighting and running computers were reserved for renewable sources. For example, after being connected to the main grid, the school started a brick-making project. Unfortunately, in the first quarter of 2018, the implemented E4G smart switchboards suffered a fatal fault that caused the switchboards to stop working. Due to logistical reasons, it was not repaired. The school grid resorted to manual management of the power sources. In order to continue with this study, it became necessary to implement a smaller, easily accessible microgrid in Arusha township; a suitable office building at Levolosi ward with a pre-existing solar system installed was identified and selected for the remaining objectives of this study. The dataset used for Load forecast did not have a blackout indicator variable because at the time Ngarenanyuki microgrid was entirely off-grid, and not connected to the main grid (TANESCO). Therefore, it wasn't possible to measure blackout incidences. When it came time to investigate blackout forecasts, a microgrid site in Arusha township (Levolosi ward) was selected as a case study area due to logistical reasons.

3.1.1 Levolosi Blackout Pilot System Overview

Another pilot test site for this work was an office building located in the Levolosi ward of Arusha municipality in Tanzania. The energy management system as shown in Fig. 7, comprises 200 W PV, 100 Ah lead-acid battery, and a 2.5 kW inverter. Such pico-size systems are typical in SSA where PV systems are undersized due to financial constraints (Abubakar Mas'Ud *et al.,* 2016; Muchunku *et al.,* 2018). The experimental test site already had the PV-battery-inverter system in place before this study was conducted. Therefore, the PV-battery-inverter sizing and setup were not part of our study, rather the goal was to investigate blackout forecast from the customer's vantage point. The PV-inverter and battery bank at the site are essentially used as backup power for merely small loads and applications such as lighting, laptop, phone charging, and a surveillance camera; the backup system additionally was used to

power the datalogger used in this study during blackouts. Other relatively bigger loads at the site like photocopy machine, electric fan, microwave, and electric kettle are only used when utility power is available.

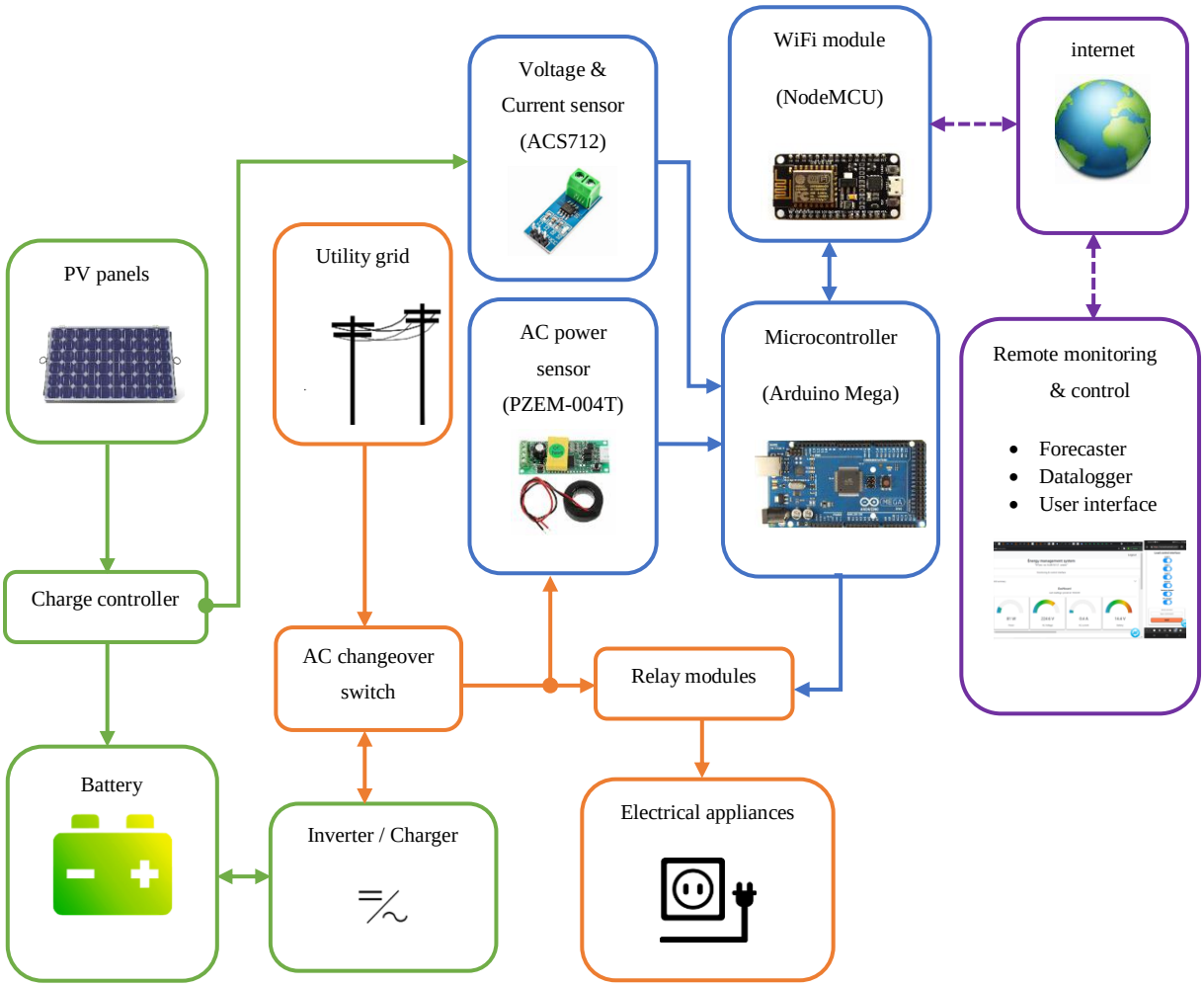


Figure 7: Proposed Levolosi test site energy management system architecture scheme

The battery bank at the site can be charged either by PV or by utility power through an inverter charger. The main electricity supply to the pilot site comes from the national electricity supply company - TANESCO. This consists of single phase supply line of 230 ± 5% V at 50 Hz ± 2.5% as per the Tanzania national grid standards. Power from both the utility and the inverter is managed by the local smart controller (smart meter). The low-cost implemented smart meter comprises a PZEM-004T module used to measure AC voltage, current, and frequency. The PV voltage and battery voltage were measured with a voltage divider circuit interfaced to Arduino Mega microcontroller ADC (Analog-to-digital converter) pins whereas, PV current was measured by ACS712 hall effect current sensor module also interfaced to the ADC port of Arduino Mega board. The PV-battery DC voltage and current sensor measurements facilitated PWM charging of the battery using P-channel MOSFET high side switching controlled by Arduino Mega board. All the measured sensor data were uploaded to the cloud via WiFi module

ESP8266 (NodeMCU). Received data from the site were stored in a MySQL database on a Linux-based server also used to perform blackout forecasts. A web page on the server was used as an energy dashboard for monitoring and remote-control operations. Figure 8, shows the main components of the implemented low-cost EMS smart meter. More details of the smart meter are provided later in Section 3.7.

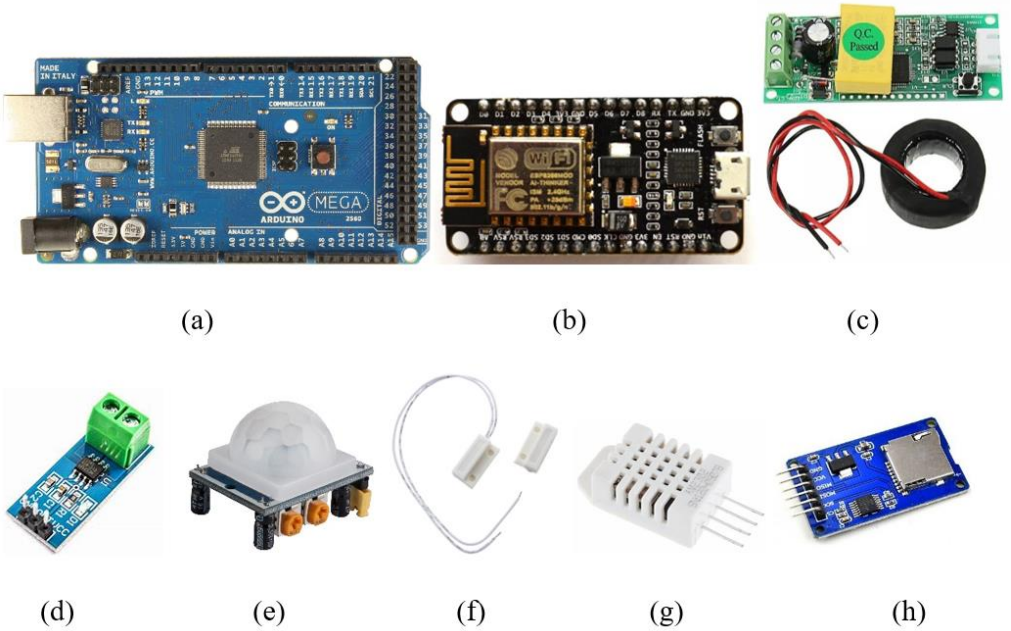


Figure 8: Low-cost electronics components used in the implemented EMS smart-meter, (a) Arduino Mega; (b) NodeMCU; (c) PZEM-004T energy meter; (d) ACS712 current sensor; (e) PIR motion sensor; (f) Reed switch; (g) DHT22 temperature and Humidity sensor; (h) SD card module

3.2 Data Collection

3.2.1 Ngarenanyuki Microgrid Data Collection

With reference to Fig. 6 and Fig. 9, the PLC had the capacity to store the school's grid electric data equivalent to 2 months, still, due to a PLC software bug, it saved 5 files (equivalent to 5 days of data), and thereafter the PLC overwrote the files. Initially one of the school staff volunteered to manually download the data, but inconsistency led to missing some data. An internet satellite dish equipment was installed in the school, which among many benefits, it facilitated remote access to the installed switchboards and PLC. The daily PLC data download process was then fully automated thanks to a combination of a computer batch script, Cloud computing services (Google Drive), and Microsoft Windows task scheduler. In this configuration, a computer with internet access was tethered to the Switchboards PLC and linked via FTP (File Transfer Protocol) in order to daily automatically fetch data logged CSV files into a designated Google drive folder.

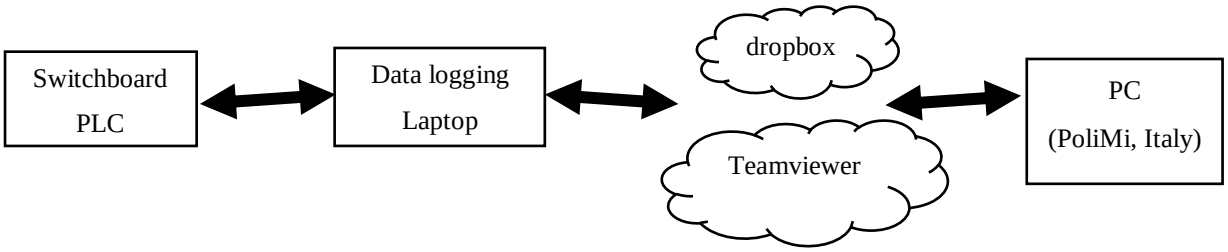


Figure 9: Ngarenanyuki microgrid data collection set-up

3.2.2 Levolosi Microgrid Data Collection

With reference to Fig. 7, data from different sensors in the grid was mainly processed by the Arduino Mega microcontroller and transferred to the NodeMCU microcontroller which would finally upload all the data to a web server MySQL database for storage. Internet connectivity hotspot was provided by a smartphone which was also meant to act as a user interface for wireless remote control operations, Machine learning operations on collected data, and an energy dashboard. An SD card module was also included as an option for local data storage. A script was set up to run on the webserver to sense when data was not uploading or other anomalies such as power outage and low battery SoC.

3.3 Datasets used in Model Development

3.3.1 Ngarenanyuki Secondary School Microgrid

Ngarenanyuki microgrid dataset used in this work was from 15 May 2015 to 7 March 2018. The data sampling was one second, but during pre-processing, it was converted to hourly aggregated observations. The microgrid dataset contains a total of 12 912 samples. The training and cross-validation dataset used was from May 2015 to January 2018, while the test data used was from 1 February 2018 to March 2018. Data used are available for research purposes thanks to the Energy4Growing project (E4G, 2018).

The dataset shown in Table 2 comprises a total of 23 features from the previous day as predictors and the next day (24 hours) load profile as the target or response. Prediction of the entire next day is performed at midnight (00 hrs) which is the start of the next day. For each hour *h* of the next day, the forecast is based on the values of the 23 predictors at the hour *h* of the day before. Weather data from a nearby airport was used as part of the 23 features (Raspisaniye Pogodi Ltd, n.d.). The weather parameters incorporated in the dataset were outdoor relative humidity, air temperature, wind speed, atmospheric pressure, and dew point temperature.

Table 3: Ngarenanyuki microgrid's dataset features description

#	Feature / predictor	Feature Description	Evaluation time
1	Month	Month number of the year	day D
2	Day	Day of the month	day D
3	WeekDay	Day of the week	day D
4	Hour	Hour of the day	day D hour h
5	Weekend	Weekend and holiday indicator	day D
6	temp	Ambient temperature (control room temperature)	day D hour h
7	P_DG	Back-up diesel generator power	day D hour h
8	P_HYD	Micro hydro power	day D hour h
9	P_inv	Power from inverter	day D hour h
10	Vdc_bus	PV-inverter DC bus system voltage	day D hour h
11	PPV	PV array output power	day D hour h
12	SOC	Battery bank state of charge	day D hour h
13	AirTemp	Outdoor air temperature from nearby airport	day D hour h
14	atmPressure	Atmospheric pressure	day D hour h
15	RHumidity	Relative humidity	day D hour h
16	WindSpeed	Wind speed	day D hour h
17	DewpointTemp	Dew point temperature	day D hour h
18	T2_temp	Ambient temperature 2 days before	day D-2 hour h
19	T2_Load	Load 2 days before	day D-2 hour h
20	T1_temp	Previous day control room ambient temperature	day D-1 hour h
21	Year	Data log Year	day D
22	T1_Load	Previous day load	day D-1 hour h
23	Load	Current day load	day D hour h

3.3.2 Levolosi Experimental Blackout System

The blackout dataset used in this work covers the period of January 2021 to December 2021, amounting to a total of 8016 hourly aggregated samples or 30 144 15-minute interval samples. The subset of the dataset used comprised 10 relevant input variables from the previous 14 days as predictors. These 10 predictors were selected after performing a correlation test with the blackout indicator variable. The 10 input features used in this work are shown in Table 3.

Weather data for the area could not be obtained, thus, was not used. Figure 10 shows the correlation of the input variables to the blackout variable. The 'ac_voltage' and 'frequency' variables exhibited negative correlation values because they are inversely related to blackout, in that, the presence of blackout (electric power = 0) implies zero AC line volts and zero frequency on the test site power line.

Table 4: Levolosi microgrid dataset input features details

#	Input variable	Input variable detail	Evaluation time
1	Day	Day of the month	day D-1
2	WeekDay	Day of the week	day D-1
3	Hour	Hour of the day	day D-1
4	AC_voltage	Average AC supply line voltage from electric supply company	day D-1
5	Frequency	Average AC supply line frequency	day D-1
6	B14	Average Blackout profile 14 days before	day D-14
7	B7	Average Blackout profile 7 days before	day D-7
8	B3	Average Blackout profile 3 days before	day D-3
9	B2	Average Blackout profile 2 days before	day D-2
10	B1	Average Previous day blackout profile	day D-1

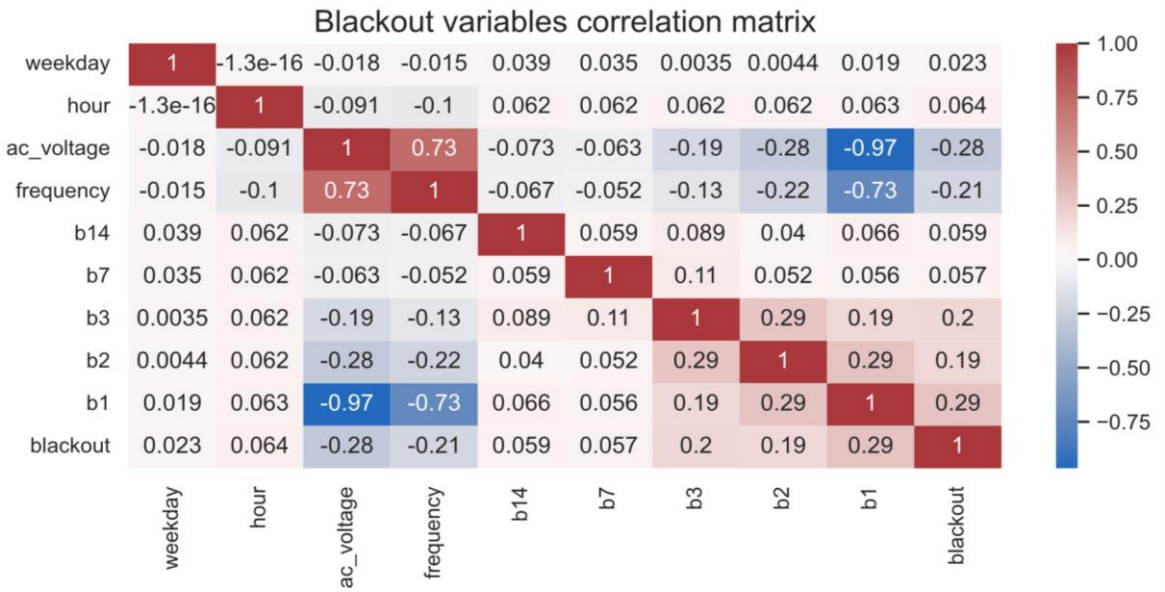


Figure 10: Input variables correlation to blackout variable. A value close to '1' signifies a high correlation, whereas a value close to zero shows a low correlation to the blackout variable. Negative values show a negative correlation

Three different strategies were performed for the short-term blackout profile prediction target or response, namely: (a) blackout prediction for the next 15 minutes, which is a single scalar value; (b) blackout prediction for the next hour, which is also a single scalar value; (c) lastly, blackout prediction for the entire next day (24 hours), which is a vector of 24 blackout index values corresponding to hours of the following day. Blackout prediction of the next 15 minutes is performed 15 minutes prior, whereas prediction of the next hour is performed an hour in advance. Blackout prediction of the entire next day is performed at midnight (00 hrs) which is the start of the next day. For example, for each hour *h* of the next day, the forecast is based on the values of the 10 predictors at the hour *h* of the days before. Similarly, 15 minutes-ahead prediction and 1-hour-ahead prediction are performed in the same manner based on the corresponding time step in the past records. During the calibration phase, measurements for important parameters such as voltage, current, and frequency were taken at intervals with a commercial multimeter and corroborated with measurements of the test site smart meter datalogger.

3.4 Design and Research Approach

In order to achieve the objectives of this study, the framework in Fig. 11 was adopted. The fishbone diagram shows the four main components contributing to the EMS considered in this study namely: load forecast, blackout forecast, user engagement, and battery management. Figure 12 flowchart highlights the generalized load and blackout forecast procedure employed in this study, where the objective function is forecast optimization. The typical approach involves data source, data pre-processing, model training and testing, error evaluation, and model deployment. Section 3.3 describes the data pre-processing approach followed in this study; which is an important step before performing load forecast or blackout forecast.

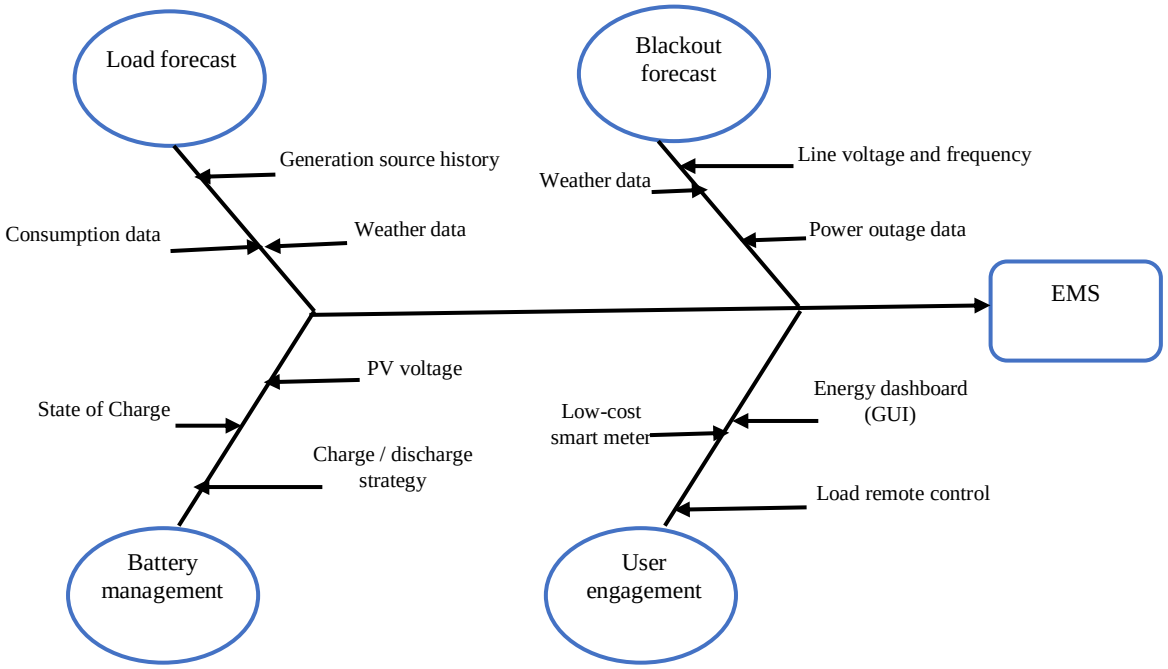


Figure 11: The conceptual framework of the main components addressed in this study

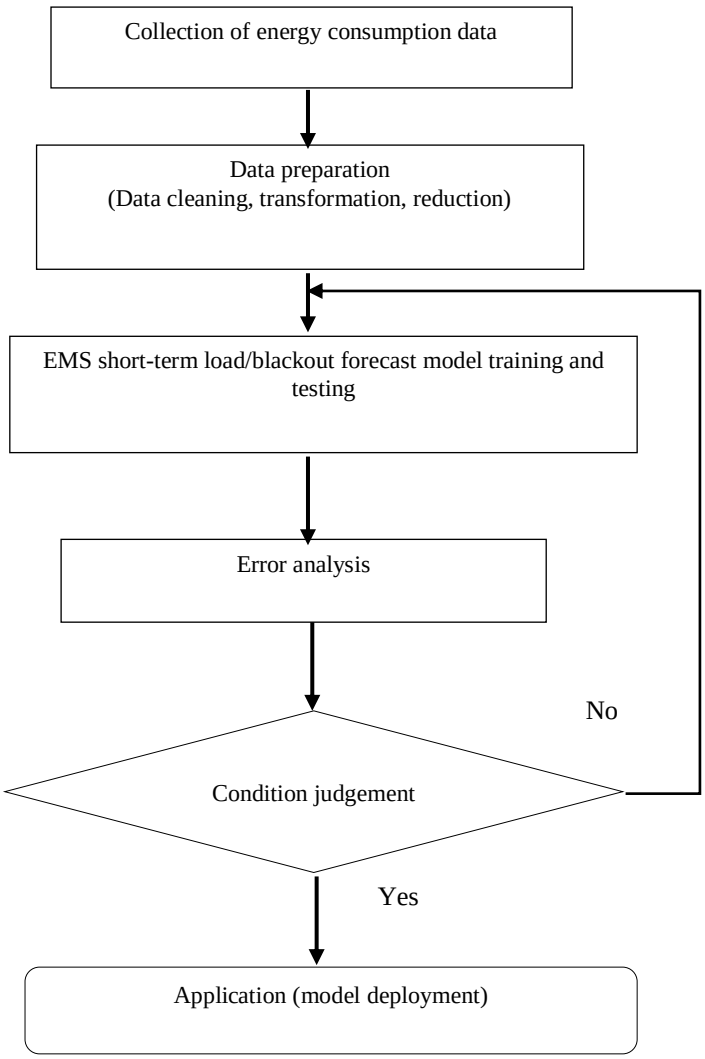


Figure 12: Load/blackout forecasting flow chart

3.5 Data Preprocessing

3.5.1 Ngarenanyuki Dataset Preprocessing

The first step in this work was to explore the dataset in order to check for integrity, spot missing values, and examine the relationship between load and the other features in the dataset. As an example, Fig. 13 shows the unstable nature of load consumption of the microgrid for October 2017. The data cleaning stage involved the detection and correction of missing values and outlier anomalies. Days in the dataset found with missing values were filled with mean values of adjacent neighbouring records. Days with no logged entries were ignored. For smoothing, the dataset was transformed by retiming from 1-second observation entries into mean hourly observations and further smoothed to remove outliers. Sgolay (Savitzky-Golay filter) algorithm-based data smoothing method was opted out of seven other data smoothing methods which are *movmean*, *movmedian*, *lowess*, *rlowess*, *loess*, *rloess*, and *Gaussian* smoothing method available in MATLAB software. *Sgolay* was used because it is effective in preserving higher moment peaks in a signal (de Oliveira *et al.*, 2018; Kaneko *et al.*, 2016). Data smoothing has the effect of removing noise from data hence improving the performance of the prediction model.

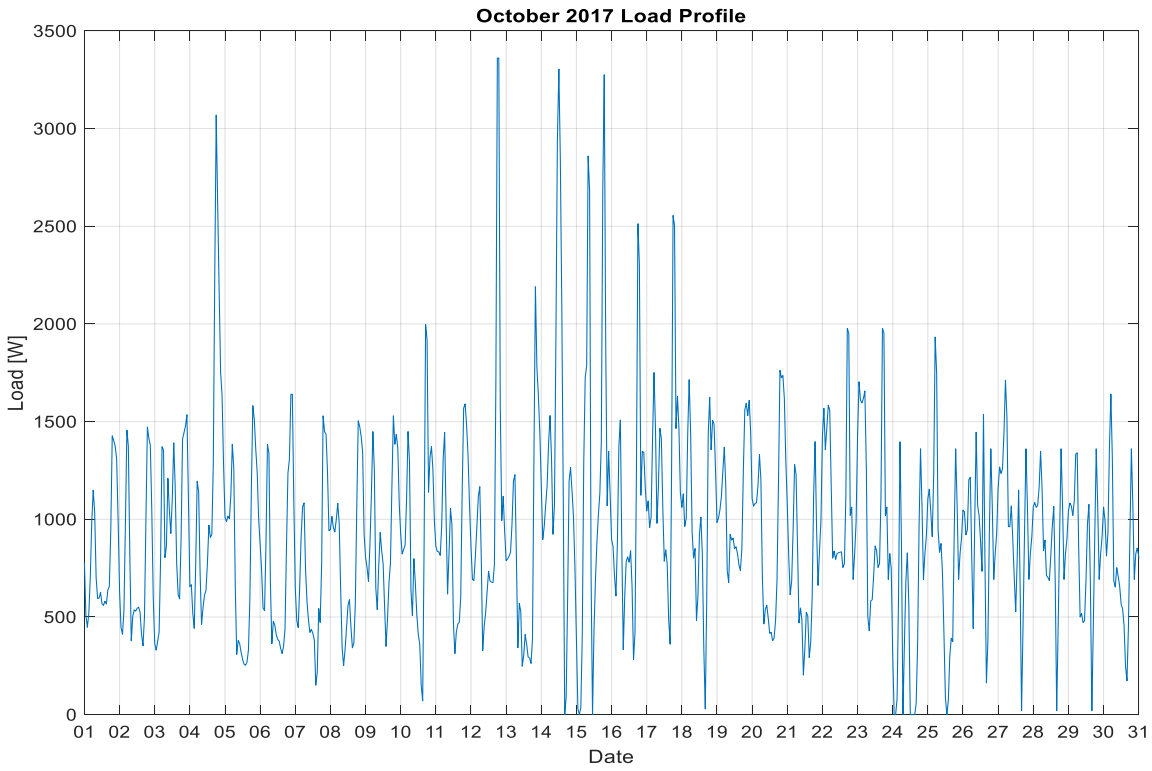


Figure 13: Ngarenanyuki microgrid unstable load profile nature

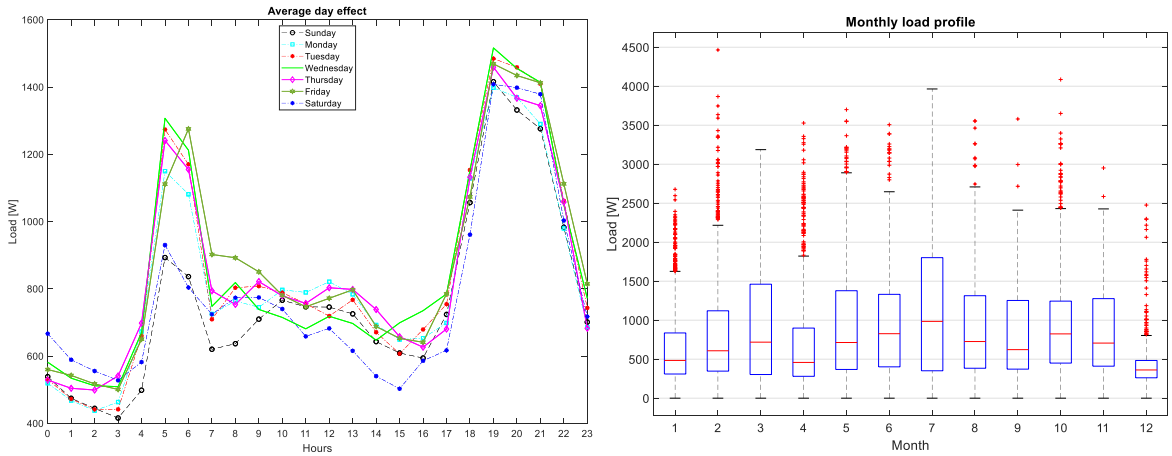


Figure 14: (a) Week Day average load, (b) Box plots of load values for each month

Figure 14 (a) is the average hourly load profile line plot for every single day of the week. Power consumption peak hours are observed to be around 05:00 and 18:00 - 21:00 hours. Figure 14 (b) shows the hourly load consumption distribution for each month. The central line mark on each box plot is the median value, and the dotted line whiskers are the extreme data points while outliers are plotted using the '+' symbols. The consumption pattern of the box plot in Fig. 14 (b) is linked to the academic calendar of the school. It had a population of about 500 students and staff in the period covered by the dataset. Resident students are of four classes. Long School holiday breaks were in June whereby 2 classes of students break for four weeks while the other classes break for one week. All students break for two weeks in April and September. They all break for four weeks in December.

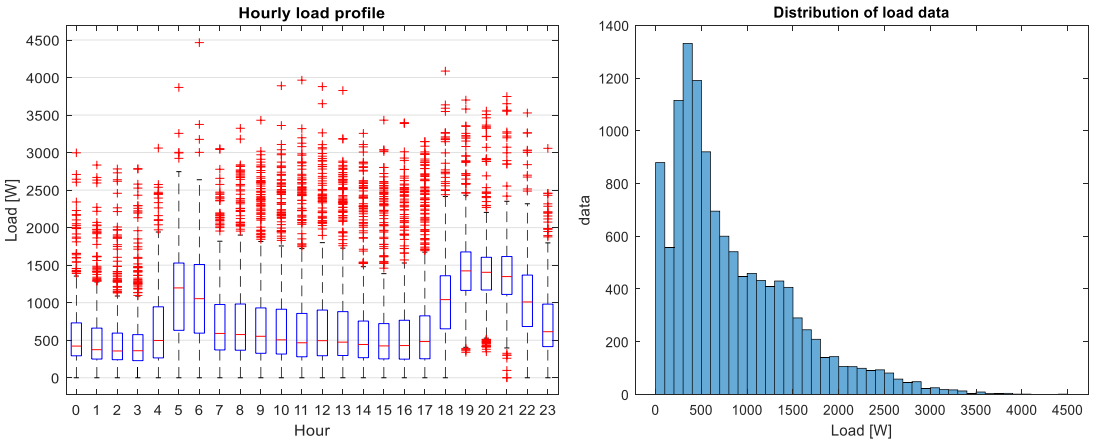


Figure 15: (a) Hourly Load Profile of Ngarenanyuki microgrid, (b) Histogram plot showing the distribution of load consumption

Figure 15 (a) gives insight into the hourly energy usage trend of the microgrid, it can be seen there is an increase in power consumption during morning hours around 5 and 6 am; another increase in consumption is later in the evening from 18 pm till 22 pm. This is explained by the daily schedule of Ngarenanyuki boarding students when they have to rise early in the morning when it is still dark outside, therefore, needing to turn on the lights, and again in the evening

when student study in their classrooms before they retire to sleep. During the day lights are off, with only a few office appliances being turned on at the teachers' offices. Figure 15 (b) gives an insight into the dataset's load distribution; Most of the load consumption is around 200 W to 1000 W, and power consumption hardly exceeds 3500 W. Evidently this consumption is small for a community of about 500 people; during the time covered by the dataset, electric power was mostly used for lighting and operating a few office appliances, later the school consumption grew. The first bin in the histogram also reveals incidents of blackout in the school grid. Figure 16a shows the typical daily load pattern of the microgrid whereby hourly consumption peaks in the morning and evening, but on average power doesn't exceed 2400 W. On the other hand, Fig. 16b, shows incidents where more power was consumed on some mornings and evenings, and the pattern changes from day to day randomly.

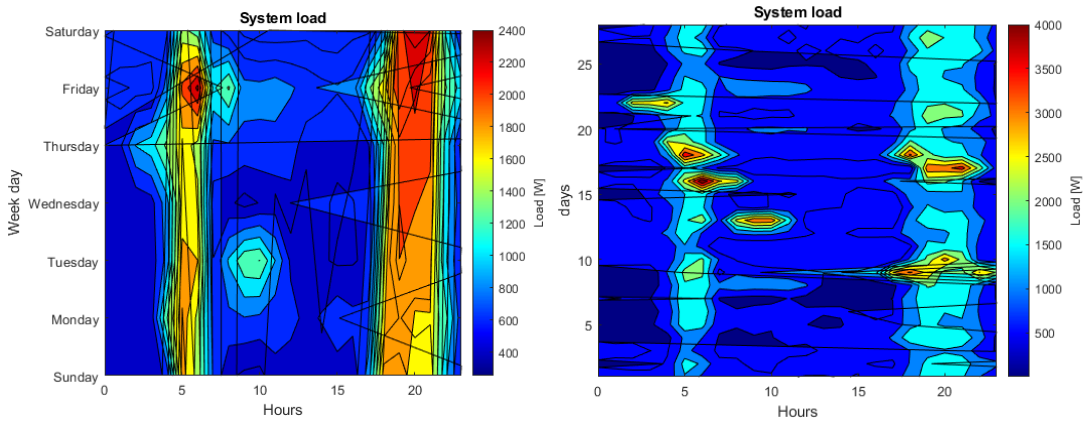


Figure 16: Load consumption heat map, (a) Week Day versus hour of the day (b) Day of the month versus hour of the day

3.5.2 Levolosi Dataset Preprocessing

Levolosi dataset was transformed from minute resolution observation entries into 15 minutes average observations as well as mean hourly observations. Dataset retimed to 15 minutes resolution was used for 15 minutes very short-term blackout prediction, whereas, the dataset retimed to hourly resolution was used for hour-ahead and day-ahead blackout short-term prediction. Evidently, there could be a risk of misrepresenting the duration of the blackout say if it occurred on the 55^{th} minute of the hour in question. In order to overcome this challenge, the blackout variable represents normalized blackout in an hour using values ranging from '0' to '1' (60 minutes of blackout). For example, in the case of a blackout that occurred in the 55^{th} minute of the hour and lasted for 5 minutes this is represented by the blackout variable as '0.08' whereas a blackout that occurred at the 45^{th} minute of the hour and lasted for 15 minutes would be represented by the blackout variable as '0.25', and so on. This logic was used to solve blackout prediction as a regression machine learning problem. On the other hand, the short-term blackout prediction was also formulated as a classification machine learning problem.

Obviously, the shorter the data sampling period the more meaningful and truthful it captures occurred blackout events. In this regard, 15 minutes interval data is better than hourly and so forth. Furthermore, input data was smoothed to remove outliers/noise to optimize the prediction skill of the model.

Power supplied at the pilot test site from the local utility company has been observed to be at times unstable, and suffering from irregular power outages may be experienced on the client side. In the period between January to December 2021, aggregated blackouts measured at the pilot site amounted to an equivalent of about 16 days. As shown in Fig. 17, the months of March, April, November, and December were the worst hit by power outages. Power outages are irregular and the pattern differs from month to month, for instance from Fig. 18 the month of May suffered fewer power interruptions than April. From the heatmap, '1' indicates complete blackout for an entire hour, whereas '0.5' signifies blackout for 30 minutes during the respective hour being considered. In some extreme cases blackout may last more than 24hrs, as was the case on April 18th and 19th.

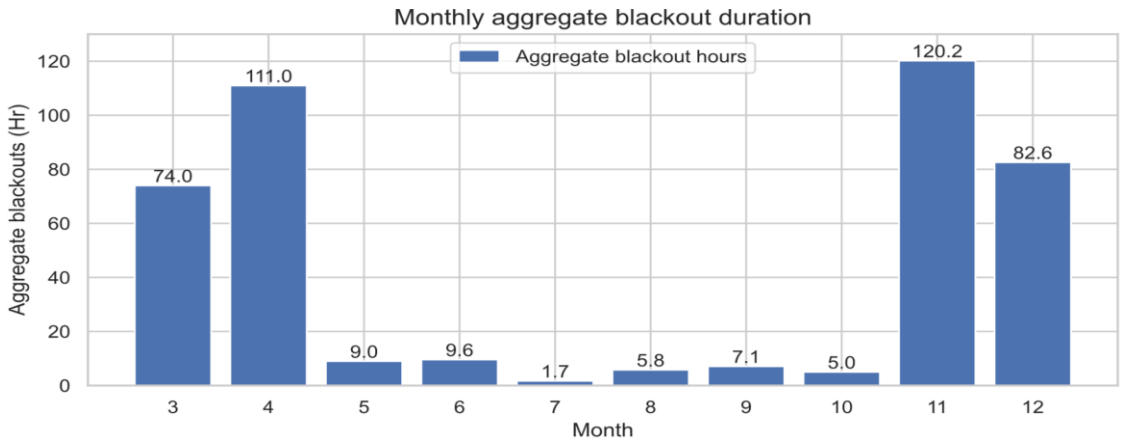


Figure 17: Monthly aggregate blackout history

Power outage notifications are typically sent out in advance through media if the scheduled power interruption affects or covers a large area such as an entire city or district. However, few or no power interruption notifications were received in advance at the pilot's site street office buildings, because only a small neighbourhood was affected. These kinds of localized blackouts are usually due to distribution line faults or maintenance work. Therefore, this work endeavours to predict power outages merely by using little information regarding electric power parameters as observed from the point of coupling at the customer side and without having prior information of scheduled power outages. The energy management system installed at the test site is tasked to predict blackout, without access to information about the grid status from TANESCO's substation control centre, or without knowledge of any fault or protection relay that may have tripped upstream.

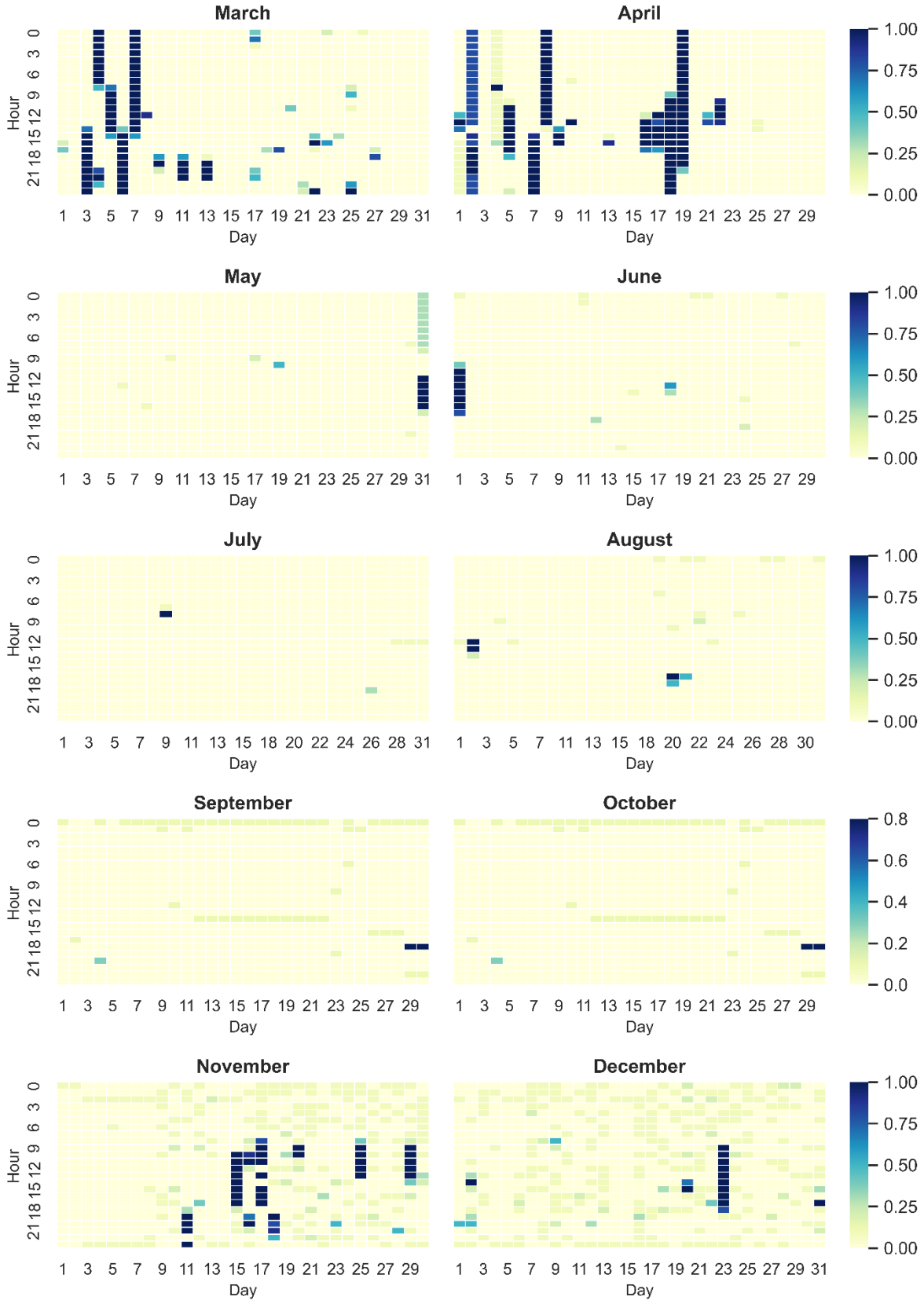


Figure 18: Power outage heatmap across different months

3.6 Load Forecast Model Development

Figure 19 shows the proposed model framework adopted in this work. The model was implemented in the MATLAB© 2018 version environment. The main toolbox that was used were: statistics and machine learning toolbox, and curve fitting toolbox. Specifications of computer used where: Intel(R) Core (TM) i7-4790 CPU @ 3.60 GHz 3.60 GHz 64-bit Processor, and 12GB DDR3 RAM. Data exploration, cleaning, and transformation stages were implemented so as to enhance input data and the model's behaviour.

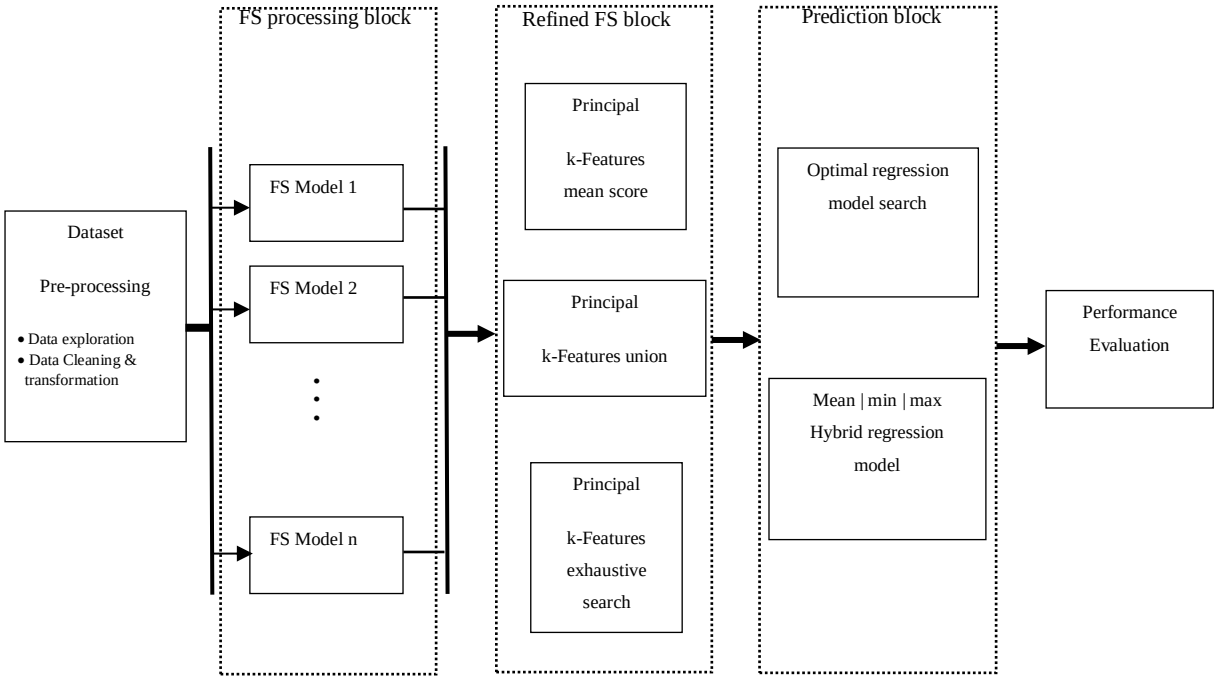


Figure 19: Proposed day-ahead load forecasting model framework

The feature selection stage aims at minimizing the number of predictors in order to reduce computation complexity without compromising prediction performance. As it will also be shown later, feature selection models differ in their mechanism, thus they yield different results. It is therefore, proposed to combine more than one FS model in order to increase performance at the prediction stage of the load forecast procedure. With respect to the proposed model in Fig. 19, input data is fed to each of the FS models. In this work, 5 FS models were used. Each of the 5 feature selection models ranked the same features with different weight scores. In general, it was observed that features that were voted to be the most important according to one FS model, were also given high importance in the other FS models. To overcome the dilemma of feature selection, three approaches are proposed in the selection of a subset of k principal features from a superset of n features through (a) mean score; (b) subset union; (c) refined exhaustive search based on k-combination, they are described later. For computational time reasons, either the mean score or subset union or both approaches can be

used and compared to select features for training and prediction. The refined exhaustive search method should be opted as the last resort since it is computationally much more intensive than the mean score and subset union approaches.

The identified k principal features can be trained by 20 different regression models, thereafter, performance efficacy is assessed by the mean absolute error (MAE) and root mean square error (RMSE) of each regression model compared. All the possible combinations of 2 regression models out of the 20 models can be fused to form a hybrid model based on the mean or min or max values of the best 2 models. The resulting hybrid regression model show improved prediction performance. Ultimately, the choice of which path to follow in the proposed framework depends on the computational resources available as well as the degree of prediction performance desired.

3.6.1 Feature Analysis

Feature selection is a dimensionality reduction technique that ranks and selects an influential subset of the possible predictors or features with the best predictive power of a prediction model. Feature selection is application-oriented (Jovic *et al.*, 2015). Studies show that the right combination of features is important as the individual features are included in the prediction model (Li *et al.*, 2017). The 5 feature selection methods used in this work are Random Forest; Relieff; Ensemble regression tree; Compact regression tree; Neighborhood component analysis (NCA).

Random Forest (RF) algorithm is a conventional approach to embedded feature selection. In this paper, a random forest of 200 bagged ensemble regression trees was grown and used to estimate unbiased feature importance. Relieff algorithm works by favouring features that give different values to neighbours of dissimilar response weights while punishing features that give dissimilar weights to neighbours of the same response values (The MathWorks, n.d.). Matlab ReliefF function used in this work was configured to 10 nearest neighbours and regression method for computing weights. The predictor Importance Matlab function was used with ensemble regression tree and compact regression tree to compute estimates of feature importance. The larger the estimated value the more important the feature. Neighborhood component analysis (NCA) is an embedded feature selection method. The fsrnca Matlab function was used as NCA in this work.

Each of the 5 feature selection models ranked the same features with different importance and weight scores. In order to increase prediction performance, 3 global ranking approaches derived from the 5 FS models are proposed in the selection of a subset of k principal features

from a superset of N features through: (a) mean score; (b) subset union; (c) refined exhaustive search based on k-combination. A description of the 3 approaches is given in subsequent subsections.

3.6.2 Principal K-Features Mean Score

One approach was to find an arbitrary k number of principal features by finding the overall final ranking of the individual features taken as an average score from all the FS methods used. If the total number of features is N, then the weight rank will be N, N-1, N-2, …1. Then, the mean score for each feature was evaluated using Equation 1:

$$\overline{W_f} = \frac{1}{n}\sum_{i=1}^{n} W_{fi} \tag{1}$$

Whereby, n is the total number of FS models used, W_{fi} is the feature weight score.

In this work, for each FS method, the most important feature was assigned a weighted score of N=23 while the least important feature was assigned a weighted score of 1. For example, the hour of the day had an estimated importance weight score values of 4.36, 0.02, 9.6, 29.4, 2.3, and 0.27 computed in random forest, relieff, ensemble regression tree, compact regression tree, fsrnca , and robust NCA features selection respectively. Corresponding values assigned are 21, 20, 15, 20, 7, 19. Resulting in a mean weight score of 17, thus making the hour of the day the fourth most important feature. The final weight score of the hour of the day becomes 20 out of 23. Thus, the final rank of a feature is a mean of the individual weighted votes a feature scores from each of the 6 FS algorithms. The overall top 5 most important features from the principal k-features mean score approach were found to be ‘Load’, ‘T1_Load’, ‘T2_Load’, ‘Hour’, and ‘Day’. ‘Load’ is the first in importance and ‘Day’ is the fifth in importance.

3.6.3 Principal K-Features Union

A second approach proposed in this work is to create a features subset comprised of the set union between top k-features from each FS model without redundancy. In this work, principal features were obtained from a union of the top 5 most important features from each FS model without redundancy in the features selected. The resulting subset had 9 out of the 23 superset features, as follows: 'Load', 'T1_Load', 'Hour', 'T2_Load', 'Day', 'P_DG', 'Month', 'Vdc_bus', 'P_inv'.

3.6.4 Refined K-Features Exhaustive Search

Features recommended by the principal k-features mean score and union can further be minimized to find the best k-features by performing a mathematical k-combination of features given in Equation 2. This brute-force approach was refined in this work by only selecting without repetition 5 combinations of features that included the 'Load' feature which prior to this step was voted to be the most important feature by principal k-features mean score and union approaches. Computing all the 5 features subset from the 23 features set, resulted in 33 649 combinations, then afterward choosing only combinations with 'Load' feature inclusive reduced the number of combinations down to 7315. Taking only combinations that include both 'Load' and 'T1_Load' (previous day load) gives 1330 combinations. This refined number of combinations can reasonably be run through the 20 conventional load forecasting models used in this work in order to find a good enough regression model.

$$nCk = \frac{n!}{k!(n-k)!} \tag{2}$$

3.6.5 Load Forecasting Model Selection

Principal k-features from the features selection stage of the proposed framework are used to train and validate 20 different regression models. The regression models used are classified into 5 categories namely: Regression tree; Neural network; Gaussian process regression (GPR); Support Vector Machine (SVM); and linear regression. An overview of the forecast models is given as follows.

Linear regression is a linear approach to prediction modelling. The Matlab implementation used in this work used *least-squares*, *robust*, and *stepwise* fitting methods. Matlab implementation used in this work for SVM analysis is the linear epsilon-insensitive SVM (ε-SVM) regression.

For GPR this work used the *fitrgp* Matlab function to train the dataset. The kernel function options used in this work were: 'exponential' for the exponential kernel, abbreviated eGPR; 'squaredexponential' for squared exponential kernel, abbreviated seGPR; 'matern52' for Matern kernel with parameter 5/2, abbreviated MaternGPR; and 'rationalquadratic' for rational quadratic kernel, abbreviated rqGPR. This work used a two-layer feed-forward neural network consisting of 10 hidden layers and linear output neurons for regression. The network was trained with the Levenberg-Marquardt backpropagation algorithm (trainlm).

Regression trees in this work were implemented with the *fitrtree* Matlab function. Variations of the regression trees used were 'FineTree' with 'MinLeafSize' of 4; 'MediumTree' with 'MinLeafSize' of 12; 'CoarseTree' with 'MinLeafSize' of 36. This work used *fitrensemble* Matlab function with the input method 'bag' for bootstrap aggregation (bagging) forming a deep 'BaggedTree' prediction model. A shallow 'BoostedTree' was formed by employing the method 'LSBoost' (Least-Squares Boosting) on the fitrensemble. 'BoostedTree' model fits to minimize mean-squared error.

3.7 Power Outage Forecast Model Development

This section explains the 3 blackout models investigated in this work namely: Random Forest (RF) algorithm, Adaptive Similar Days (ASD) model, and a hybrid RF-ASD model. Their performance and efficacy are given later in the result section. RF algorithm already exists in literature; however, ASD and the hybrid RF-ASD are novel and have been proposed in this study. In this work, blackout forecast is tackled as both a regression problem as well as a classification problem. The objective of blackout regression is to predict continuous value output which indicates the blackout event and duration. On the other hand, the objective of the blackout classifier is to predict a binary output that indicates the occurrence of a blackout event.

Adaptive Similar Days (ASD) blackout prediction approach: it has long been established that the past day's data in a time series, can be used to make a short-term forecast. As already observed in the preceding Fig. 13 and 14, some months suffer from more power outage episodes than others, and the outage trend may evolve dramatically from one month to the next. Due to the stochastic nature of blackouts, a longer moving window could corrupt the training algorithm and yield lower accuracy. Therefore, it was deemed necessary to develop a model that uses fewer training data for prediction rather than a model that requires a large dataset to gain prediction competency. It is on this premise that past historical blackout data going 2 weeks back were used in our method for short-term (15 minutes-ahead, hour-ahead, and day-ahead horizon) power outage prediction. Given below is the adaptive similar days (ASD) algorithms equation for short-term blackout prediction

$$Bf_t = \frac{e_t}{N} + \frac{1}{n}\sum_{t=1}^{n}(b1_t + b2_t + b3_t + b7_t + b14_t) \tag{3}$$

From Equation (3), assume a short-term (either 15 minutes-ahead, hour-ahead, or 24 hrs-ahead) blackout forecast to be represented by Bf_t . Whereby, n is the number of blackout data points. $b1_t$ is the blackout index value at the time t one day before, $b2_t$ is blackout index at time t two days before, $b3_t$ is blackout index at time t three days before, $b7_t$ is blackout index at time t

seven day before, $b14_t$ is blackout index at time t fourteen days earlier. e_t is the forecast error obtained as a difference between the prediction value and the observed value. The N is a positive number forming the fraction error term $\frac{e_t}{N}$. The vectors $b1$ to $b14$ form 2 weeks sliding window. They are chosen because they are closer to the short-term forecast target, and it is assumed that they will capture any blackout pattern nuances that may exist in the time series data if any cyclical trends exist. For example, in predicting blackout events of the current week's Monday, the previous week's Monday's blackout data ($b7_t$) as well as the Monday 2 weeks ago ($b14_t$) are assumed to have some influence on the blackout prediction dynamics.

Short-term blackout forecast Bf_t in this work is computed according to Equation (3), using the following algorithm steps:

(i) Step 1: then Bf_t will be determined by the mean of the past blackout vectors for $b1_t$, $b2_t$, $b3_t$, $b7_t$, and $b14_t$. For example, if we wish to predict the second hour of the day, t = 01hrs, then $b1_t$ will be the blackout index value in the previous 1 day at the corresponding second hour of the day (01 hrs), whereas $b2_t$ is the blackout index value 2 days before at the corresponding second hour of the day (01 hrs), and so on.

(ii) Step 2: The resulting short-term power outage prediction is compared to the actual observed blackout data for that respective day (prediction target day), and the difference (error), $\frac{e_t}{N}$, is saved in a lookup table.

(iii) Step 3: The resulting short-term blackout prediction vector is summed with the fraction of the previous prediction error vector, $\frac{e_t}{N}$, from the lookup table (memory). After performing sensitivity analysis, N=2 was found to increase prediction accuracy. The term $\frac{e_t}{N}$ is used to correct the weights of day-ahead predicted power outage profile - Bf_t.

(iv) Step 4: The 2 weeks sliding window is moved one interval step forward, to make the next short-term power outage prediction.

(v) Step 5: Repeat steps 1 to 4.

Additionally, some general assumptions that govern the mechanism of the ASD and RF algorithms used in this work are:

(i) Assumption 1: Recent records or observations closer to the short-term prediction target have a stronger influence on the intended target period being forecasted. In other words,

they have a higher probability of being similar to the forecast target period. Therefore, in the vein of the Pareto principle, past records used in the 2 weeks sliding window of the ASD algorithm have more importance in predicting the short-term target than past records beyond or before the sliding window. In case there were no blackouts in the 2 weeks window, the probability of an imminent blackout is assumed to be very low.

(ii) Assumption 2: Power outage duration for the short-term target period does not exceed the maximum outage event duration of the preceding outages within the sliding window.

(iii) Assumption 3: By extension to assumption 1, power interruptions, and disturbances are cyclical in nature, thus, the interval of the third outage event is assumed to be the same as the interval between the previous two blackout events. For example, if the first blackout event in a day is observed at time t, and the second blackout occurs at time 3t, then the algorithm will expect the third blackout event to occur at time 5t. Granted, this may not hold true in all cases due to the stochastic behaviour of blackouts. For this reason, assumption 4 is used in effect to reset the learnt interval pattern if an incorrect prediction is made.

(iv) Assumption 4: Under normal conditions, the power line is assumed to have electric power by default, and without power under abnormal (blackout) conditions. At the start of the prediction day, the ASD algorithm is given a temporary prediction token or permit, after which it can make a prediction. This involves asserting a flag variable. However, if it incorrectly predicts the occurrence of blackout in the short-term target period contrary to the observed data showing no outage or being under normal powered conditions. In this case, the token is temporarily revoked or withheld by clearing the flag variable, and the power line is now considered to be back to normal operating conditions. Yet again, when a new power outage is detected a prediction token is granted back to the ASD algorithm by re-asserting the flag variable, and it continues to participate in subsequent outage predictions.

Random forest (RF) algorithm is a nonparametric machine learning method that can handle non-linear regression as well as classification challenges. It is based on decision trees which individually act as weak learners but overall become strong learners. The RF algorithm is robust and has been found to work well with small datasets as well as datasets with some missing values. A random forest of 10 bagged ensemble regression trees, and 100 classifier trees were grown and used to forecast short-term power outages. In this kind of problem, short-

term forecasts depend on past recorded data, therefore the walk-forward validation method was used instead of cross-validation. After training and fitting the RF model, a single-step forecast is made followed by error measurements, and then updating the RF model with observed data for the predicted day (target day) by appending it to input data ready for the next one-step forecast loop. The model steps through the entire test data in this manner of predicting and updating, until the last test data is reached by the walk-forward validation.

Random Forest Adaptive Similar Days (RF-ASD) hybrid model: Output from ASD module that predicts day-ahead from a sliding window lookup-table spanning 14 days prior to the target day, is fused with RF prediction, giving a final optimised RF-ASD blackout prediction via element-wise vector mean of both models. In other words, the output of the ASD model is combined with that of the RF model to obtain the mean of the two models. Thus, the RF-ASD hybrid model is an average of both RF and ASD models. Figure 18 also describes the RF-ASD blackout prediction algorithm flowchart.

Availability of localized blackout data from the utility company such as TANESCO regarding distribution line faults, maintenance work, tree trimming schedule, and rainfall could potentially improve the BF model due to correlation between such inputs and power outage occurrence.

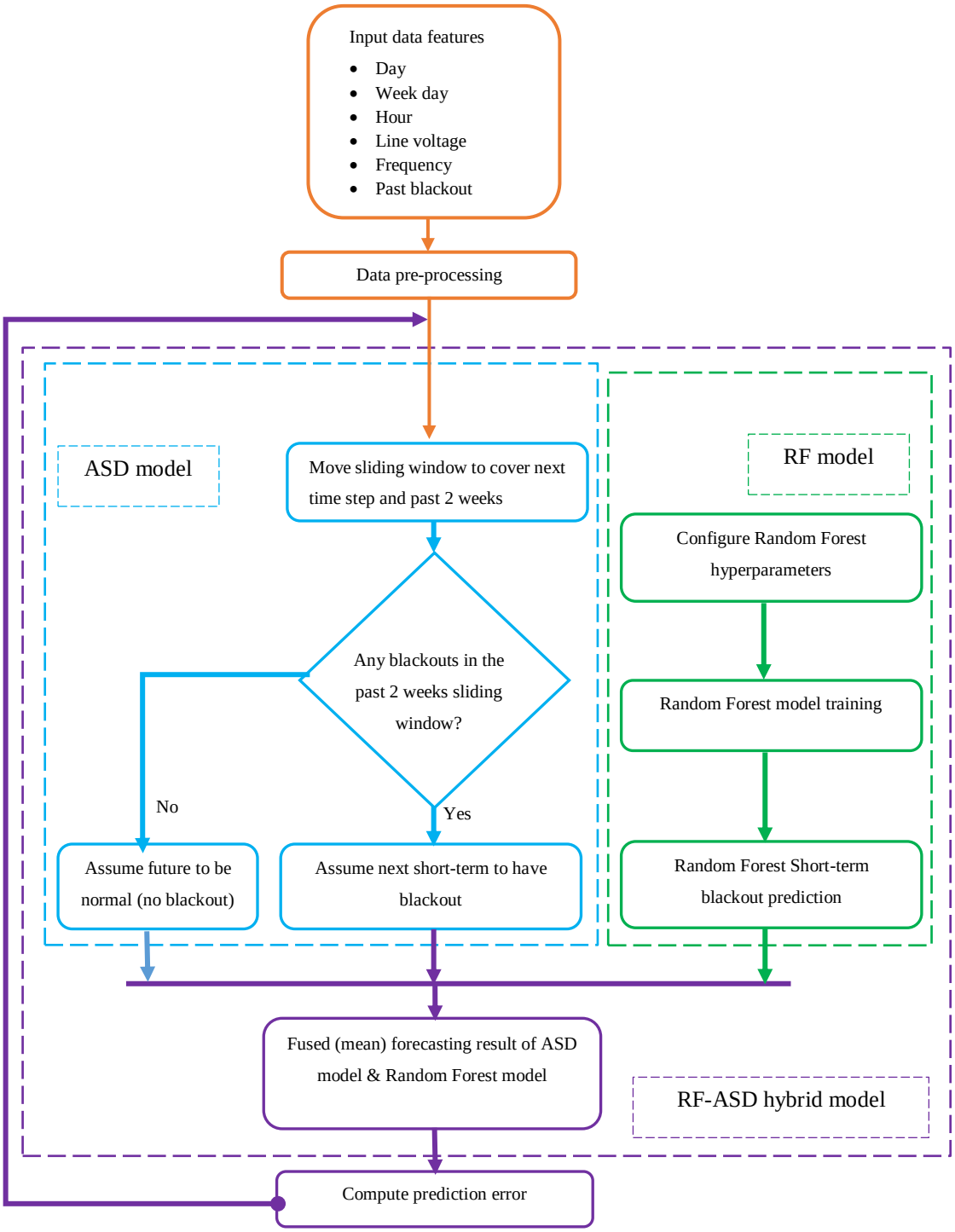


Figure 20: The RF-ASD hybrid model algorithm flow chart

3.8 Model Evaluation Metrics

To evaluate the load forecast and blackout regression algorithm's prediction skill, the mean absolute error (MAE) and root mean square error (RMSE) were employed. For the case of blackout forecast, MAE and RMSE of predicted values were measured only against instances when blackout events were actually observed, in order to focus and gauge the efficacy of the models in blackout prediction since grid power is ON most of the time. The MAE metric

(Equation 4) is more resilient to outliers in the results, whereas RMSE (Equation 5) penalises outliers in the results.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \quad (4)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \quad (5)$$

Whereby, n is the number of load/blackout data points, y_j is the observed load/blackout value and $\hat{y}_j$ is the forecasted short-term blackout value.

R-squared (Equation 6) provides a measure of how well the linear regression model fits the data points on a scale from 0 to 1. However, it does not provide information about the practical significance or meaningfulness of the relationship between the predictors and the response variable. R-squared alone cannot indicate whether the model's predictions are accurate or if the relationship between the variables is meaningful in the real-world context. The R-squared assumes that the model satisfies the underlying assumptions of linear regression, such as linearity, independence of errors, and homoscedasticity (constant variance of residuals). If these assumptions are violated, R-squared may not provide a reliable measure of model performance. Since the forecast challenge in this study is non-linear, R-squared metric was not preferred.

$$R - squared = 1 - \frac{SSE}{SST} \quad (6)$$

Whereby, SSE is the sum of squared error, SST is the sum of squared total.

The MSE (Equation 7), is a commonly used metric for evaluating the performance of regression models, while the MAE provides a measure of the average absolute difference between the predicted and actual values. The MSE gives equal weight to all errors, including outliers. Outliers can have a significant impact on the squared errors due to the squaring operation, which may distort the evaluation of the overall model performance. Therefore, MSE is highly sensitive to outliers and can be influenced disproportionately by them. Squaring the errors in MSE amplifies the effect of larger errors more than smaller errors. Consequently, MSE may prioritize minimizing larger errors at the expense of overall accuracy. This can lead to models that perform well on extreme values but poorly on the majority of the data.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \quad (7)$$

Whereby, n is the number of load/blackout data points, y_j is the observed load/blackout value and $\hat{y}_j$ is the forecasted short-term blackout value.

The MAPE (Equation 8), is a commonly used metric for evaluating the accuracy of forecasting models, as it provides the average percentage deviation between the predicted and actual values. However, it has weaknesses which make it unsuitable for the dataset used in this study, since the dataset set had zero values in some cases. The MAPE becomes undefined when the actual value (denominator) is zero. This can happen when dealing with datasets that contain zero values or very small values. It restricts the use of MAPE in such cases. The MAPE is sensitive to extreme values or outliers in the dataset. Since the calculation involves dividing by the actual value, a large or small actual value can greatly impact the percentage error. As a result, MAPE may not accurately represent the overall accuracy if extreme values are present.

$$MAPE = \frac{100\%}{n}\sum_{j=1}^{n}\left|\frac{y_j - \hat{y}_j}{y_j}\right| \tag{8}$$

Whereby, n is the number of load/blackout data points, y_j is the observed load/blackout value and $\hat{y}_j$ is the forecasted short-term blackout value.

To assess blackout classifier model performance, a confusion matrix was used along with classification accuracy score, precision, recall, and F1 score. The classification accuracy score gives the ratio of correct forecasts to the remaining forecasts, it denotes how accurate the prediction model is. The accuracy score is represented in Equation 6. True positive values are those whose observed value and forecasted value are true. False negative is a misclassification where the observed value is true but the predicted value turns out false. False positive is also a misclassification where the forecast value is true whereas the observed value is false. True negative is the case where the observed value is false and the forecast value is also false.

$$accuracy\ score = \frac{true\ positive + true\ negative}{true\ positive + true\ negative + false\ positive + false\ negative} \tag{9}$$

Recall metric may be defined as the ratio of correctly classified data divided by the total actual samples of the target class. The blackout recall used in this work is represented below by Equation 10. On the other hand, the *precision* metric is the ratio of correct positive predictions relative to total positive predictions. It is represented in Equation 11. A classifier with good precision will not label as positive a data sample that is negative. Another classification metric especially useful in this work due to the imbalanced dataset employed is the F1 score, shown in Equation 12. The F1 score is the harmonic mean of precision and recall. The best F1 score is 1 while the worst is 0.

$$blackout\ sensitivity\ (recall) = \frac{correctly\ predicted\ blackouts}{total\ actual\ blackout} \quad (10)$$

$$Precision\ = \frac{correctly\ predicted\ blackouts}{total\ predicted\ blackout} \quad (11)$$

$$F1\ score\ = 2 \times \frac{(precision \times recall)}{(precision + recall)} \quad (12)$$

3.9 Low-cost EMS Smart Meter and Energy Dashboard Prototype

This section gives an overview of the low-cost EMS smart meter developed in this work that could be suitable for hybrid microgrids even in a rural setting. The main features considered in the design are load forecast and blackout forecast; electrical parameters sensors; Load control; Occupancy detection; battery management; data logging; weather sensors; user engagement; security and privacy. Figure 21 gives a visual summary of the developed EMS smart meter features followed by the description of key features.

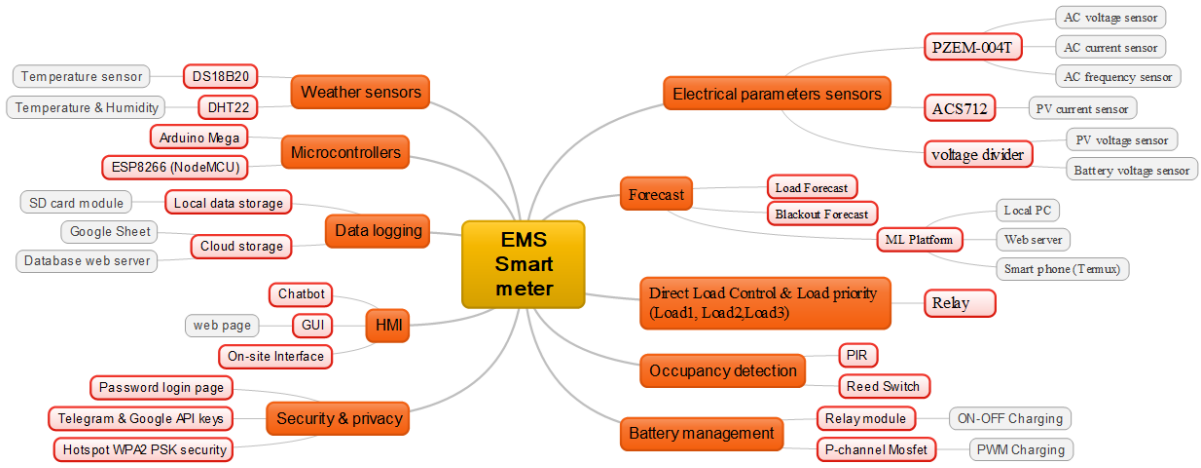


Figure 21: Developed experimental low-cost EMS smart meter main features

3.9.1 Microcontroller

Microcontrollers play an important role in low-cost smart meter designs; they integrate well with various analogue sensors and peripherals. Arduino Mega and ESP8266 (NodeMCU) microcontrollers were employed in the smart meter design because they were available in the local market and they met the project requirement. Arduino Mega was selected because it has many ADC and GPIO pins; thus, being scalable in that, extra sensors and modules could be added if the need arose. The NodeMCU was selected since it has wifi capability and can therefore aid in connecting the smart meter online for remote monitoring and upload of data to the cloud services.

3.9.2 Sensors

Weather measurements can be useful inputs to ML models; the Dallas DS18B20 module was used for temperature measurements whereas the DHT22 module was used for both temperature and humidity measurements. The DS18B20 and DHT22 sensors were interfaced to the Arduino Mega board. One of the core roles of a smart meter is to be able to measure electrical parameters such as the utility grid's AC voltage, Alternating Current, AC frequency; PV voltage and current; and battery voltage in order to estimate SOC. The PZEM-004T electronic module filled the role of measuring AC voltage, current, and frequency. The PV voltage and battery voltage were measured with a voltage divider circuit interfaced to Arduino Mega's ADC pins whereas, PV current was measured by ACS712 hall effect current sensor module also interfaced to the ADC port of Arduino Mega board.

3.9.3 Occupancy Detection

The EMS modules are meant to increase energy efficiency for instance by turning OFF appliances when there is no one using them – to prevent wastage. Therefore, a passive infrared (PIR) sensor and reed switch were used in the developed prototype. The PIR sensor picks up infrared signals emitted by warm bodies of humans; they are useful in detecting the presence of a person in the vicinity. Reed switch was placed at the door to indicate when someone enters or leaves the room; it was also connected to the GPIO pin of Arduino Mega. With the help of these two sensors, Arduino Mega turns off the lights and other non-essential appliances when the room is empty.

3.9.4 Data Logging

Storage of measured sensor data in regular intervals – say minute intervals, is important to an EMS smart meter in order to provide insights that can help improve energy efficiency. Stored historical data are important inputs to machine learning models necessary for load forecast and blackout forecast. In this work, data from the developed smart meter sensors was uploaded to a SQL database using HTTPS protocol via NodeMCU. Sensor data was additionally uploaded to a Google spreadsheet, and also locally stored on an SD (Secure Digital) memory card to prevent loss of data in the event of internet connection problems. One of the office telephones at the prototype pilot site – a smartphone, was used as an internet MODEM; providing internet connectivity to the NodeMCU via password protected internet hotspot.

3.9.5 Human-Machine Interface

A graphical user interface (GUI) goes a long way in helping a user increase energy efficiency; it can provide the end user with an energy dashboard that raises awareness of energy consumption and forecasts. This way the end user gets a tool to visually aid in counting the cost of energy and help make informed better decisions. Energy dashboards implemented in this work were: A web page; Telegram chatbot; ThingsBoard; google spreadsheet and Google Apps Script; and an on-site hardware interface. The main energy dashboard used in this work was a dynamic web page that interacts with the MySQL database sensor data repository. The webpage was developed using HTML, CSS, and JavaScript for the front-end; whereas PHP language was used on the back-end – server side. A password-protected login page was added to the web page energy dashboard portal. Since there is a growing number of people on social media, Telegram was used as a tool to engage the end user and provide an interactive user interface, thanks to its chatbot API that conveniently integrates with IoT devices allowing remote monitoring and control of the smart grid. The community version of ThingsBoard was tested, however this only provided monitoring functions, there was no option for controlling appliances remotely as was the case with the web page developed, telegram chatbot, and Google spreadsheet. Google spreadsheet and Google Apps Script were linked with NodeMCU with the daily spreadsheet sensor generated was saved automatically saved on Google drive. Both Telegram, Google spreadsheet, and Apps Script use world-class security features implementation, thus ensuring privacy when using these services in the developed EMS smart meter prototype.

3.9.6 Low-cost Machine Learning

The ML can be computationally intensive and at times IoT applications require ML elements, therefore, numerous cloud services have sprung up. The ML learning model development was primarily done on a desktop personal computer (PC) using Matlab programming language and Python language; however, the developed ML algorithm was successfully run on a shared hosting web server in the cloud that remotely communicated with the smart meter and visually offered load forecast and blackout forecast charts to the end user. This was possible because the web server was Linux OS based, thus, able to run Python open-source ML libraries such as Pandas, Matplotlib, and Scikit-learn. In testing a real-world deployment of the developed ML model, the Python ML scripts were run daily on the web server at predetermined intervals via a task-schedular (cron job) in order to produce a short-term load forecast and blackout forecast. Since the smart meter prototype pilot site used a smartphone as an internet modem, the same

Android mid-range smartphone was also used to capitalize on its ML potential tool via the Termux app. Python open-source ML-related libraries were installed and used on the test site dataset. In order to realize a low-cost ML platform Termux app running ML libraries could potentially be linked to cloud storage services such as Google drive.

3.9.7 Direct Load Control and Prioritization

Appliances in the pilot site where the prototype was deployed were connected to Arduino Mega via three electrical relay modules corresponding to 'load 1', 'load 2', and 'load 3' classification of loads. 'Load 1' is the primary load of highest priority whereas, 'load 3' is of lowest priority. Under this scheme, the on-premise video surveillance system was placed under 'load 1' category; the implication being that the surveillance camera system is the last to be turned off in case of a power outage and when battery SOC is low. The user has an option to turn the loads ON/OFF remotely via an internet-connected NodeMCU linked to Arduino Mega. A second NodeMCU was used in a relay-based changeover circuitry in order to remotely choose a power source – between PV-inverter or TANESCO grid power.

3.9.8 Battery Management System

The EMS smart meter prototype was deployed at a site with PV battery already installed. Solar batteries – in this case, lead-acid batteries, need to be protected against overcharging and over-discharging; the two things that kill a battery and reduce its lifespan. A PWM battery charging system was developed as part of the smart meter. Figure 22 shows the proposed battery management algorithm. The algorithm works under the assumption that the site has an undersized grid-connected solar system installed that is only capable of supporting a few light loads, with heavy loads connected to the main grid. This is so because, in emerging countries, you may have a consumer connected to a weak grid that experiences short-lived power cuts for a few hours, therefore, consumers opt to deploy a backup solar system that continues to support a few important loads during a power outage. Since the smart meter performs a blackout forecast, therefore, the battery can be maintained near full charge (by float charging) if a blackout is imminent as per generated forecast or allowed to be discharged to low SOC if no power outage is imminent. The PWM charging signals were produced by one of the PWM pins of the Arduino Mega into a P-channel MOSFET (IRF9540N) high-side switch battery charging circuitry; a circuit that can be operated under the MPPT charging principle. Additionally, a relay module interfaced to NodeMCU was also connected between the PV array and the battery to control charging by automatically disconnecting/reconnecting the PV array and another for engaging/disengaging the main grid charging of the batteries.

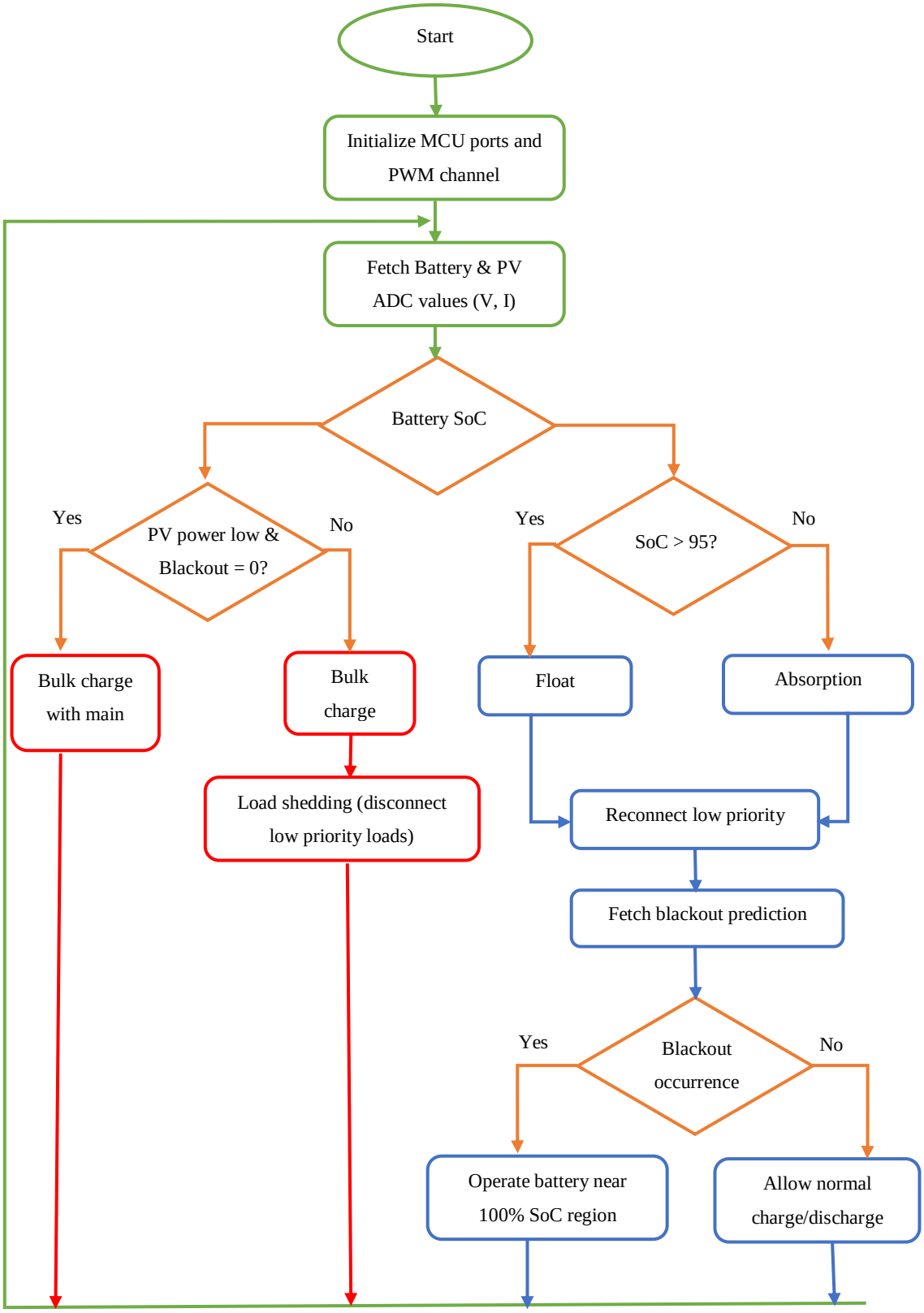


Figure 22: Proposed battery management algorithm employing blackout prediction

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Load Forecast Results

This section shows the results of the comparative analysis on the impact of day-ahead (24hrs) load profile forecasting with respect to variations in input features and prediction models. This results in different forecasting strategies namely: (a) load forecast using all the 23 features; (b) load forecast using features from standard FS such as Random Forest, ReliefF algorithm, ensemble regression, compact tree, NCA, (c) proposed FS methods (principal k-features union, principal k-features mean score, refined exhaustive features search); and lastly (d) hybrid load forecast models formed by fusion of two standard regression models through elementwise mean, max, min of the two models outputs.

Figure 23 (a) shows load forecast evaluation using the test dataset on the 20 conventional prediction algorithms. Each of the 8 feature selection methods in Fig. 23 (a) was applied to all the 20 conventional prediction models. The principal k-features union approach performed best with 'linearSVM' prediction model, however it performed poorly in the case of: 'FineGSVM'; 'MaternGPR'; 'rqGPR'; and 'eGPR' models. This observed poor performance is contributed by the 'All features MAE' being an inherent part of the k-features union. The principal k-features mean score approach performed relatively good across all the 20 prediction models. The authors, Dai and Zhao (2020), developed a SVM-PSO-based day-ahead load forecast model using minimum redundancy maximum relevancy (mRMR) approach which also gave attention to real-time pricing as an essential feature of Singapore grid data. Their model outperformed other NN models used as their benchmarks, however, their model only used mRMR feature selection it failed to compare performance with other feature selection methods. It also failed to show how the model would perform on a smaller microgrid like the one used in this work. Phyo and Jeenanunta (2022), developed an LR-SVR ensemble STLF model which outperformed DNN, LSTM, and LSTM-CNN deep learning models. Only Spearman's correlation coefficient was used for input features selection, their work failed to compare the impact of other feature selection methods, and limited their STLF comparison to the three deep learning models.

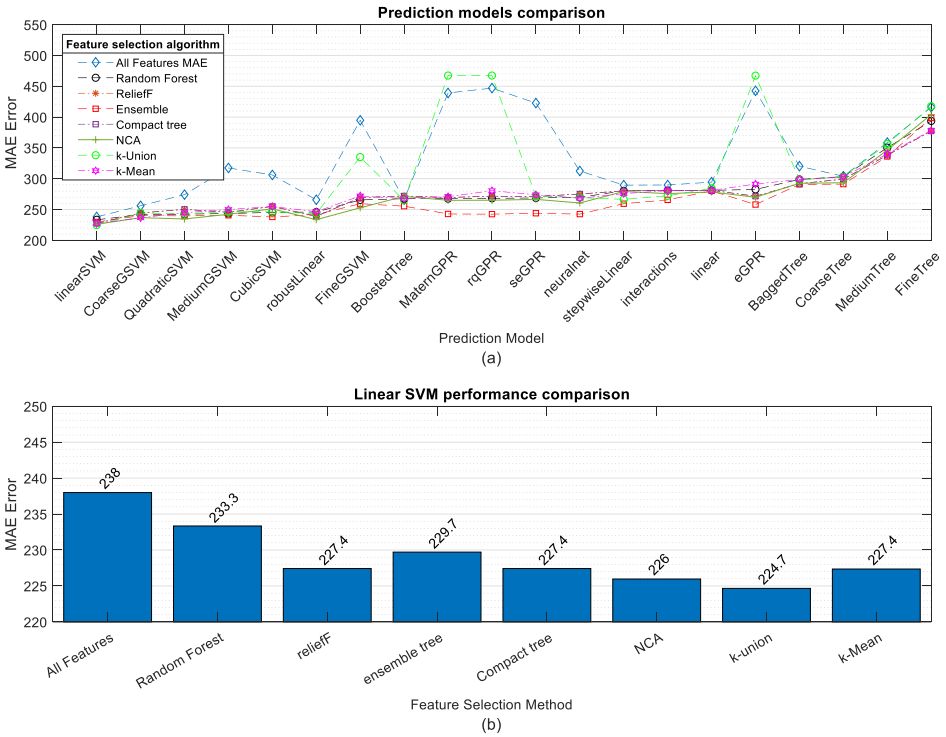


Figure 23: (a) Prediction models evaluation; (b) Linear SVM performance comparison on each of the 8 feature selection methods

Figure 24 shows a prediction algorithms comparison using a subset of predictors obtained from the proposed k-Union feature selection method. Predictors used were 'Load', 'T1_Load', 'Hour', 'T2_Load', 'Day', 'P_DG', 'Month', 'Vdc_bus', and 'P_inv'. Generally, the performance trend of the 20 algorithms tested was the same across the five-evaluation metrics considered, namely: MAE, RMSE, MAPE, MSE, and R-squared. The MAE and RMSE are robust against outliers and are widely used in the literature, therefore they were the preferred metric of evaluating the prediction models. The MAPE is not suitable when dataset contains zero values. The load profile used in some instances contained zero values. Electric load forecast problems are typically considered as non-linear regression problems rather than linear regression problems. This is because the relationship between the independent variables (such as weather conditions, time of day, day of the week, etc.) and the dependent variable (electric load) often exhibits non-linear patterns and interactions. Thus, R-squared metric is not well suited for non-linear regression models. The MSE is usually larger than MAE since it squares the difference between predicted and actual values. This is also evident in Fig. 24. The MSE is more suitable when large errors should be heavily penalized or when optimization algorithms that require differentiability are used. On the other hand, MAE is preferred when outliers should be downplayed or when interpretability and equal weighting of errors are crucial. The dataset used is from a small microgrid susceptible to many outliers, therefore MAE was preferred over MSE. Notably, ‘linearSVM’ algorithm had an overall good performance in all the five error evaluation metrics.

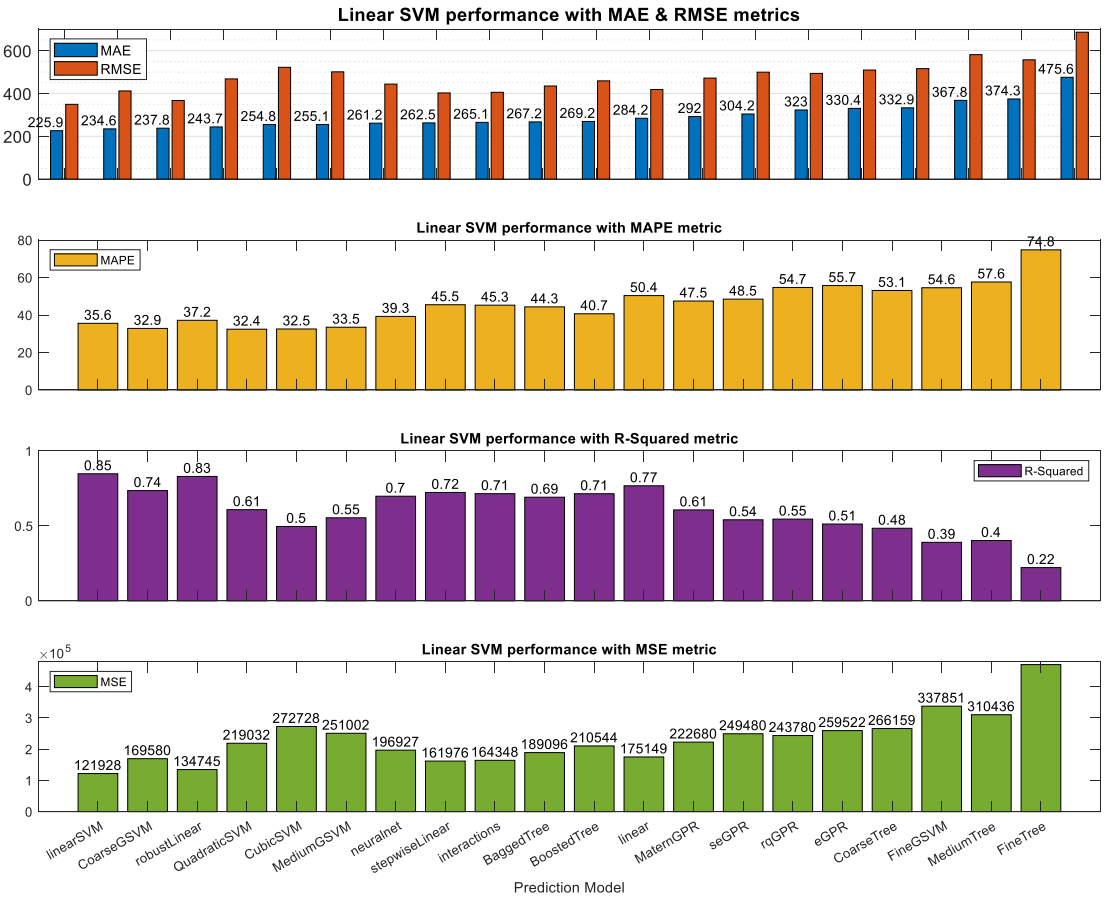


Figure 24: Evaluation error metrics comparison on the LF prediction models

The Linear SVM prediction algorithm was identified as the best prediction model. For prediction examples, Fig. 25 shows individual next-day load forecast plots for February 3rd to February 6th 2018 using the 'LinearSVM' prediction model. The forecast was done with 95% confidence band. The prediction model is used with the features selected from the principal k-features union approach. Prediction accuracy varies from one day to the next. However, the actual consumption pattern modestly lies within the predicted confidence band. Agrawal *et al.* (2018), performed a long-term load forecast based on LSTM networks. Similar to this work, the authors also used 95% confidence bands, however, their target was a yearly forecast horizon using a large dataset (the ISO New England public dataset). The authors Zhai and Che (2022), reported good performance of their PSO-SVR STLF model using real-life data with MAE, RMSE, MAPE, and R^2 (coefficient of determination) evaluation metrics. In their work, missing values were filled with predictions from RF. Although the authors did not employ confidence interval bands, their observations agree with the findings of this work, in that, the prediction skill of the model with feature selection is better than without.

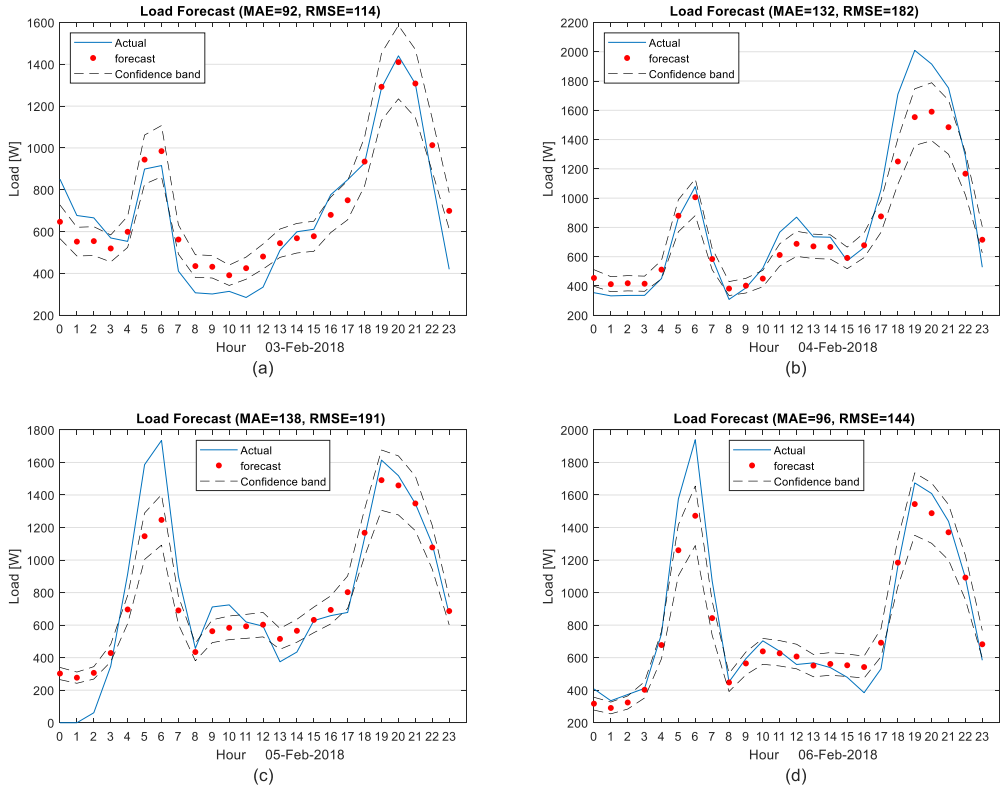


Figure 25: Linear SVM Forecast model performance on Ngarenanyuki dataset

Table 5 shows the top 20 next-day load forecast results obtained using the refined exhaustive search and Linear SVM prediction model. The refined exhaustive search involved choosing 5 features out of all 23 features and using the chosen 5 features on the 'LinearSVM' prediction model. In other words, referring to equation (2), a 5 features subset of 23 features set was used in predicting day-ahead load forecast using 'LinearSVM' prediction model. This resulted in a total of 5989 different combinations of 5 features which at least include the 'Load' variable but exclude the 'Year' variable. The latter was correctly omitted since it has no meaningful influence on day-to-day short-term load forecast because it remains constant from one day to the next. Bouktif *et al.* (2018), made a comparison between machine learning models and LSTM deep learning models in load forecasting using feature selection and GA. They found that their LSTM model outperformed machine learning models, however, their model was only tested on one dataset. They experimented with different forecasting horizons, from 2 weeks to 4 months, but they failed to report the performance of their model for day-ahead forecasts. The authors (Hu *et al.*, 2015), developed an STLF model based on SVR for load forecast and partial mutual information coupled with the firefly algorithm for feature selection. Their work successfully demonstrated the viability of using filter methods and wrapper methods for feature selection. However, their work failed to extensively compare their feature selection with other established feature selection methods, and only used SVR as a modeler without considering other established prediction models.

Table 5: Refined exhaustive search load forecast results

#	Feature subset	RMSE	MAE
1	'Load,SOC,Vdc_bus,T1_temp,Weekend'	283.3	125.8
2	'Load,RHumidity,SOC,Vdc_bus,temp'	289.6	127.9
3	'Load,SOC,Vdc_bus,T1_temp,DewpointTemp'	288.5	129.1
4	'Load,SOC,Vdc_bus,T2_temp,temp'	289.8	131.5
5	'Load,Vdc_bus,DewpointTemp,atmPressure,Weekend'	296.3	132.3
6	'Load,RHumidity,SOC,Vdc_bus,Weekend'	295.1	132.6
7	'Load,SOC,Vdc_bus,T1_temp,P_HYD'	292.4	133.5
8	'Load,SOC,Vdc_bus,P_HYD,P_DG'	284.7	133.7
9	'Load,SOC,Vdc_bus,temp,P_HYD'	300.2	133.9
10	'Load,SOC,Vdc_bus,atmPressure,Weekend'	295.4	134.0
11	'Load,RHumidity,SOC,Vdc_bus,T1_temp'	300.1	134.2
12	'Load,SOC,Vdc_bus,P_HYD,Weekend'	294.8	134.4
13	'Load,RHumidity,SOC,Vdc_bus,P_HYD'	302.6	134.6
14	'Load,SOC,Vdc_bus,DewpointTemp,Weekend'	290.0	135.4
15	'Load,RHumidity,Vdc_bus,T1_temp,Weekend'	302.4	135.6
16	'Load,Vdc_bus,T1_temp,DewpointTemp,Weekend'	305.6	136.2
17	'Load,SOC,Vdc_bus,WeekDay,Weekend'	298.6	136.7
18	'Load,SOC,Vdc_bus,T1_temp,P_DG'	294.5	136.7
19	'Load,SOC,Vdc_bus,WeekDay,atmPressure'	300.8	137.0
20	'Load,SOC,Vdc_bus,DewpointTemp,P_HYD'	302.1	137.0

Figure 26 (a) shows the MAE and RMSE prediction result evaluation when a subset of top 5 most important features selected from mean score votes of the 6 FS conventional models was trained and tested on each of the 20 regression models. Figure 26 (b) shows a general improvement in MAE error performance when hybrid models are formed by the fusion of two regression models through elementwise mean, max, and min of the two conventional regression models outputs. The resulting hybrid models formed from the maximum forecast instances of two regression models exhibited the lowest MAE prediction error. This is possibly because the load profile of the microgrid being considered fluctuates a lot, any appliance that is turned on or off the effect is immediately observable in the baseline load profile, and is therefore likely to be captured by the ‘mae-max’ hybrid model. Overall, the results agree with error reduction enhancements typically obtained from STLF model aggregation (Feng & Zhang, 2020).

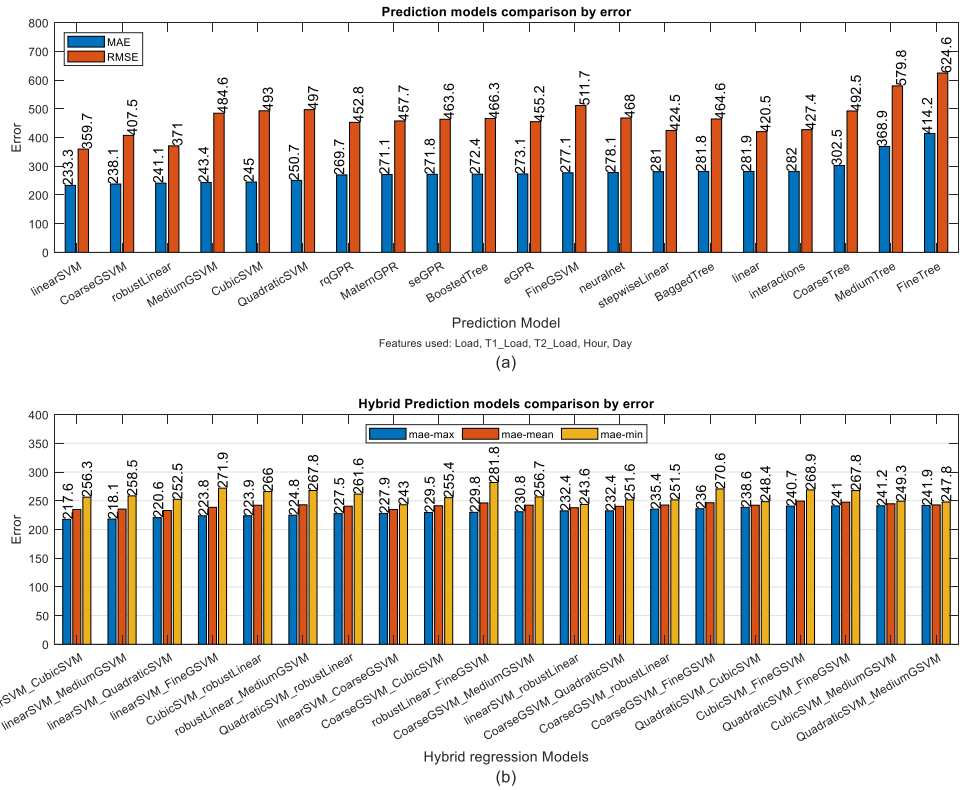


Figure 26: (a) Conventional prediction models evaluation; (b) Hybrid regression models

Based on the microgrid dataset used in this work, both the k-features mean score and subset union approaches registered the lowest error values when they were used with 'linearSVM' prediction model. The principal k-features union approach model registered MAE error of 224.7 while the principal k-features mean score approach registered 227.4. The refined exhaustive search used together with 'linearSVM' prediction model registered the lowest MAE error of 125.8, however, it was computational more intensive. Furthermore, a hybrid prediction model formed from the elementwise maximum forecast instances of two regression models yielded better MAE prediction error than the individual regression models fused to form the hybrid. In this case study, the overall best prediction model was found to be the hybrid regression model formed from 'linearSVM' and 'cubicSVM' regression models and showed improved prediction performance than the individual regression models, MAE was reduced by 5.4%. Therefore, given a different microgrid it is recommended to find the best features using the proposed principal k-features union approach and in turn form a hybrid regression model based on the top two performing conventional prediction models.

With regards to load forecast, at times it is not straight-forward to do a direct comparison between load forecast study cases because the performance of the model is dependent on data (which may not be publicly available) it was trained on, platform, and computer specifications where the model was developed. In the study by Eseye *et al.* (2019), the authors proposed using binary Genetic Algorithm (BGA) and GSR in feature selection when undertaking load forecast

procedure. Although the model performed better than the conventional NCA FS method, the study failed to compare the proposed BGA-GSR with other bio-inspired optimization algorithms like PSO, and Ant Colony Optimization (ACO). Chen *et al.* (2019), implemented an hour-ahead load forecast RF classifier model that converts the forecasted numerical value into 3 energy levels (high, medium, low) using percentile ordinal partitioning in order for the consumer to easily comprehend the model's output. The accuracy of the RF classifier model decreased with increased energy levels when 7 levels were used. This study performed LF using 24 hours ahead regression with RF for feature selection instead of the 1 hour ahead forecast employed in their work. However, their work could serve as an alternative extension to this study.

Yildiz *et al.* (2018), performed a short-term load forecast of residential loads using ANN and SVM models. They tested the impact of different data resolutions and forecast horizons on the model's prediction ability. They found that higher input resolution data yielded results with more error than low-resolution data whereas the short forecast horizon had more accuracy than the distant forecast horizon. This agrees with the observations made in this study. Moon *et al.* (2020), showcased an STLF model that uses an N-number of base models for prediction which in turn feeds a meta-model that trains on the received values in order to give a final more accurate output. The stacking ensemble model approach they proposed used 4 DNNs. The DNN models are effective on large datasets, this dissertation used a small dataset and a maximum of two base models without stacking. Stacking and using many base models increases complexity and computation time.

4.2 Power Outage Forecast Results

Figure 27 below shows the blackout classification accuracy score for RF, ASD, and RF-ASD models. The results are for a 15-minutes-ahead power outage forecast. The three models showed an overall accuracy score of about 90%. However, because most of the time power is available, observed actual blackout events were fewer in the test dataset, inevitably resulting in an imbalanced dataset. Therefore, the machine learning models inadvertently also get few effective observed blackout samples to train on. It is ideally desired for any classifier model to classify all samples appropriately as True Negatives (TN) and True Positives (TP) – the two diagonal parts of the 2x2 confusion matrix in Fig. 27.

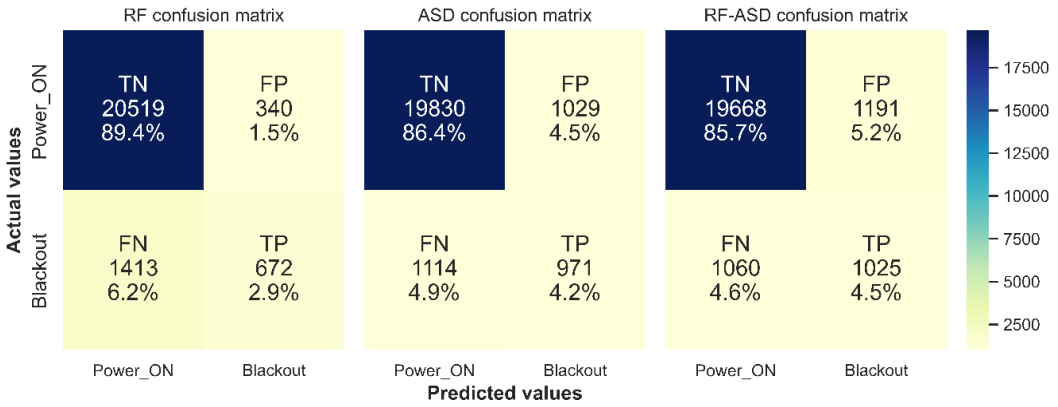


Figure 27: Power outage classification accuracy scores for 15-minutes-ahead predictions

The stand-alone RF model correctly classified 89.4% of the test data samples as the power available (TN), while 1.5% were misclassified as power outages (FP) instead of being predicted as instances where the supply line had power ON (available). The RF model misclassified 6.2% of the blackout events and mistook them for power ON instances instead of blackouts. The RF model only correctly predicted 672 power outages out of the total 2085 blackout events. These 2085 blackouts formed only 9.1% of the total samples, whereas actual power was available/ON 90.9% of the recorded data. The RF model did relatively better at predicting the presence of power and fared badly at predicting blackout events as compared to the ASD model and RF-ASD hybrid model. The ASD model did better than the RF model in predicting blackouts, it accurately predicted 971 counts of blackout events. The RF-ASD hybrid model predicted accurately almost half of the blackouts (1025 counts), thereby performing slightly better than the RF, and ASD models.

Typically, in emerging countries scenario, the user may want to know in advance if there is going to be any power outage the following day in order to take appropriate actions to alleviate the effects of lack of electricity supply. For this, the 24 hrs-ahead blackout predictions may be useful. The hour-ahead and 15-minutes ahead blackout prediction may be advantageous to a grid operator. Table 6 summarizes the overall blackout forecast classification of the 3 models RF, ASD, and RF-ASD along 3 forecast horizons namely: 15-minutes ahead, hour-ahead, and 24 hrs-ahead forecast horizon. Considering firstly accuracy score metric, it was found to be 92.4%, 90.6%, and 90.2% for RF, ASD, and RF-ASD models respectively for the 15-minutes-ahead forecast horizon. These are high accuracy scores for a classification model; however, they have been driven up by the majority class (power ON class data) instead of the blackout minority class. The accuracy score is also high for the hour-ahead forecast horizon. The accuracy score was found to be low for the 24-hours-ahead forecasts due to the stochastic nature of blackouts. The ASD model fared better with an accuracy score of 85%. Attempting to predict a blackout many time steps in advance is more prone to forecast errors. As was found

to be the case in 24-hours-ahead blackout prediction. All in all, the accuracy score, in this case, does not correctly reflect the performance of the model in predicting blackouts, which is our target.

Looking at the 15-minutes-ahead forecast horizon, the recall metric was found to be 32.2%, 46.6%, and 49.2% for RF, ASD, and RF-ASD models respectively. For our case the higher the recall value the better, as it implies the model was able to predict accurately more blackout events. Combining the RF and ASD model to form RF-ASD had a good effect of increasing blackout recall by up to 49.2%. The same effect was observed in the case of hour-ahead and 24-hours-ahead forecasts. Just like recall, it is desired to have a model with a higher precision value. The RF model had higher precision values than ASD and RF-ASD models in both 15-minutes-ahead and hour-ahead forecast horizons. However, the ASD had higher precision in the 24-hours-ahead horizon. The F1 score is a suitable metric for identifying an overall good-performing classifier model for an imbalanced dataset as was the case in this work. The RF-ASD model scored slightly higher with an F1 score of 47.7%, with the other models lagging. Therefore, making the RF-ASD model a better candidate for 15-minutes-ahead blackout forecasting. In hour-ahead forecasts, the RF model lagged in performance compared to ASD, and RF-ASD which both scored 45.7%. The ASD model had outperformed RF and RF-ASD models in the 24-hours-ahead blackout forecast.

Table 6: Overall blackout forecast classification performance

Forecast horizon	classifier model	Accuracy score	Blackout sensitivity (recall)	Blackout precision	F1 score
15-minutes-ahead	RF	92.4%	32.2%	66.4%	43.4%
	ASD	90.6%	46.6%	48.6%	47.6%
	RF-ASD	90.2%	49.2%	46.3%	47.7%
Hour-ahead	RF	88.4%	17.0%	52.1%	25.6%
	ASD	87.3%	45.7%	45.8%	45.7%
	RF-ASD	86.8%	47.2%	44.3	45.7%
24hrs-ahead	RF	62.0%	67.8%	19.4%	30.2%
	ASD	85.0%	38.0%	36.6%	37.3%
	RF-ASD	60.9%	73.4%	19.4%	30.7%

Figure 28 gives insight into the performance of the blackout classifier across different months. The chart below considers the RF-ASD classifier model for 15-minutes-ahead predictions. The overall accuracy score is observed to be relatively high from March to October, where it then drops to 64% in November and 57% percent in December.

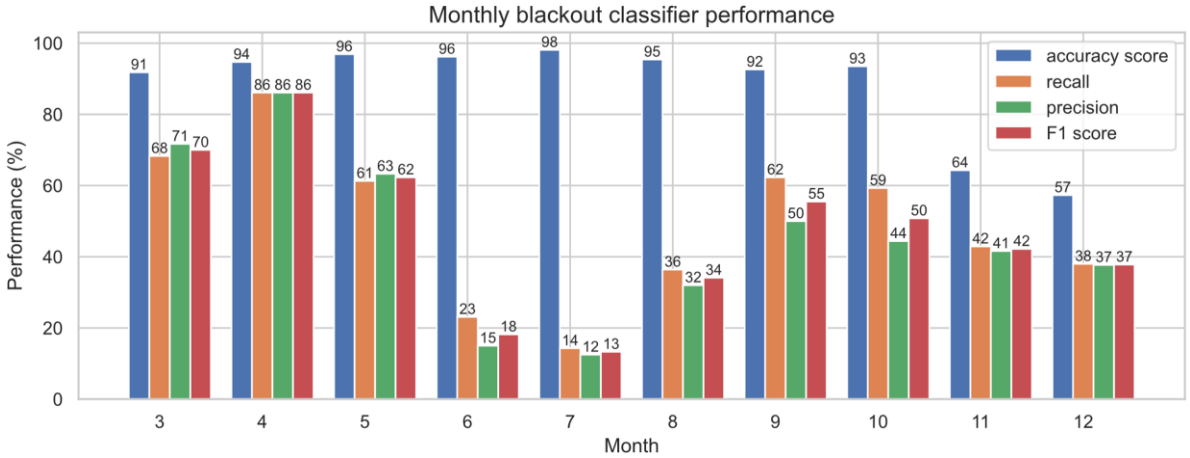


Figure 28: Monthly blackout forecast performance for 15-minutes-ahead RF-ASD classifier model

The F1 score is derived from both recall and precision metrics and is a better indicator of the performance of a classifier. The F1score for March, April, and May perform fairly well above 50%. However, it drops to 18%, 13%, and 34% in June, July, and August. In contrast, the accuracy score remains high above 90%. With respect to Fig. 16 of Section 3, June, July, and August had less consistent blackouts which occur sparsely and thus affecting negatively the performance of the model in predicting blackouts. For this reason, the classifier model performed poorly in the recall, precision, and F1 score metrics. November and December had many random blackouts thus causing the classifier model to perform poorly in accuracy, recall, precision, and F1 scores. Although March and April had about the same level of blackouts as November and December, blackouts in March and April were contiguous and more converged, unlike those of November and December which were dispersed. Therefore, the classifier had better prediction skills in those earlier months than in November and December.

Table 7 summarizes the performance of the RF, ASD, and RF-ASD regression models. The three models were formulated as regression models in order to tackle the challenge of quantifying the duration of the short-term blackout prediction. The overall prediction skill for the three models had small differences from each other in all the 3 forecast horizons considered. For the case of 15-minutes-ahead forecasts, the ASD model slightly improved with respect to both MAE and RMSE. Although the RF model had a similar MAE score to the RF-ASD model, its RMSE was slightly worse than that of RF-ASD. Implying that the output of the RF model suffered more from outliers. For the case hour-ahead forecasts, the RF model had a lower MAE value, but the RF-ASD model had the lowest RMSE value meaning that the RF model suffered from slightly more outlier results than the RF-ASD model. The ASD model had slightly better MAE and RMSE values than RF and RF-ASD models for the case of 24hrs-ahead forecasts. For the case of 15-minutes-ahead forecasts, 96 (24x4) predictions had to be made per day, while the hour-ahead approach requires only 24 predictions per day. Statistically, the 15-

minutes-ahead approach ends up being more prone to forecast errors than the hour-ahead forecast approach. This is visible in Table 7 results.

Table 7: Overall performance for blackout forecast regression models

Forecast horizon	Regression model	MAE	RMSE
15-minutes-ahead	RF	0.253	0.333
	ASD	0.251	0.325
	RF-ASD	0.253	0.329
Hour-ahead	RF	0.169	0.298
	ASD	0.203	0.323
	RF-ASD	0.185	0.296
24hrs-ahead	RF	0.268	0.431
	ASD	0.254	0.409
	RF-ASD	0.258	0.418

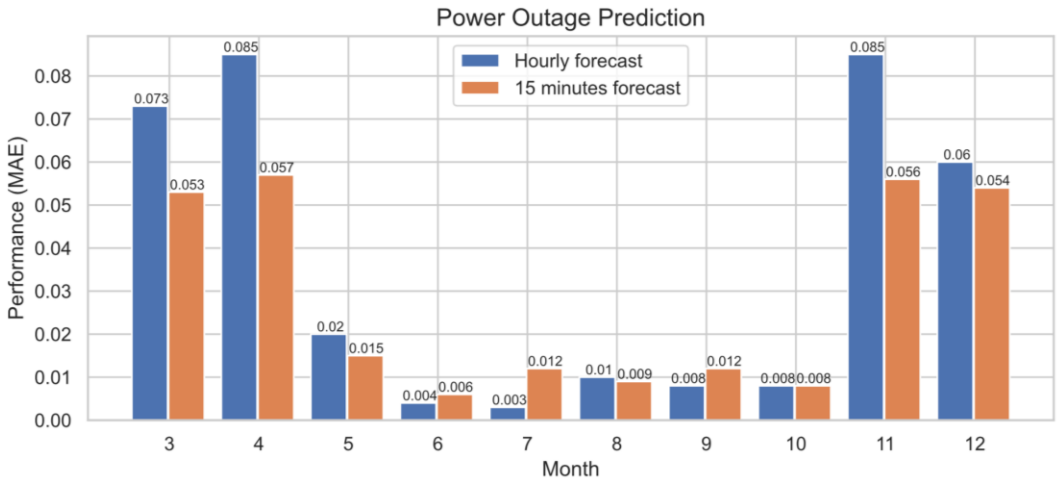


Figure 29: Performance of the RF-ASD blackout forecast model on different month's data, considering the hour-ahead, and 15-minutes-ahead forecast horizons

Figure 29 gives further insight into the average performance of the RF-ASD regression model compared across different months of the test data. All the models performed relatively poorly in March, April, November, and December whereby the grid was under severe blackout disturbances, and also due to the haphazard nature of the outages. It is worth noting that, in the preceding named months where blackouts were prevalent, 15-minutes-ahead forecasts outperformed their counterparts, namely, 24 hours-ahead forecasts, and hour-ahead forecast. The RF-ASD 24 hrs-ahead prediction produces a blackout prediction for the entire 24hrs of the next day, whereas the RF-ASD hour-ahead blackout prediction algorithm only makes prediction one hour ahead at a time, and is, therefore, able to update and notice the developing

blackout trend of the grid. Thus, gaining prediction skills and being able to self-learn any prior inaccurate predictions.

Figure 30 shows the blackout forecast for three reference days – April 17th to 19th 2021, whereby the grid was under severe blackout disturbances. In this hybrid regression model case, the RF-ASD hour-ahead short-term blackout prediction performed better than the RF-ASD 24 hrs-ahead prediction, by almost a factor of 4. Although past values influence future values, the model is not entirely persistent, since short-term forecast values are not in all cases replicas of past observed values.

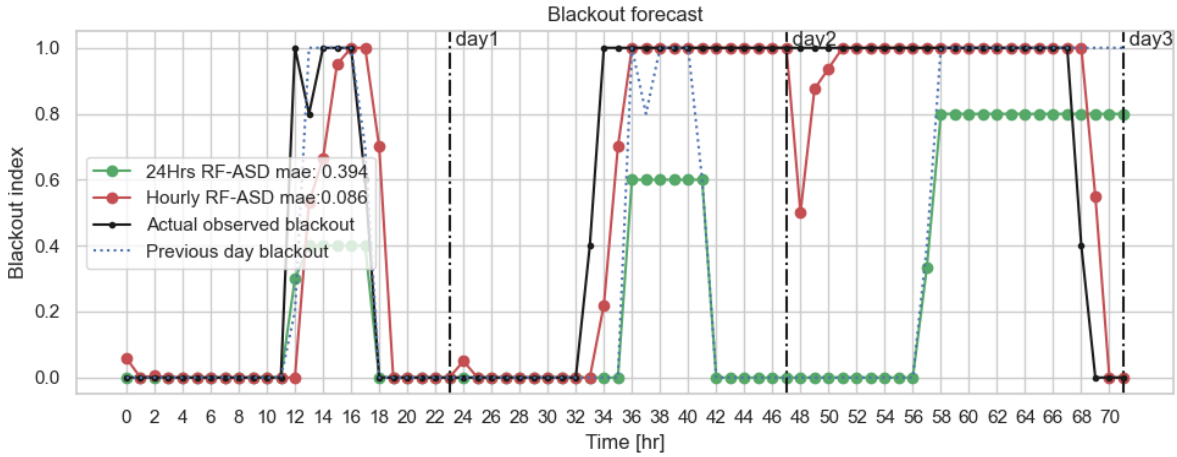


Figure 30: Blackout forecast plot for three reference days

The models developed in this work were reasonably able to predict power outages using only few information regarding electric power parameters as observed from the point of coupling at the customer side and whilst having no prior information of scheduled power outages nor weather forecasts regarding any looming extreme weather events that might cause a blackout. It is more difficult to predict the duration of the blackout event than the mere occurrence of the blackout. Generally, the shorter the forecast horizon the more realistic and practical the prediction result is than when a long forecast horizon is used since grid dynamics may quickly change between one forecast and the next.

With regards to blackout forecast, Kogo *et al.* (2014) conducted similar research on demand-side blackout prediction their goal was merely a day-ahead prediction of power cut start time and not the duration of the power cut whereas this study addressed the prediction horizon of 15 minutes, 1 hour, and 24 hours ahead. Similar to this study they used a similar-day approach of day-ahead power outage prediction assigning power cut binary value. The performance of their statistical model had a high prediction success ratio for times when blackouts were heavy (frequent) whereas they obtained a low prediction success ratio for times when blackouts were light (less frequent), this phenomenon resembles results obtained in this study. Although this

study aimed at a hybrid model combining a statistical approach (ASD) with machine learning (RF). Their study did not use a publicly available dataset; therefore, it is difficult to judge how the model in this study would fare given their dataset. In addition, their work used hit ratio and inclusion ratio as evaluation metrics in contrast to this study.

Other researchers have used RF for blackout prediction, for example, Nateghi *et al.* (2014c), used random forest with reasonable accuracy for blackout prediction however their model focused on the blackout of the large power system from the utility point of view and it targeted blackouts emanating from hurricanes. In contrast to the approach adopted in this work, their model used many inputs including the poles quantity, transformers, switches, distribution line length, served clients quantity, wind speed, and tree trimming variable. So far, it is not easy to compare the blackout prediction model from one author to the next since in most cases the dataset used involves power system inventory information for the geographical area where the model is developed upon.

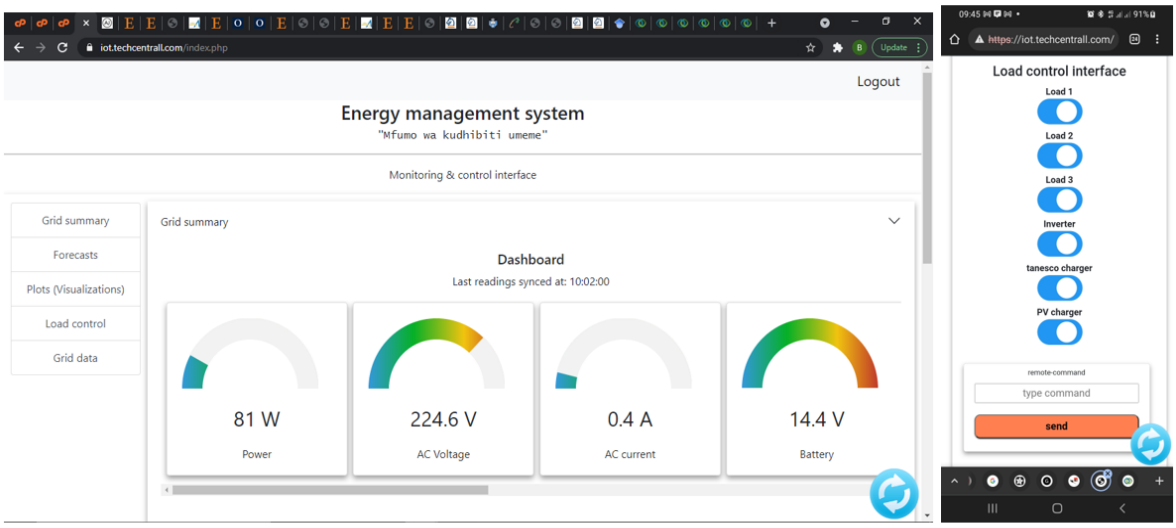
Kankanala *et al.* (2014), used a boosting algorithm ADABOOST^{+} on a large dataset (6 years data) to predict power outages due to wind and lightning in overhead distribution systems of four cities in Kansas. The concept of boosting used in their work is similar to this work, whereby weak learners are combined to form a strong learner. Not surprising for the power outage prediction challenge, their model had low accuracy for sparse outage data as was the case in this study. The ensemble (hybrid) in their work was based only on boosting weak learner neural networks to form a final strong learner neural newtowork with less MSE error, whereas, this study used ASD and RF models to form a hybrid RF-ASD model. A common trait from the study by Kankanala *et al.* (2014), and other studies that address power outages from a utility point of view is the inclusion of weather data in the model's input (Davidson & Stedinger, 2005; Li *et al.*, 2010; Liu *et al.*, 2008; Zhu *et al.*, 2007).

4.3 EMS Smart Meter Prototype Results

The test site microgrid deployed in this work is connected to the TANESCO utility grid under the prepaid meter scheme. Under this scheme, a customer is provided with a prepaid meter console where they can manually recharge purchased electric units and continue using electricity till the time purchased units run out. When a customer's purchased units run out electricity supply is automatically cut-off, unless the customer recharges by purchasing new electric units, otherwise, the customer experiences a power outage. The TANESCO's prepaid meters do not send notifications to the customer of the imminent power outage due to energy credits running out. The smart EMS deployed in this work prevents the consumer from

mistakenly thinking that the power outage is from TANESCO, or even preventing power outage due to finished energy credits. The smart energy management system does so by continuously monitoring power consumption status based on the data logged. If energy credits are low the user is notified to recharge in order to prevent a power outage. This feature does not currently exist in TANESCO's prepaid meters. The energy credit balance alerts are sent to the user's phone via a Telegram app chatbot. The chatbot provides real-time grid system information regarding day-ahead consumption and blackout forecast, and the ability to remotely control turning on/off loads (appliances).

This work proposes leveraging any combination of the following four options as an energy dashboard for monitoring and reporting purposes in emerging countries scenarios, which have been tested: (a) using a hosted shared web server shown in Fig. 31 and 32; (b) using cloud services such as Google apps services (shown in Fig. 33), which includes spreadsheet and Google drive storage which can be utilized for grid monitoring and control; (c) Using IoT platform services like Things Board (Shown in Fig. 34); (d) using chatbots services such as Telegram's chatbot (shown in Fig. 35a) which easily engages the user from their mobile phone. Apart from remotely controlling appliances, the user can select the battery bank charging option between TANESCO or PV (when there is enough sunlight), in order to ensure sufficient backup in case of power outage; (e) using apps like Termux (Fig. 35b) for machine learning, this can provide basic computation needed to perform forecast from the user's mobile phone.



(a) (b)

Figure 31: Energy dashboard web portal with: (a) Grid parameters monitoring, (b) Remote load control

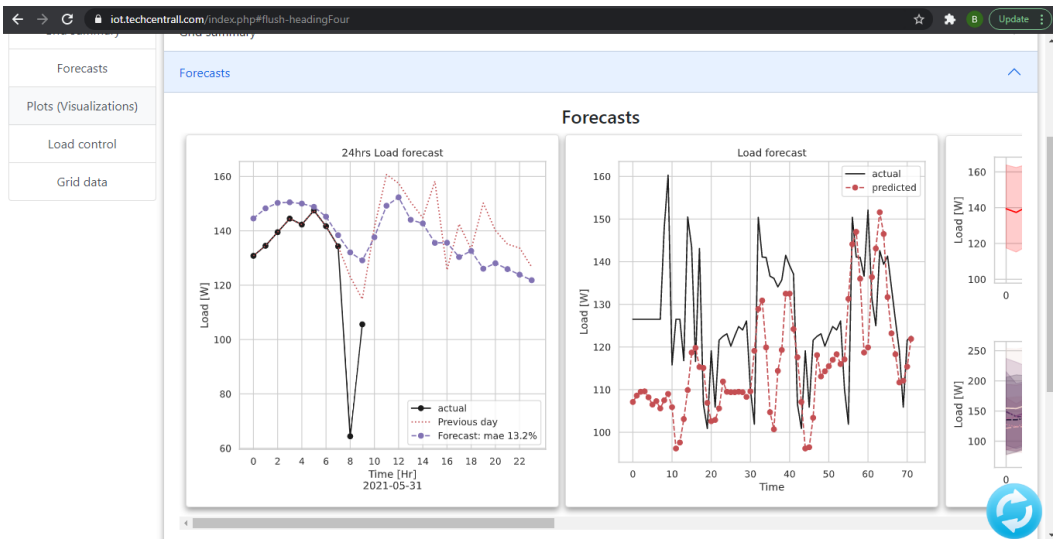


Figure 32: Energy dashboard interface showing consumption forecast and historical load charts

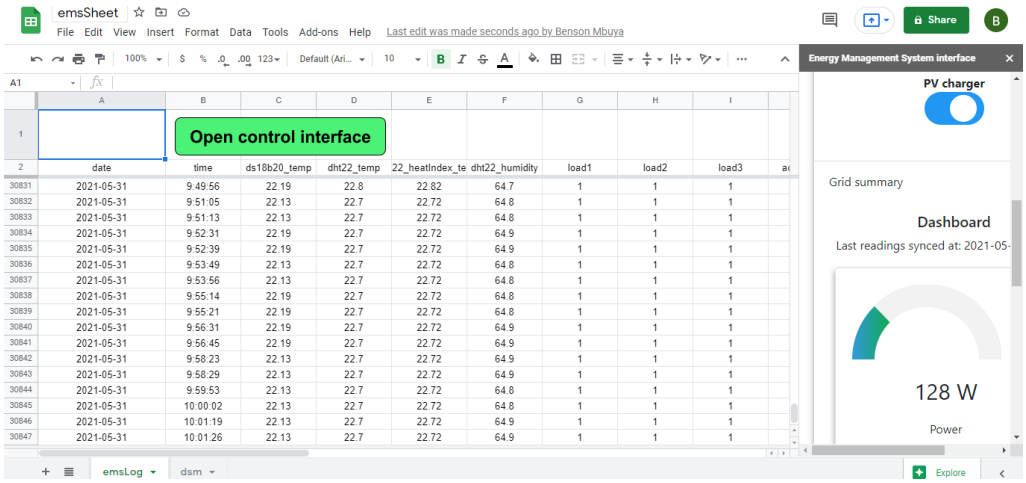


Figure 33: Low-cost EMS energy dashboard implementation via Google spreadsheet

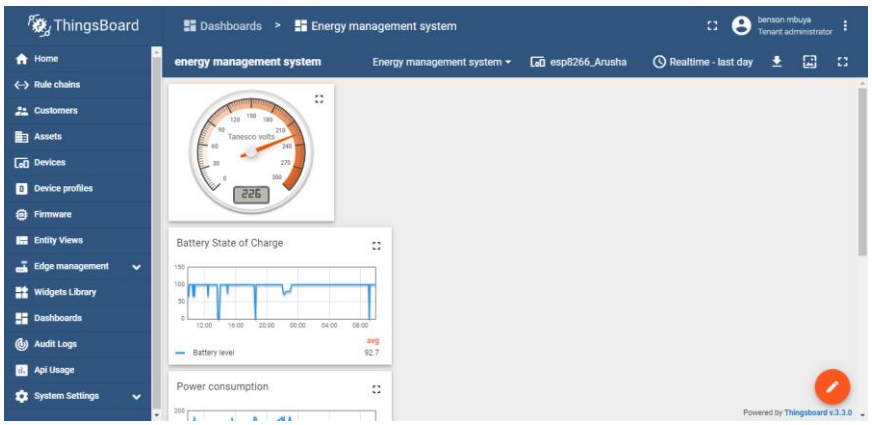


Figure 34: EMS dashboard interface implementation using ThingsBoard community version account

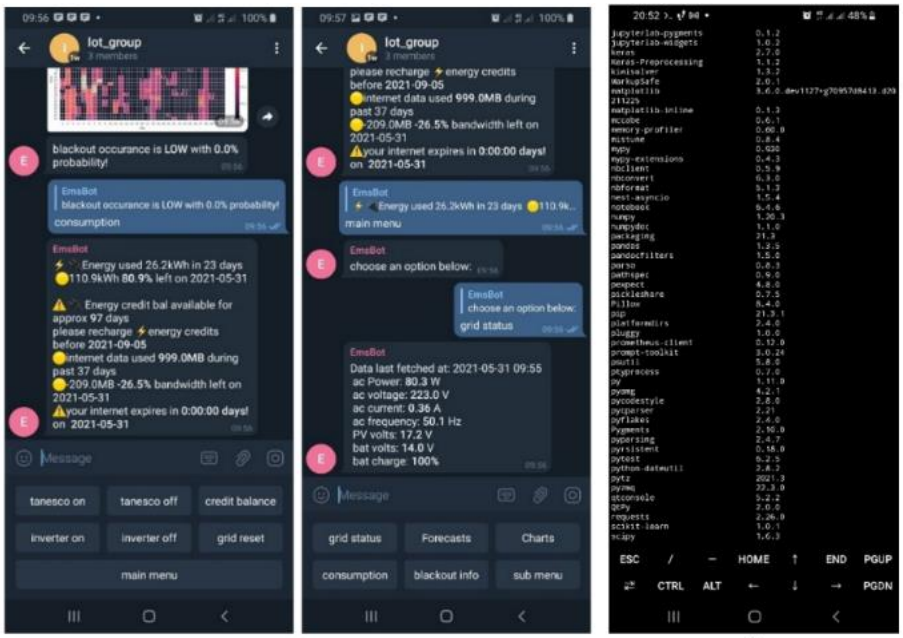


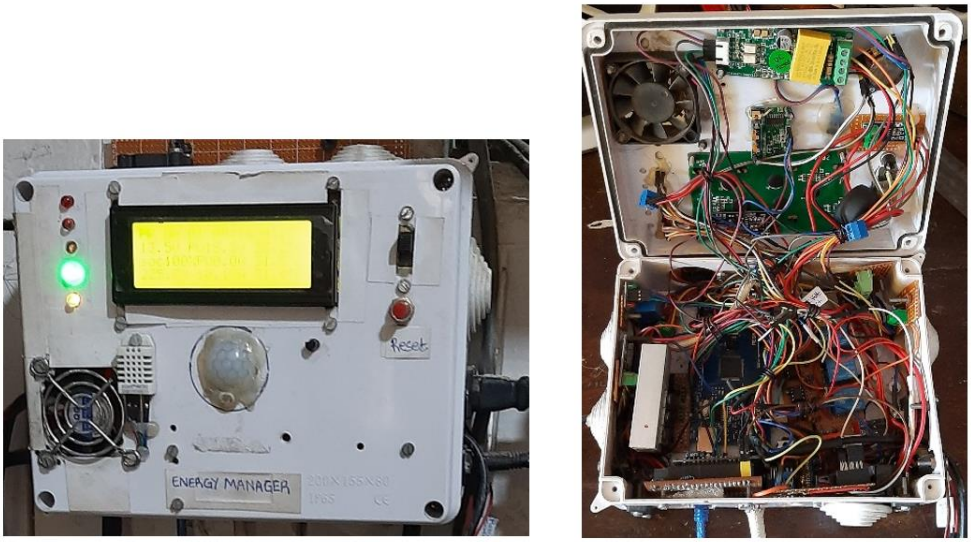
Figure 35: EMS user engagement via mobile phones. (a) Telegram app offers developers with API which could be tailored for energy monitoring activities. (b) Termux mobile phone terminal emulator can be adopted for mobile phone machine learning EMS forecasts

Free versions (community versions) of IoT platforms the likes of ThingsBoard have limited functionality – in this case study, it was only possible to monitor the grid without remote control functionality unless paid version was unlocked. Therefore, compromises have to be made when deploying microgrids with such IoT platforms; they still remain attractive because of their ease of use with a rapid learning curve. On the other hand, developing an energy dashboard for user engagement via a web portal on a shared website requires relatively more effort to build the website, but it has more advantages of customization and the freedom to add desired features, unlike IoT platforms which limits the subscriber only to the features available – in store. On the plus side, machine learning which is critical for EMS load and blackout forecast can also be performed on the same web server, eliminating the need to rent ML processing power services from elsewhere. Apps like Google spreadsheet, Google Drive, and Google Apps Scripts also have the potential of being used in low-cost microgrid energy dashboards however, if deployed in a project they may suffer from any policy changes rolled out by Google, thus, affecting the implementation already in place in the field. Their merit includes offering world-class security and authentication features.

The number of social media users in emerging countries is substantial, especially since mid-range and low-end smartphones tend to support social media apps; what better way to engage a user than using mobile phone Apps? Unlike WhatsApp which is monetized, Telegram offers a free API that allows developers to link Telegram chatbot with their IoT applications, thus, being the most cost-effective EMS option than the aforementioned approaches; the only caveat

being that the user requires to have active internet connectivity and the only cost being that of internet connection. For the case of emerging countries, if the site for EMS deployment has poor or expensive internet connectivity then an alternative solution would be using GSM modules (like SIM800L or SIM900L modules) to notify the user or perform remote control operations, however, graphical visualization charts are not possible with this approach since it relies on SMS.

Figure 36 shows the low-cost EMS prototype hardware implemented in this work at Levolosi ward, Arusha – Tanzania. It was implemented with open-source Arduino Mega and NodeMCU microcontrollers which are relatively easier to program and obtain. They have wide online community supporters and contributors making them ideal for emerging countries' EMS applications. As per Table 8, the overall EMS prototype hardware cost was around 100 USD being more economical than commercial EMS/SCADA systems which may cost up to thousands of US dollars. Figure 37, shows the low-cost SG3525-based inverter module used in this work; a NodeMCU unit was added to the inverter for IoT cloud connectivity; a set of relays linked to the NodeMCU forms a remote controllable change-over circuit which allows the selection of either utility grid power or inverter power supply.



(a) (b)

Figure 36: Developed low-cost EMS smart meter prototype hardware, (a) device exterior (b) device interior

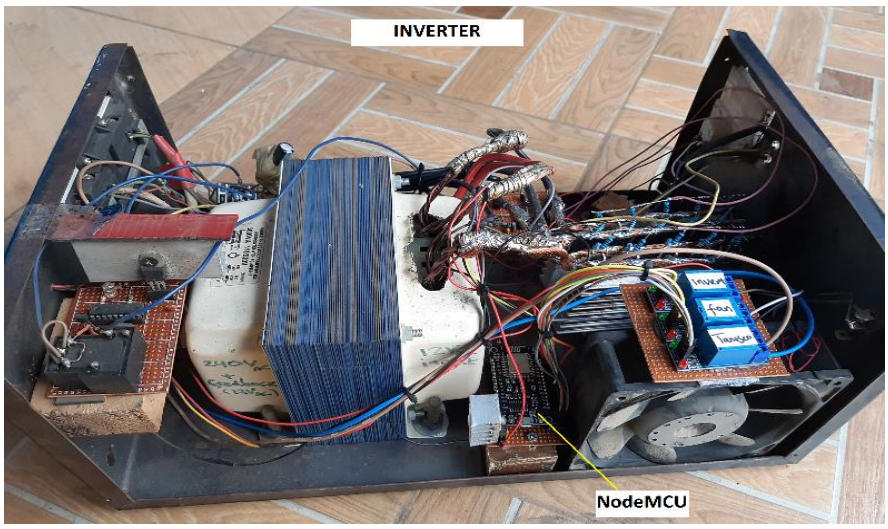


Figure 37: SG3525 based solar modified sine wave prototype inverter interfaced with NodeMCU for IoT operations

Table 8: Developed EMS smart meter bill of material

S/N	Component	Quantity	Cost (USD)
1	Arduino Mega	1	14.00
2	NodeMCU	2	8.00
3	DHT22	1	3.00
4	DS18B20	1	1.50
5	PIR sensor	1	1.50
6	Reed switch	1	1.50
7	LCD display	1	5.00
8	ACS712 current sensor	2	1.50
9	PZEM-004T energy meter	1	5.00
10	SD memory card module	1	1.00
11	Miscellaneous (casing, resistor, wires etc.)		58.00
	TOTAL		**100.00**

Karthick *et al.* (2021), proposed a smart meter for energy monitoring and control with DSM that also tracked power quality issues. Similar to this work, they used low-cost IoT devices like ESP8266 and ESP32 as WiFi modules, Relays for switching loads, a Raspberry pi 4B+ for local storage, and cloud linkage with Blynk App. The Blynk App is a commercial IoT platform that simplifies remote monitoring and control, but it may add to the cost of EMS implementation since it's a paid App. Their study did not incorporate any machine learning, although the Raspberry pi module is ML capable, it was underused in their study. This study did not use Raspberry pi since it is relatively more expensive than popular IoT microcontrollers like ESP8266 and ESP32. This study capitalized on the fact that most deployment sites have

users already with smartphones that could be used as internet hotspot modem, machine learning, and user interaction. Thus, bringing down the total cost of EMS implementation in emerging countries. The smart SCADA system implementation by researchers Aghenta and Iqbal (2019), was similar to that of Karthick *et al.* (2021), save for using the ThingsBoard IoT platform instead of Blynk App. Free versions of Apps like Blynk, and ThingBoard have many limitations, with useful features unlocked only for paying subscribers. Though this study explored the free version of ThingsBoard as an energy dashboard, it fell short of providing machine learning and remote control of loads.

In the study by Pawar *et al.* (2020), an IoT-based EMS was proposed with DSM load scheduling strategy and PV power generation prediction. Their PV power prediction model used PSO and SVM regression for hourly and day-ahead forecasts. They used solar irradiation data of the site from NREL (National Renewable Energy Laboratory) instead of locally measured data. The Xbee IoT modules were used and consumption data was stored on a local server. Their proposed EMS omitted load forecast and battery monitoring. Researchers Hijawi *et al.* (2020), implemented a lightweight and cost-effective EMS. They also considered environment sensing units for temperature, humidity, light, and occupancy detection. However, their work neither implemented any DSM control strategies nor load forecasting and energy storage management. In a study by Chen *et al.* (2014), the authors experimented to engage residents of a dense campus building via a web-based energy dashboard and email interactions/reminders. They observed that 90% of energy dashboard interactions were due to half of the participants and that participants interacted with the energy dashboard at mid-day. Email notifications increased energy conservation effectiveness and user engagement. This observation tallies with the hypothesis of this dissertation, that EMS user engagement in emerging countries can be achieved through smartphones web Apps, and Social Apps (especially instant messaging Apps). The study by Chen *et al.* (2014), achieved energy conservation simply through reporting to the end-users their real-time energy usage, arguably this has a limitation, it is far better to combine energy use reporting with remote control, forecasting, and suggestions (on load shedding or shifting) to the end-user.

4.4 Results Highlights

Input data feature selection methods have a substantial impact on the load forecast model's performance. Reducing the number of input features had a positive effect of reducing computation time and boosting the model's performance. Established feature selection methods can be blended by averaging to produce a subset pool of input features more effectively in prediction. Using many input features does not translate into higher prediction

efficacy, on the contrary, results show that only a few important input features should be used in the model. The STLF model aggregation results in a robust more accurate model than the stand-alone individual models constituting the hybrid model. In general, SVM-based models were found to perform better than decision-trees-based models in load forecast tasks. If there is no limitation on available computational resources and highest degree of prediction performance is desired, then it is recommended to use refined exhaustive principal k-features selection approach and hybrid model formed from element-wise min of two regression models selected from a superset of all the available LF models.

Short-term blackout forecast can either be solved as a classification challenge or a regression challenge. A hybrid Adaptive Similar Day (ASD) and Random Forest (RF) model for short-term power outage prediction has been proposed in this work. Accuracy score was found to be a less appropriate metric for blackout forecast since data was imbalanced with more records of power ON (no power cut) than instances of a power outage. Therefore, a better metric was found to be the F1score. The random nature of blackouts caused the models to struggle to predict power outages and resulted in only modest performance. The models developed found it easier to forecast blackouts when they were frequent than when they were sparse. Results indicate more work is needed to improve the blackout forecasting accuracy perhaps by using more exogenous inputs such as poles quantity, transformers, switches, distribution line length, served clients quantity, wind speed, rainfall, and tree trimming variable. Literature shows that, there is correlation between exogenous inputs and electric energy forecast (Mir *et al.,* 2020; Nateghi *et al.,* 2014c; Vivas *et al.,* 2020).

Low-cost IoT components were used to implement a practical EMS smart meter with user interaction using readily available open-source components such as Arduino boards and sensors. The smart meter developed, successfully exploited the use of social media apps such as the Telegram app platform for remote energy monitoring and control. Other energy dashboard platforms exploited were: A web page; a combination of cloud services (Google spreadsheet, Drive cloud storage, Google Apps script); and the Things Board IoT platform as an energy dashboard. Furthermore, the developed smart meter incorporated artificial intelligence with load forecast and blackout forecast functionality which can run on smartphone apps such as Termux. The goal of the smart meter linked to the energy dashboard is to engage the end user, shape energy use for the better, and increase energy efficiency by curbing energy wastefulness.

Although there are various web services, applications, and platforms which can be used in implementation of energy dashboard, the main technical parameter to be considered include

availability of API compatible with smart meter microcontroller, speed of data transmission, and ability to provide interactive visualization charts or widgets to the end user. The examined remote monitoring services are equally accurate, however, the main difference was in speed, aesthetic, and number of free remote operations available as already discussed in Section 4.3. In the end, a trade-off has to be made in choosing the proposed solutions by taking into account the specific needs of the deployment site.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

This study developed machine-learning based EMS with electric energy forecasting and monitoring for use in microgrids. In the first objective, machine learning techniques suitable for EMS in hybrid microgrids have been identified and employed to achieve objectives two and three. In the second objective, a short-term load forecasting model framework has been proposed. It has been shown that the blending of conventional feature selection methods gives more reliable global subset principal features that improve prediction model performance. Furthermore, blending two conventional regression models forms hybrid regression models with improved prediction performance in agreement with other works in the literature. There was a 43.6% error reduction from ‘finetree’ decision tree model to 'linearSVM’ model. An additional improvement was obtained when the overall best prediction model ‘linearSVM’ and ‘cubicSVM’ were combined to form a hybrid model resulting in a 5.4% MAE reduction. Three approaches were proposed for the supervised selection of a subset of principal k-features from a superset of N features through 1) mean score; 2) subset union; 3) refined exhaustive search based on k-combination. For computational time reasons, the mean score and subset union approach can be applied and the best of the two chosen after evaluating their performance on prediction models. If further prediction performance is desired, then a refined exhaustive principal k-features search can be applied although it is more resource intensive. Ultimately, the choice of which path to follow in the proposed framework depends on the computational resources available as well as the degree of prediction performance desired.

In the third objective, a short-term blackout forecasting model framework has been proposed. Generally, the regression and classification algorithms considered in this work namely: RF, ASD, and RF-ASD had about the same performance in blackout prediction. RF-ASD predicted accurately almost half of the blackouts (49.16%), thereby performing slightly better than the stand-alone RF (32.23%), and ASD (46.57%) models. The models developed were only able to predict blackouts if they occurred frequently and contiguously but performed poorly if they were sparse or dispersed. This is because the algorithm heavily relies its prediction on recent historical data, hence, if there no recent blackouts, it assumes there will continue to be no blackout in the short-term horizon. Including exogenous inputs correlated to power outage occurrence such as distribution line faults, utility lines maintenance work, utility lines tree trimming schedule, weather forecast (including rainfall) could potentially improve blackout

forecast even for dispersed power outage incidents. The developed models merely make an educated guess on the possible occurrence of a blackout, but not the precise time of the outage incidence. Overall, the blackout regression and classification models investigated in this work had fair performance in power outage prediction challenges along the test data considered months.

In the fourth objective, A low-cost EMS smart meter has been developed and implemented in this work as a tool for load control, battery management, user engagement, and energy awareness. The combination of the smart meter and energy dashboard technologies explored was found to be practical for low-income countries where most users are oblivious to their energy expenditure and footprint. A smart meter is a viable option for increasing energy efficiency and in the long run can be serviced easily with affordable components, in case of any breakdown that may arise, since the parts used are based on open-source projects with substantial online support material, forums, and communities. Most studies usually focus on only one entity of EMS – either load forecast, blackout forecast, or monitoring and control – in undertaking the third objective, all the aforementioned pillars of EMS were implemented and harmonised in one synergetic system. Thus, attesting to the uniqueness and practicality of the proposed EMS for emerging countries implementation, although, improvements are in order – as given in the recommendations section that follows.

Furthermore, in emerging countries scenario, a smartphone should be part and parcel of the EMS implementation due to increased smart phones penetration and use. It makes sense because smartphones are central in people's lives making it potentially easier to shape energy usage habits for the better – especially with smartphones at people's fingertips and their versatility. Their role in EMS proposed in this work is many folds: from providing physical interactions with the microgrid through energy dashboard (GUI); instant notifications; to being an internet hotspot for cloud connectivity and remote control – and if it has 2 or 3 SIM cards reliability of remote-control increases – this is key for rural deployment; to being a cost-effective option of running basic machine learning models for use in EMS.

All things considered; this study has attempted to answer the three research questions laid out at the outset. It is safe to conclude that, it's difficult to obtain an optimal electric energy forecast model for either load forecast and blackout forecast challenges since: the performance of most machine learning models is dependent on the platform and hardware specifications from which they are trained on; the modern-day grid and lifestyle keeps evolving as new consumer products and technologies are adopted, thus influencing how energy is produced and consumed; as long

as the issue of climate change remains unresolved, the weather will keep being difficult to forecast and so will load forecast and blackout forecast – due to the correlation between weather and electric energy forecast (LF and BF). In deploying EMS hardware and software (energy dashboard), it is better to opt for open-source material with substantial online support community and forum. A user-friendly manual and a service manual should be provided for the long-term sustainability of the project.

5.2 Recommendations

Other bio-inspired FS methods and prediction models could be explored to study their performance in load forecasting and blackout forecasting. It is worth investigating efficacy of blending and stacking multiple algorithms in load and blackout forecasting.

When monitoring weak grids, such as is the case in emerging countries, special attention should be given to the data logging system which should have multiple data storage contingency options to prevent any loss of data that leads to datasets with 'holes' or missing values. Missing data equals missed opportunities and insights. This study used a simple approach of estimating the remaining internet bundle credit of the internet modem used based on the number of data uploaded and received at the server side. A better approach, that prevents network loss – albeit, data loss due to finished internet credit would be to direct poll and scan SMS for the actual remaining subscribed internet credit balance from the Internet Service Provider (ISP) or mobile network operator.

Blackout prediction is a challenging task because it is caused by many factors such as weather, various faults in the grid, and their complex interactions. This work has endeavoured to predict blackouts only from the customer's point of connection to the grid without having other information about the grid as a whole which could be experiencing disturbances that may result in a blackout. If the blackout prediction model in this work is supplied with more data from the local electricity utility distributor (TANESCO) about the status of the grid at large, this would give the model a better vantage point in predicting imminent blackouts and increase performance. In addition, the load forecast and blackout forecast models developed have the potential to improve demand-side battery management. Prediction of blackouts can help battery management systems (BMS) determine whether to go into conservation mode whereby the smart BMS operates the battery storage near full charge. This prepares the system to cope with any sudden power outage, however, if the probability of blackout is very low, as predicted by the model, then the system can go into a relaxed mode where the battery state of charge level is allowed to discharge to low levels.

The low-cost EMS smart meter implemented in this work as an extension of the load and blackout forecast functions of EMS is reliant on internet connectivity. This may be unsuitable in areas with a poor internet connection, other internet-independent technologies should be studied for example long-range communication devices like LoRa, Zigbee, and so on. Grids studied in this work are small scale, there is the potential of gaining more insights from large grids as well as larger datasets. There is potentially more insight to be gained if blackout studies are performed on multiple communities within the same city but distant, in order to study cascading blackouts and rolling blackout predictions; specifically, how the onset of a blackout in one community may predict imminent blackout in another nearby in a chain reaction.

User engagement of EMS can be investigated when applied to multiple trending social media apps to determine the viability of such platforms in increasing energy efficiency in emerging countries scenarios. Conducting such a study on a dense community such as a campus could potentially be a good litmus test of the efficacy of the approach. Currently, users are not able to input energy tokens in TANESCO prepaid meters remotely, this function was not implemented in the proposed smart energy meter in this study, however, future studies could implement this feature.

REFERENCES

Abdullah, S., & Markandya, A. (2012). Rural electrification programmes in Kenya: Policy conclusions from a valuation study. *Energy for Sustainable Development*, *16*(1), 103–110. https://doi.org/10.1016/j.esd.2011.10.007

Abera, F. Z., & Khedkar, V. (2020). Machine Learning Approach Electric Appliance Consumption and Peak Demand Forecasting of Residential Customers Using Smart Meter Data. *Wireless Personal Communications*, *111*(1), 65–82.

Abubakar Mas'Ud, A., Wirba, A. V., Muhammad-Sukki, F., Albarracín, R., Abu-Bakar, S. H., Munir, A. B., & Bani, N. A. (2016). A review on the recent progress made on solar photovoltaic in selected countries of sub-Saharan Africa. *Renewable and Sustainable Energy Reviews*, 62, 441–452.

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2, 100006.

Afzalan, M., & Jazizadeh, F. (2019). Data-driven identification of consumers with deferrable loads for demand response programs. *Embedded Systems Letters*, 12(2), 54-57.

Agana, N. A., Oleka, E., Awogbami, G., & Homaifar, A. (2018). *Short-Term Load Forecasting based on a Hybrid Deep Learning Model*. https://scholar.google.com

Aghenta, L. O., & Iqbal, T. (2019). Design and implementation of a low-cost, open source IoT-based SCADA system using ESP32 with OLED, ThingsBoard and MQTT protocol. *Electronics and Electrical Engineering*, *4*(1), 57–86.

Aguilar Madrid, E., & Antonio, N. (2021). Short-term electricity load forecasting with machine learning. *Information*, 12(2), 1-21.

Aidoo, K., & Briggs, R. C. (2019). Underpowered: Rolling blackouts in Africa disproportionately hurt the poor. *African Studies Review*, 62(3), 112–131.

Ainah, P. K., & Folly, K. A. (2015). Development of micro-grid in Sub-Saharan Africa: An overview. *International Review of Electrical Engineering*, 10(5), 633–645.

Akhil-Srinivas, T. V., Subash, A., & Amutha, A. L. (2021). Load Forecasting Using Deep Learning. *Proceedings of the 5th International Conference on Electronics, Communication and Aerospace Technology*. https://scholar.google.com

Al Amin, M. A., & Hoque, M. A. (2019). *Comparison of ARIMA and SVM for Short-Term Load Forecasting. In 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (pp. 1-6).* https:// scholar. google. com

Alahmed, A. S., & Al-Muhaini, M. M. (2020). An intelligent load priority list–based integrated energy management system in microgrids. *Electric Power Systems Research*, 185, 106404.

Alkar, K. M., Meto, M., Jamjum, M., & Amar, K. (2019). *Performance Evaluation of the Blackout and Power Outages in Libyan Power Grid-Al-zawia Combined Cycle Power Plant Case Study. In 2019 1st International Conference on Sustainable Renewable Energy Systems and Applications (ICSRESA) (pp. 1-6).* https://scholar.google.com

Alkhathami, M. (2015). Introduction to electric load forecasting methods. *Journal of Advanced Electrical and Computer Engineering*, 2(1), 1-12.

Anoop, K. J., & Kanchana, K. (2017). *Short Term Load Forecasting using Fuzzy Logic Control. In 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI) (pp. 2983-2986).* https://scholar.google.com

Anwar, T., Sharma, B., Chakraborty, K., & Sirohia, H. (2018). Introduction to Load Forecasting. *International Journal of Pure and Applied Mathematics*, *119*, 1527–1538.

Aslam, S., Herodotou, H., Mohsin, S. M., Javaid, N., Ashraf, N., & Aslam, S. (2021). A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids. *Renewable and Sustainable Energy Reviews*, *144*, 110992.

Baharudin, Z., & Kamel, N. (2008, December). *Autoregressive Method in Short Term Load Forecast. In 2008 IEEE 2nd International Power and Energy Conference (pp. 1603-1608).* https://scholar.google.com

Balatsky, A. V., Balatsky, G. I., & Borysov, S. S. (2015). Resource demand growth and sustainability due to increased world consumption. *Sustainability*, *7*(3), 3430–3440. https://doi.org/10.3390/su7033430

Batista, N. C., Melício, R., Matias, J. C. O., & Catalão, J. P. S. (2013). Photovoltaic and wind energy systems monitoring and building/home energy management using ZigBee devices within a smart grid. *Energy*, *49*(1), 306–315.

Benyezza, H., Bouhedda, M., & Rebouh, S. (2021). Zoning irrigation smart system based on fuzzy control technology and IoT for water and energy saving. *Journal of Cleaner Production*, *302*, 127001. https://doi.org/10.1016/J.JCLEPRO.2021.127001

Bildirici, M., & Özaksoy, F. (2016). Woody Biomass Energy Consumption and Economic Growth in Sub-Saharan Africa. *Procedia Economics and Finance*, *38*(October 2015), 287–293. https://doi.org/10.1016/S2212-5671(16)30202-7

Blynk. (n.d.). *Blynk IoT platform: for businesses and developers. https://blynk.io*

Bo, Z., Shaojie, O., Jianhua, Z., Hui, S., Geng, W., & Ming, Z. (2015). An analysis of previous blackouts in the world: Lessons for China′ s power industry. *Renewable and Sustainable Energy Reviews*, *42*, 1151–1163.

Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. (2018). Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*, 11(7), 1636.

Bourdeau, M., qiang Zhai, X., Nefzaoui, E., Guo, X., & Chatellier, P. (2019). Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society*, 48, 101533.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L. (2017). *Classification and Regression Trees.* Routledge. https:// scholar. google. com

Carmeli, M. S., Guidetti, P., Mancini, M., Mandelli, S., Mauri, M., Merlo, M., Perini, R., Tomasini, G., Marchegiani, G., & Rosati, D. (2014). *Hybrid Distributed Generation System for a Rural Village in Africa.* https://scholar.google.com

Carmeli, M. S., Mauri, M., Brivio, C., Guidetti, P., Mandelli, S., Merlo, M., Perini, R., Marchegiani, G., & Rosati, D. (2015). *Hybrid micro-grid experimental application in Tanzania. In 2015 International Conference on Clean Electrical Power (ICCEP) (pp. 534-541).* https://scholar.google.com

Carmeli, M. S., Mauri, M., Brivio, C., Guidetti, P., Mandelli, S., Merlo, M., Perini, R., Milano, P., Meccanica, D., Masa, V. La, Milano, P., Energia, D., & Lambruschini, V. (2015). *Hybrid Micro-Grid Experimental Application in Tanzania.* https://scholar.google.com

Chandramowli, S. N., & Felder, F. A. (2014). Impact of climate change on electricity systems and markets: A review of models and forecasts. *Sustainable Energy Technologies and Assessments*, *5*, 62–74.

Chang, Y. P. (2014). Design and implementation of intelligent composite battery charger for solar energy lighting systems. *Journal of Technology*, *29*(1), 79–92.

Chaturvedi, D. K. (2008). Applications of genetic algorithms to load forecasting problem. *Studies in Computational Intelligence*, *103*, 383–402. https://doi.org/10.1007/978-3-540-77481-5_10/COVER

Chen, H., Liu, S., Liu, Q., Shi, X., Wei, W., Han, R., & Küfeoğlu, S. (2021). Estimating the impacts of climate change on electricity supply infrastructure: A case study of China. *Energy Policy*, *150*, 112119.

Chen, V. L., Delmas, M. A., & Kaiser, W. J. (2014). Real-time, appliance-level electricity use feedback system: How to engage users? *Energy and Buildings*, *70*, 455–462. https://doi.org/10.1016/J.ENBUILD.2013.11.069

Chen, Y. T., Piedad Jr, E., & Kuo, C. C. (2019). Energy consumption load forecasting using a level-based random forest classifier. *Symmetry*, 11(8), 956.

Cheng, L., Liu, M., Wang, D., Wu, D., Xu, Z., & Su, Y. (2017). *The Outage Rate Model based on the Improved Evidence Theory Considering Various Outage Factors. In 2017 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC) (pp. 1-6).* https://scholar.google.com

Cheng, Q., Yao, J., Wu, H., Chen, S., Liu, C., & Yao, P. (2013). *Short-Term Load Forecasting with Weather Component based on Improved Extreme Learning Machine. In 2013 Chinese Automation Congress (pp. 316-321)*. https://scholar.google.com

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273-297.

Dai, Y., & Zhao, P. (2020). A hybrid load forecasting model based on support vector machine with intelligent methods for feature selection and parameter optimization. *Applied Energy*, *279*, 115332. https://doi.org/10.1016/J.apenergy.2020.115332

De Oliveira, M., Araujo, N., Da Silva, R., Da Silva, T., & Epaarachchi, J. (2018). Use of savitzky--golay filter for performances improvement of SHM systems based on neural networks and distributed PZT sensors. *Sensors*, *18*(1), 152.

Demir, E., Köseoğlu, E., Sokullu, R., & Şeker, B. (2017). Smart Home Assistant for Ambient Assisted Living of Elderly People with Dementia. *Procedia Computer Science*, *113*, 609–614. https://doi.org/10.1016/J.PROCS.2017.08.302

Dhaval, B., & Deshpande, A. (2020). Short-term load forecasting with using multiple linear regression. *International Journal of Electrical and Computer Engineering*, 10(4), 3911-3917.

Dinh, H. T., Yun, J., Kim, D. M., Lee, K. H., & Kim, D. (2020). A Home Energy Management System with Renewable Energy and Energy Storage Utilizing Main Grid and Electricity Selling. *IEEE Access*, *8*, 49436–49450.

Dudek, G. (2015a). Pattern similarity-based methods for short-term load forecasting - Part 1. *Applied Soft Computing*, *37*, 277–287.

Dudek, G. (2015b). Pattern similarity-based methods for short-term load forecasting – Part 2: Models. *Applied Soft Computing*, *36*, 422–441.

Dudek, G. (2016). Pattern-based local linear regression models for short-term load forecasting. *Electric Power Systems Research*, *130*, 139–147.

Dudek, G. (2022). A Comprehensive Study of Random Forest for Short-Term Load Forecasting. *Energies,* 15(20), 7547.

Dufo-López, R., Cortés-Arcos, T., Artal-Sevil, J. S., & Bernal-Agustín, J. L. (2021). Comparison of lead-acid and li-ion batteries lifetime prediction models in stand-alone photovoltaic systems. *Applied Sciences,* 11(3), 1099.

E4G. (2018). *Ngarenanyuki Microgrid Data*. PoliMi. www.e4g.polimi.it

EKO-energy. (n.d.). Home – EKO-energy. https://www.ekoenergy.org

Elahe, M. F., Jin, M., & Zeng, P. (2021). Review of load data analytics using deep learning in smart grids: Open load datasets, methodologies, and application challenges. *International Journal of Energy Research*, *45*(10), 14274–14305.

Eseye, A. T., Lehtonen, M., Tukia, T., Uimonen, S., & Millar, J. (2019). *Efficient Feature Selection Strategy for Accurate Electricity Demand Forecasting. In 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe) (pp. 1-5).* https://scholar.google.com

Eseye, A. T., Lehtonen, M., Tukia, T., Uimonen, S., & Millar, R. J. (2019). Machine learning based integrated feature selection approach for improved electricity demand forecasting in decentralized energy systems. *IEEE Access*, 7, 91463-91475

Falentina, A. T., & Resosudarmo, B. P. (2019). The impact of blackouts on the performance of micro and small enterprises: Evidence from Indonesia. *World Development*, *124*, 104635.

Feng, C., & Zhang, J. (2020). Assessment of aggregation strategies for machine-learning based short-term load forecasting. *Electric Power Systems Research*, *184*, 106304. https://doi.org/10.1016/J.EPSR.2020.106304

Ferdoush, S., & Li, X. (2014). Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. *Procedia Computer Science*, *34*, 103–110. https://doi.org/10.1016/J.PROCS.2014.07.059

Fuks, M., & Salazar, E. (2008). Applying models for ordinal logistic regression to the analysis of household electricity consumption classes in Rio de Janeiro, Brazil. *Energy Economics*, 30(4), 1672-1692.

Gasparin, A., Lukovic, S., & Alippi, C. (2022). Deep learning for time series forecasting: The electric load case. *Transactions on Intelligence Technology*, *7*(1), 1–25. https://doi.org/https://doi.org/10.1049/cit2.12060

GeP, B., Jenkins, G. M., & Reinsel, G. C. (2008). *Time Series Analysis: Forecast and Control. Hoboken*. New Jersey: John Wiley \& Sons.

Ghiasi, M., Ahmadinia, E., Lariche, M., Zarrabi, H., & Simoes, R. (2018). A new spinning reserve requirement prediction with hybrid model. *Smart Science*, 6(3), 212-221.

Gou, B., & Wu, W. (2008, July). *Is the Prediction of Power System Blackouts Possible? In 2008 IEEE Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century (pp. 1-4).* https://scholar.google.com.

Hammad, M. A., Jereb, B., Rosi, B., & Dragan, D. (2020). Methods and models for electric load forecasting: a comprehensive review. *Logistics, Supply Chain, Sustainability and Global Challenges*, 11(1), 51-76.

Hashmi, S. A., Ali, C. F., & Zafar, S. (2021). Internet of things and cloud computing-based energy management system for demand side management in smart grid. *International Journal of Energy Research*, *45*(1), 1007–1022.

Hijawi, U., Gastli, A., Hamila, R., Ellabban, O., & Unal, D. (2020). *Qatar Green Schools Initiative: Energy Management System with Cost-Efficient and Lightweight Networked IoT. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT) (pp. 415-421).* https://scholar.google.com

Hong, T., & Dickey, D. A. (2015). *Electric Load Forecasting: Fundamentals and Best Practices.* https://www.otexts.org/elf

Hong, T., & Fan, S. (2016). Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, *32*(3), 914–938.

Hu, Z., Bao, Y., Xiong, T., & Chiong, R. (2015). Hybrid filter–wrapper feature selection for short-term load forecasting. *Engineering Applications of Artificial Intelligence*, *40*, 17–27. https://doi.org/10.1016/j.engappai.2014.12.014

Huang, G. Bin, Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *42*(2), 513–529.

Huang, W., Zhang, D., Song, Z., Wang, H., & Liu, H. (2019). *Short-Term Load Forecasting based on Similar Day Approach and Intelligent Algorithm using Analytic Hierarchy Process. Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference. https://scholar.google.com*

IEA. (2014). Africa Energy Outlook. A focus on the energy prospects in sub-Saharan Africa. *World Energy Outlook Special Report, International Energy Agency Publication*, 2014, 122–130.

International Energy Agency. (2014). *Africa Energy Outlook*. www.iea.org

International Energy Agency. (2019). *Africa Energy Outlook*. https:// doi. org/ 10. 1787/ g2120ab250-en

Iqbal, T., & Manzoor, S. (2020). *IoT based Renewable Energy Management and Monitoring System for the First Passive House in Newfoundland*. https://scholar.google.com

Islam, B., Baharudin, Z., Raza, Q., & Nallagownden, P. (2014). A hybrid neural network and genetic algorithm based model for short term load forecast. *Research Journal of Applied Sciences, Engineering and Technology*, *7*(13), 2667–2673.

Ji, P., Xiong, D., Wang, P., & Chen, J. (2012). *A study on Exponential Smoothing Model for Load Forecasting. Asia-Pacific Power and Energy Engineering Conference. https://scholar.google.com*

Jović, A., Brkić, K., & Bogunović, N. (2015, May). A Review of Feature Selection Methods with Applications. *In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (pp. 1200-1205).* https://scholar.google.com

Kanakaraja, P., Agarwal, V., Sri Harsha, K., Jaswanth, N., & Preethi, P. (2021). Smart BMS pilot project using LoRa networks. *Materials Today: Proceedings*, *46*, 3893–3902.

Kaneko, H., Matsumoto, T., Ootakara, S., & Funatsu, K. (2016). Practical use of savitzky-golay filtering-based ensemble online svr. *IFAC-PapersOnLine,* 49(7), 371-376.

Kankanala, P., Das, S., & Pahwa, A. (2014). Adaboost+: An ensemble learning approach for estimating weather-related outages in distribution systems. *Transactions on Power Systems*, *29*(1), 359–367. https://doi.org/10.1109/TPWRS.2013.2281137

Kapoor, A., & Sharma, A. (2018). *A Comparison of Short-Term Load Forecasting Techniques.* https://scholar.google.com

Karami, M., McMorrow, G. V., & Wang, L. (2018). Continuous monitoring of indoor environmental quality using an Arduino-based data acquisition system. *Journal of Building Engineering*, *19*, 412–419. https://doi.org/10.1016/j.jobe.2018.05.014

Karthick, T., Charles Raja, S., Jeslin Drusila Nesamalar, J., & Chandrasekaran, K. (2021). Design of IoT based smart compact energy meter for monitoring and controlling the usage of energy and power quality issues with demand side management for a commercial building. *Sustainable Energy, Grids and Networks*, *26*, 100454.

Kern, D., Ensinger, A., Hammer, C., Neufeld, C., Lecon, C., Nagl, A., Bozem, K., Harrison, D. K., & Wood, B. M. (2022). *Application Possibilities of Artificial Intelligence in a Renewable Energy Platform.* https://scholar.google.com

Keszocze, O., Soeken, M., & Drechsler, R. (2018). The complexity of error metrics. *Information Processing Letters*, 139, 1-7.

Khan, A. R., Mahmood, A., Safdar, A., Khan, Z. A., & Khan, N. A. (2016). Load forecasting, dynamic pricing and DSM in smart grid: A review. *Renewable and Sustainable Energy Reviews*, *54*, 1311–1322. https://doi.org/10.1016/j.rser.2015.10.117

Khan, R. A., Dewangan, C. L., Srivastava, S. C., & Chakrabarti, S. (2018). *Short Term Load Forecasting using SVM Models. The 8th IEEE Power India International Conference. https://scholar.google.com*

Kim, W., Han, Y., Kim, K. J., & Song, K. W. (2020). Electricity load forecasting using advanced feature selection and optimal deep learning model for the variable refrigerant flow systems. *Energy Reports*, *6*, 2604–2618.

Kogo, T., Nakamura, S., Pravinraj, S., & Arumugam, B. (2014). *A Demand Side Prediction Method for Persistent Scheduled Power-Cuts in Developing Countries*. https://scholar.google.com

Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., & Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, *40*, 100364.

Kunqiao, Y., & Jiandong, J. (2021). Short-term load forecasting based on ELM combined model. *Proceedings - 2021 International Conference on Computer, Blockchain and Financial Development, CBFD 2021*, 1–6.

Kyi, S., & Taparugssanagorn, A. (2020). Wireless sensing for a solar power system. *Digital Communications and Networks*, *6*(1), 51–57.

Laghari, J. A., Almani, S. A., Kumar, J., Mokhlis, H., & Bakar, A. H. A. (2018). A smart under-frequency load shedding scheme based on Takagi-Sugeno fuzzy inference system and flexible load priority. *International Journal of Advanced Computer Science and Applications*, *9*(3), 1-7. https://doi.org/10.14569/IJACSA.2018.090319

Lahouar, A., & Slama, J. B. H. (2015). Day-ahead load forecast using random forest and expert input selection. *Energy Conversion and Management,* 103, 1040-1051.

Li, H., Treinish, L. A., & Hosking, J. R. (2010). A statistical model for risk management of electric outage forecasts. *IBM Journal of Research and Development*, 54(3), 8-1.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *Computing Surveys* (CSUR), 50(6), 1-45.

Li, Y., Su, Y., & Shu, L. (2014). An ARMAX model for forecasting the power output of a grid connected photovoltaic system. *Renewable Energy*, *66*, 78–89.

Liang, Z., Chengyuan, Z., Zhengang, Z., & Dacheng, Z. (2021, May). *Short-Term Load Forecasting based on Kalman Filter and Nonlinear Autoregressive Neural Network. In 2021 33rd Chinese Control and Decision Conference (CCDC) (pp. 3747-3751)*. https://scholar.google.com

Liu, H., Davidson, R. A., & Apanasovich, T. V. (2008). Spatial generalized linear mixed models of electric power outages due to hurricanes and ice storms. *Reliability Engineering & System Safety*, *93*(6), 897–912.

Liu, H., Davidson, R. A., Rosowsky, D. V., & Stedinger, J. R. (2005). Negative binomial regression of electric power outages in hurricanes. *Journal of Infrastructure Systems*, 11(4), 258-267.

Lopez-Martin, M., Sanchez-Esguevillas, A., Hernandez-Callejo, L., Arribas, J. I., & Carro, B. (2021). Novel data-driven models applied to short-term electric load forecasting. *Applied Sciences*, 11(12), 5708.

Makarov, Y. V, Reshetov, V. I., Stroev, A., & Voropai, I. (2005). Blackout prevention in the United States, Europe, and Russia. *Proceedings of the institute of Electrical and Electronics Engineers*, *93*(11), 1942–1955.

Mamun, A. Al, Sohel, Md., Mohammad, N., Haque Sunny, Md. S., Dipta, D. R., & Hossain, E. (2020). A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models. *Access*, *8*, 134911–134939.

Mandelli, S., Brivio, C., Moncecchi, M., Riva, F., Bonamini, G., & Merlo, M. (2017). Novel LoadProGen procedure for micro-grid design in emerging country scenarios: Application to energy storage sizing. *Energy Procedia*, *135*(2), 367–378.

Mandelli, S., Merlo, M., Tedeschi, E., & Molinas, M. (2015, March). *Electro-Mechanical Model for Understanding the Operation and Dynamic Behavior of a Micro-Grid: A case Study in Tanzania. In 2015 Tenth International Conference on Ecological Vehicles and Renewable Energies (EVER) (pp. 1-9).* https://scholar.google.com

Mandelli, S., Molinas, M., Park, E., Leonardi, M., Colombo, E., & Merlo, M. (2015). The Role of Storage in Emerging Country Scenarios. *Energy Procedia*, *73*, 112–123.

Martínez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., & Riquelme, J. (2015). A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting. *Energies*, *8*(12), 13162–13193. https://doi.org/10.3390/en81112361

Mauri, M., Carmeli, M. S., Merlo, M., Brivio, C., & Mbuya, B. (2016). *Neural Network based Load Forecasting and Fuzzy Logic EMS for Ngarenanyuki School microgrid. International Symposium on Power Electronics Electrical Drives Automation and Motion.* https://scholar.google.com

Mbanaso, U. M., Chukwudebe, G. A., & Atimati, E. E. (2015). A critical assessment of Nigeria's presence on the Cyberspace. *The 2015 International Conference on Cyberspace.* https://scholar.google.com

Mbuli, N., Mathonsi, M., Seitshiro, M., & Pretorius, J. H. C. (2020). Decomposition forecasting methods: A review of applications in power systems. *Energy Reports*, *6*. https://doi.org/10.1016/j.egyr.2020.11.238

Mei, S., He, F., Zhang, X., Wu, S., & Wang, G. (2009). An improved OPA model and blackout risk assessment. *IEEE Transactions on Power Systems*, *24*(2), 814–823.

Mir, A. A., Alghassab, M., Ullah, K., Khan, Z. A., Lu, Y., & Imran, M. (2020). A review of electricity demand forecasting in low and middle income countries: The demand determinants and horizons. *Sustainability*, 12(15), 5931.

Mogaji, E., Balakrishnan, J., Nwoba, A. C., & Nguyen, N. P. (2021). Emerging-market consumers' interactions with banking chatbots. *Telematics and Informatics, 65*, 101711. https://doi.org/10.1016/J.TELE.2021.101711

Moise, M. V., Niculescu, A.-M., & Dumitraşcu, A. (2020). *Integration of Internet of Things technology into a pill dispenser. The 2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging. https://scholar.google.com*

Moon, J., Jung, S., Rew, J., Rho, S., & Hwang, E. (2020). Combination of short-term load forecasting models based on a stacking ensemble approach. *Energy and Buildings, 216*, 109921. https://doi.org/10.1016/J.ENBUILD.2020.109921

Moradzadeh, A., Moayyed, H., Zakeri, S., Mohammadi-Ivatloo, B., & Aguiar, A. P. (2021). Deep learning-assisted short-term load forecasting for sustainable management of energy in microgrid. *Inventions*, 6(15), 1-11.

Muchunku, C., Ulsrud, K., Palit, D., & Jonker-Klunne, W. (2018). Diffusion of solar PV in East Africa: What can be learned from private sector delivery models? *WIREs Energy and Environment*, *7*(3), e282.

Mukhopadhyay, P., Mitra, G., Banerjee, S., & Mukherjee, G. (2018). Electricity load forecasting using fuzzy logic: Short term load forecasting factoring weather parameter. *2017 7th International Conference on Power Systems.* https://scholar.google.com

Nateghi, R., Guikema, S. D., & Quiring, S. M. (2014a). Forecasting hurricane-induced power outage durations. *Natural Hazards*, *74*(3), 1795–1811.

Nateghi, R., Guikema, S. D., & Quiring, S. M. (2014b). Forecasting hurricane-induced power outage durations. *Natural Hazards*, *74*(3), 1795–1811.

Nateghi, R., Guikema, S., & Quiring, S. M. (2014c). Power outage estimation for tropical cyclones: Improved accuracy with simpler models. *Risk Analysis*, *34*(6), 1069–1078.

National Bureau of Statistics. (2013). 2012 *Population and Housing Census. Population Distribution by Administrative Areas. Dar-es-Salaam: National Bureau of Statistics*. https://scholar.google.com

Nguyen, B., Hosseini, Z., & Gao, W. (2018). *Analysis of Pricing Trends and Grid Parity of Photovoltaic Systems 2018 Grid of the Future Symposium.* https://scholar.google.com

Niculescu, A. M., Pavel, D. M., Dumitrașcu, A., Elisei, N., & Moise, V. M. (2021). IoT module for air quality measurement. *The 2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging.* https://scholar.google.com

Nkosi, N. P., & Dikgang, J. (2018). Pricing electricity blackouts among South African households. *Journal of Commodity Markets, 11*, 37–47.

Nti, I. K., Teimeh, M., Nyarko-Boateng, O., & Adekoya, A. F. (2020). Electricity load forecasting: a systematic review. *Journal of Electrical Systems and Information Technology*, 7(1), 1-19.

Nyari, E. A., Castro-Sitiriche, M., & Park, S. E. (2017). An experimental study of electrical appliances consumption using panel meter data. *Journal of Power and Energy Engineering*, 5(9), 132-144.

Nystrup, P., Madsen, H., Blomgren, E. M., & de Zotti, G. (2021). Clustering commercial and industrial load patterns for long-term energy planning. *Smart Energy*, 2, 100010.

Ogunjuyigbe, A. S., Ayodele, T. R., Lasarus, C. P., Yusuff, A. A., & Mosetlhe, T. C. (2021). *Comparative Analysis of Short-Term Load Forecasting Methods.* https://scholar.google.com

Oprea, S. V., & Bara, A. (2019a). Machine Learning Algorithms for Short-Term Load Forecast in Residential Buildings Using Smart Meters, Sensors and Big Data Solutions. *Access*, *7*, 177874–177889.

Pai, P. F., & Hong, W. C. (2005). Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. *Electric Power Systems Research*, *74*(3), 417–425. https://doi.org/10.1016/j.epsr.2005.01.006

Papic, M., & Ciniglio, O. (2014). *Prediction and Prevention of Cascading Outages in Idaho Power Network. In 2014 IEEE PES General Meeting| Conference & Exposition (pp. 1-5).* https://scholar.google.com

Papic, M., Angell, D., Van Patten, P., & Efaw, B. (2018). *Historical Performance Analysis of Transmission Lines using Idaho Power Company's Outage Database GTORS. In 2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems.* https://scholar.google.com

Pawar, P., & Vittal K, P. (2019). Design and development of advanced smart energy management system integrated with IoT framework in smart grid environment. *Journal of Energy Storage*, *25*, 100846.

Pawar, P., TarunKumar, M., & Vittal K., P. (2020). An IoT based Intelligent Smart Energy Management System with accurate forecasting and load strategy for renewable generation. *Measurement*, *152*, 107187.

Perumal, V. S. A., Baskaran, K., & Rai, S. K. (2017). Implementation of effective and low-cost Building Monitoring System using raspberry PI. *Energy Procedia*, 143, 179-185.

Phyo, P. P., & Jeenanunta, C. (2022). Advanced ML-Based Ensemble and Deep Learning Models for Short-Term Load Forecasting: Comparative Analysis Using Feature Engineering. *Applied Sciences*, 12(10), 4882.

Pocero, L., Amaxilatis, D., Mylonas, G., & Chatzigiannakis, I. (2017). Open source IoT meter devices for smart and energy-efficient school buildings. *HardwareX*, 1, 54-67.

Podder, S., & Khan, M. Z. R. (2016, May*). Comparison of Lead Acid and Li-ion Battery in Solar Home System of Bangladesh. In 2016 5th International Conference on Informatics, Electronics and Vision (pp. 434-438).* https://scholar.google.com

Pollet, B. G., Staffell, I., & Adamson, K. A. (2015). Current energy landscape in the Republic of South Africa. *International Journal of Hydrogen Energy*, *40*(46), 16685–16701.

Pourdaryaei, A., Mohammadi, M., Karimi, M., Mokhlis, H., Illias, H. A., Kaboli, S. H. A., & Ahmad, S. (2021). Recent development in electricity price forecasting based on computational intelligence techniques in deregulated power market. *Energies*, 14(19), 6104.

Qin, Y. U., & Du, S. (1994*). A practical and Low Cost PWM Battery Charger using Fuzzy Logic Control for UPS application. In Proceedings of Intelec 94 (pp. 443-450)*. https:// scholar. google. com

Rahman, K. J., Munnee, M. M., & Khan, S. (2016). *Largest Blackouts Around the world: Trends and Data Analyses. In 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE) (pp. 155-159)*. https://scholar.google.com

Raspisaniye Pogodi Ltd. (n.d.). *Weather Archive in Kilimanjaro Airport*. https:// scholar. google. com

Ray, P. P. (2016). A survey of IoT cloud platforms. *Future Computing and Informatics Journal*, *1*(1–2), 35–46. https://doi.org/10.1016/J.FCIJ.2017.02.001

REA, & NBS. (2020). *Energy Access and Use Situation Survey II in Tanzania Mainland 2019/20*. http://www.rea.go.tz/Resources/E-Library/tabid/132/Default.aspx

Ribeiro, A. M. N., do Carmo, P. R. X., Endo, P. T., Rosati, P., & Lynn, T. (2022). Short-and very short-term firm-level load forecasting for warehouses: a comparison of machine learning and deep learning models. Energies, 15(3), 1-24.

Román-Portabales, A., López-Nores, M., & Pazos-Arias, J. J. (2021). Systematic review of electricity demand forecast using ANN-based machine learning algorithms. *Sensors,* 21(13), 4544.

Saxena, H., Aponte, O., & McConky, K. T. (2019). A hybrid machine learning model for forecasting a billing period's peak electric load days. *International Journal of Forecasting*, 35(4), 1288-1303.

Serna-Suárez, I. D., Ordóñez-Plata, G., & Carrillo-Caicedo, G. (2015, May). *Microgrid's Energy Management Systems: A survey. In 2015 12th International Conference on the European Energy Market (EEM) (pp. 1-6)*. https://scholar.google.com

Sforna, M., & Delfanti, M. (2006, October). *Overview of the Events and Causes of the 2003 Italian blackout. In 2006 IEEE PES Power Systems Conference and Exposition (pp. 301-308)*. https://scholar.google.com

Sharma, S., Majumdar, A., Elvira, V., & Chouzenoux, E. (2020). Blind Kalman Filtering for Short-Term Load Forecasting. *Transactions on Power Systems*, *35*(6), 4916–4919. https://doi.org/10.1109/TPWRS.2020.3018623

Sheng, F., & Jia, L. (2020). *Short-Term Load Forecasting based on SARIMAX-LSTM. In 2020 5th International Conference on Power and Renewable Energy (ICPRE) (pp. 90-94)*. https://scholar.google.com.

Shi, H., Xu, M., & Li, R. (2017). Deep Learning for Household Load Forecasting: A Novel Pooling Deep. *Transactions on Smart Grid*, *3053*(c), 1–1. https://doi.org/10.1109/TSG.2017.2686012

Sibiya, C. A., Numbi, B. P., & Kusakana, K. (2021). Performance analysis of an off-grid renewable energy hybrid powered CPU system with storage. *Energy Reports*, *7*, 43–52. https://doi.org/10.1016/J.EGYR.2021.05.062

Singla, P., Duhan, M., & Saroha, S. (2021). Review of different error metrics: A case of solar forecasting. *Journal of Science and Engineering*, 20(4), 158-165.

Singla, P., Duhan, M., & Saroha, S. (2021). Review of different error metrics: A case of solar forecasting. *Journal of Science and Engineering*, 20(4), 158-165.

Sisavath, C., & Yu, L. (2021). Design and implementation of security system for smart home based on IOT technology. *Procedia Computer Science*, *183*, 4–13.

Soliman, S., & Al-Kandari, A. M. (2010). *Electrical Load Forecasting*. https:// doi. org/ 10. 1016/ B978-0-12-381543-9.00001-4

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.

Su, P., Tian, X., Wang, Y., Deng, S., Zhao, J., An, Q., & Wang, Y. (2017). Recent Trends in Load Forecasting Technology for the Operation Optimization of Distributed Energy System. *Energies*, *10*(9), 1303. https://doi.org/10.3390/en10091303

Subbiah, S. S., & Chinnappan, J. (2020). An improved short term load forecasting with ranker based feature selection technique. *Journal of Intelligent & Fuzzy Systems*, *39(5), 6783-6800*.

Subbiah, S. S., & Chinnappan, J. (2022). Short-Term Load Forecasting Using Random Forest with Entropy-Based Feature Selection. *Lecture Notes in Electrical Engineering*, *806*, 73–80. https://doi.org/10.1007/978-981-16-6448-9_8/COVER

Suresan, A., Mohan, S. S., Arya, M. P., Anjana Gangadharan, V., & Bindu, P. V. (2021). A Conversational AI Chatbot in Energy Informatics. *Proceedings of International Conference on Intelligent Computing, Information and Control Systems*, 2021, 543–554.

Takeda, H., Tamura, Y., & Sato, S. (2016). Using the ensemble Kalman filter for electricity load forecasting and analysis. *Energy*, *104*, 184–198.

Takiyar, S., & Singh, V. (2015, September). Trend analysis and evolution of short term load forecasting techniques. In 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions) (pp. 1-6). IEEE.

The MathWorks, I. (n.d.). *Rank Importance of Predictors using ReliefF or RReliefF Algorithm - MATLAB relieff*. https://www.mathworks.com/help/stats/relieff.html

Tiwari, S., Sabzehgar, R., & Rasouli, M. (2019, February). *Short Term Solar Irradiance Forecast based on Image Processing and Cloud Motion Detection. In 2019 IEEE Texas Power and Energy Conference (pp. 1-6).* https://scholar.google.com

Ubidots. (n.d.). *IoT platform, Internet of Things and Ubidots.* https://www.ubidots.com

Vivas, E., Allende-Cid, H., & Salas, R. (2020). A systematic review of statistical and machine learning methods for electrical power forecasting with reported mape score. *Entropy*, 22(12), 1-24.

Wang, F., Yu, Y., Wang, X., Ren, H., Shafie-Khah, M., & Catalão, J. P. (2018). Residential electricity consumption level impact factor analysis based on wrapper feature selection and multinomial logistic regression. *Energies*, 11(5), 1-26.

Williams, N. J. (2017). *Microgrid Utilities for Rural Electrification in East Africa: Challenges and Opportunities* [*Carnegie Mellon University*]. https://scholar.google.com

Wukkadada, B., Wankhede, K., Nambiar, R., & Nair, A. (2018, July). *Comparison with HTTP and MQTT in Internet of Things (IoT). In 2018 International Conference on*

Inventive Research in Computing Applications (pp. 249-253). https:// scholar. google. com

Yang, J., Santamouris, M., & Lee, S. E. (2016). Review of occupancy sensing systems and occupancy modeling methodologies for the application in institutional buildings. *Energy and Buildings*, *121*, 344–349.

Yildiz, B., Bilbao, J. I., Dore, J., & Sproul, A. B. (2018). Short-term forecasting of individual household electricity loads with investigating impact of data resolution and forecast horizon. *Renewable Energy and Environmental Sustainability*, 3, 3, 1-9.

Yin, D., Miao, L., Li, G., Zhan, C., & Sun, L. (2022, November). *Moving Average-Based Performance Enhancement of Sample Convolution and Interactive Learning for Short-Term Load Forecasting. In 2022 IEEE International Symposium on Product Compliance Engineering-Asia (pp. 1-6)*. https://scholar.google.com.

Ying, X. (2019, February). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168, 1-7.

Zachariadis, T., & Hadjinicolaou, P. (2014). The effect of climate change on electricity needs: A case study from Mediterranean Europe. *Energy*, *76*, 899–910.

Zafar, U., Bayhan, S., & Sanfilippo, A. (2020). Home Energy Management System Concepts, Configurations, and Technologies for the Smart Grid. *Access*, *8*, 119271–119286. https://doi.org/10.1109/ACCESS.2020.3005244

Zhai, H., & Che, J. (2022). Combining PSO-SVR and Random Forest Based Feature Selection for Day-ahead Peak Load Forecasting. *Engineering Letters*, 30(1), 1-7.

Zhang, K., & Wu, L. (2021). Using Fractional Order Grey Seasonal Model to Predict the Power Generation in China. *Environmental Processes*, *8*(1), 413–427.

Zhang, L., Wen, J., Li, Y., Chen, J., Ye, Y., Fu, Y., & Livingood, W. (2021). A review of machine learning in building load prediction. *Applied Energy*, *285*, 116452.

Zhao, Y., Zeiler, W., Boxem, G., & Labeodan, T. (2015). Virtual occupancy sensors for real-time occupancy information in buildings. *Building and Environment*, *93*(P2), 9–20.

Zhao, Z., Zhang, Y., Yang, Y., & Yuan, S. (2022). Load forecasting via Grey Model-Least Squares Support Vector Machine model and spatial-temporal distribution of electric consumption intensity. *Energy*, *255*, 124468.

Zhou, S., & Brown, M. A. (2017). Smart meter deployment in Europe: A comparative case study on the impacts of national policy schemes. *Journal of Cleaner Production*, *144*, 22–32.

Zhu, D., Cheng, D., Broadwater, R. P., & Scirbona, C. (2007). Storm modeling for prediction of power distribution system outages. *Electric Power Systems Research*, *77*(8), 973–979. https://doi.org/10.1016/J.EPSR.2006.08.020

Zou, H., Jiang, H., Yang, J., Xie, L., & Spanos, C. J. (2017). Non-intrusive occupancy sensing in commercial buildings. *Energy and Buildings*, *154*, 633–643.

APPENDICES

Appendix 1: Ngarenanyuki Microgrid Data Collection Setup

Overview

Data access: The PLC has the capacity to store the school's grid electric data equivalent to 2 months, still, due to a PLC software bug, it saves 5 files (equivalent to 5 days data), and thereafter the PLC overwrites the files. Initially one of the school staff volunteered to manually download data, but with inconsistency leading to missing some data. An internet satellite dish equipment (Figure 1) was installed in the school in August 2015, which among many benefits, now facilitates remote access of the installed switchboards and PLC. Daily PLC data download process is now fully automated thanks to combination of a computer script, Cloud computing services and a task scheduler.

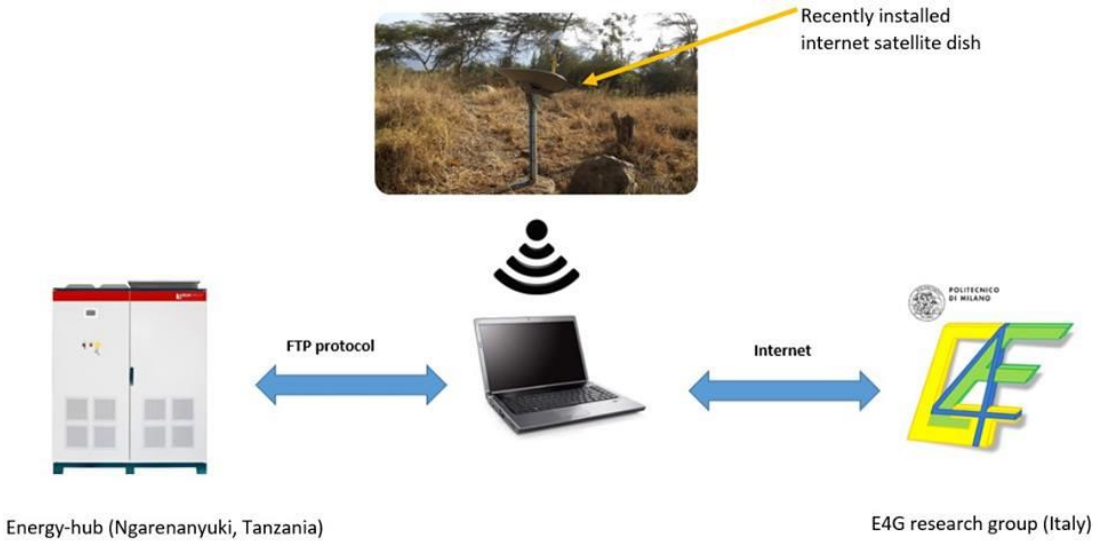


Figure 38: Internet satellite dish installed to provide E4G team with remote connectivity to the installed energy hub at Ngarenanyuki school

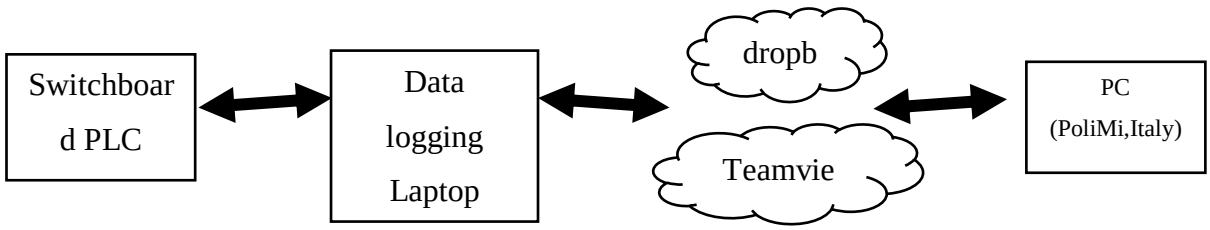


Figure 39: Ngarenanyuki Datalogging set-up

Description summary:

(i) A laptop located inside the school manager's office (James' office) runs the "E4GftpScript" which automatically copies csv files from the PLC into a dropbox

folder in the laptop. Since the Laptop is wirelessly connected to the internet, PLC csv files are automatically synchronized with another dropbox folder in a PC in Bovisa campus PoliMi. The more powerful Bovisa PC in turn, runs at least once a day, "E4GfileTransfer" script which startup Matlab code to analyse the csv files and send an email notification.

(ii) Ngarenanyuki laptop can be accessed remotely via teamviewer software, in the same manner the PLC data is accessible via teamviewer. Mostly PLC data is accessed by the "E4GftpScript" unsupervised/automatically.

(iii) **Challenges:** sometimes the Ngarenanyuki laptop goes offline and fails to synchronize via dropbox. Other times the laptop fails to fetch data files from the PLC, perhaps due to loose **Ethernet cable linking the PLC to the laptop**.

This documentation gives an overview on the automated email notification system for Ngarenanyuki Micro-grid. The automatic email notification setup involves configuration of the window task scheduler to execute a batch file, which in turn copies the most recent 8 csv files into the GUI folder. Matlab ngareEMSmain script is run to process the 8 csv files and generate the plots photos. The next script to be run is the ngareEMSmailer which is responsible for using the Matlab inbuilt sendmail function to send email with attachments to the specified recipients.

Figure 40: Automated Matlab email notification script setup

The automation batch file

In the case of Ngarenanyuki, it is only one batch file which is setup to run once each day.

Batch File explanation

1. The first section of the batch file ensures that the batch file script runs in administrator mode.

2. **del** command is used to erase previous 8 csv files from the GuiData folder in order to make room for the most recent new 8 csv files from the dropbox folder

3. **robocopy** command is used to copy the most recent 8 csv files from the dropbox folder to the GuiData folder.

4. **taskkill** command is used to close down any running matlab programs

5. With reference to figure 1, a new session/instance of matlab is run (-r) with with no startup matlab logo (-nosplash) to save time. ngareEMSmain matlab script is executed in the try-end block to avoid any possible error. When ngareEMSmain script is done processing the 8 csv files it will eventually call ngareEMSmailer in order to sendmail with attachments to the recipients.

Figure 41: Batch script for automatic email notification

Appendix 2: Load Forecast Matlab Source Code

Load data

```
clear;clc;
%load('WeatherTT_hourly.mat');
load Hourly_table.mat;
HourlyTT=table2timetable(Hourly_table);
% HourlyT.Time=datetime(HourlyT.Time,'InputFormat','yyyy.MM.dd HH:mm:ss');
% TT = table2timetable(HourlyT);
% TT = removevars(TT, {'Date','Minute'});
% HourlyTT = retime(TT,'hourly','mean');

% HourlyTT= rmmissing(HourlyTT); % removing missing values & NaNs

% DateVector = [Year,Month,Day];
% formatOut = 'yyyy.MM.dd';
% %Date=datetime(datenum(HourlyTT.Time,formatOut));
% Date= datetime(DateVector,'Format',formatOut);
% Date=string(Date);
% HourlyTT = addvars(HourlyTT,Date,'Before','Year');

clear Year Month Day H MN S
```

data smoothing

```
L=HourlyTT.Load;
%L=WeatherTT.Load;

%smoothTech = {'original','movmean','movmedian','sgolay','lowess','rlowess','loess','rloess','gaussian' };
%smoothTech = {'original','movmean'};
smoothTech = {'sgolay'};

switch smoothTech{1}
   case 'original'
%        close all;
       legend_str='original';
      y = L;
    subplot_str = ' subplot(3,3,1)';
   case 'movmean'
%        close all;
       legend_str='movmean';
      y = smoothdata(L);
      y(y<0)=0; % return to Zero all values below zero
     subplot_str = 'subplot(3,3,2)';
     %hold off
   case 'movmedian'
%        close all;
       legend_str='movmedian';
      y = smoothdata(L,'movmedian','SmoothingFactor',0.5);
      y(y<0)=0; % return to Zero all values below zero
     subplot_str = 'subplot(3,3,3)';
   case 'sgolay'
%        close all;
       legend_str='sgolay';
     y = smoothdata(L,'sgolay');
     y(y<0)=0; % return to Zero all values below zero
     subplot_str = 'subplot(3,3,4)';
   case 'lowess'
%        close all;
      legend_str='lowess';
     y = smoothdata(L,'lowess');
     y(y<0)=0; % return to Zero all values below zero
     subplot_str = 'subplot(3,3,5)';
```

```
   case 'rlowess'
%         close all;
        legend_str='rlowess';
       y = smoothdata(L,'rlowess');
       y(y<0)=0; % return to Zero all values below zero
      subplot_str =  'subplot(3,3,6)';
   case 'loess'
%         close all;
        legend_str='loess';
       y = smoothdata(L,'loess');
       y(y<0)=0; % return to Zero all values below zero
      subplot_str = 'subplot(3,3,7)';
   case 'rloess'
%         close all;
        legend_str='rloess';
       y = smoothdata(L,'rloess');
       y(y<0)=0; % return to Zero all values below zero
      subplot_str = 'subplot(3,3,8)';
   case 'gaussian'
%         close all;
       legend_str='gaussian';
       window = 5;
       SF = 0.25; %smoothing factor 0 to 1
      y = smoothdata(L,'gaussian','SmoothingFactor',SF);
      y(y<0)=0; % return to Zero all values below zero
     subplot_str = 'subplot(3,3,9)';
     hold off
   otherwise
       warning('Unexpected plot type. No plot created.')
end
```

Replace original load with smoothed load

```
Load=y;
smoothT=HourlyTT;
smoothT = removevars(smoothT, 'Load');
smoothT = addvars(smoothT,Load,'Before','SOC');
save('smoothT.mat','smoothT')
```

Test data

```
TestDays=fetchTestData('2018.02.01',30,smoothT);
ans = 'finished in 0.01 minutes'
Day-ahead load
% subroutine to introduce nexday load
D=[];
for i=2:length(TestDays)
A=TestDays{i-1};
B=TestDays{i};
nextDayLoad=B.Load;
C=[A table(nextDayLoad)];
D=[D;C];
end
TestData=D;
TestData = movevars(TestData, 'Load', 'After', 'T1_Load');
TestData = timetable2table(TestData);
```

Training data

```
myNNtable = fetchTrainData('2015.05.03','2018.01.31',smoothT);
ans = 'finished in 0.0 minutes'
myNNtable = movevars(myNNtable, 'Load', 'After', 'T1_Load');
```

```
toc/60
ans = 0.0020
Day-ahead load
% subroutine to introduce nexday load
Dates=unique(myNNtable.Date); % identify total number of unique dates
trainData = cell(length(Dates),1); %create empty cell
%myNNtable=table2timetable(myNNtable,predictorNames); %convert table to timetable in order to use
timerange function

for i=1:length(Dates)
S=timerange(datetime(Dates{i},'InputFormat','yyyy.MM.dd'),'days');
trainData{i}=timetable2table(myNNtable(S,:));
end

myNNtable=timetable2table(myNNtable);
D=[];
for i=2:length(Dates)
A=trainData{i-1};
B=trainData{i};
nextDayLoad=B.Load;
C=[A table(nextDayLoad)];
D=[D;C];
end
myNNtable=D;

clear A B C D nextDayLoad S i


%-------------------------------------------------------------------------------
% predictorNames = {'Month', 'Day', 'WeekDay', 'Hour', 'Weekend', 'temp', 'P_DG', 'P_HYD', 'P_inv',
'Vdc_bus', 'PPV', 'atmPressure', 'RHumidity', 'T2_temp', 'T2_Load', 'T1_temp', 'T1_Load', 'Load'};
predictorNames = {  'P_DG', 'P_HYD', 'P_inv',  'Load'};

function varargout = RegressionLearner(trainingData,TestData,predictorNames,varargin)
%function Ypred = RegressionLearner(trainingData,Method)

inputTable = trainingData;



Standardize Data
For a better fit and to prevent the training from diverging, standardize the training data to have zero mean and
unit variance. At prediction time, you must standardize the test data using the same parameters as the training
data.
if length(varargin)>=2  % only execute if there is more than two optional inputs
if strcmp(varargin{2}, 'standardize')
% mu = mean(smoothT);
% sig = std(smoothT);
%
% dataTrainStandardized = (smoothT - mu) / sig;

%important features
impFeatures=predictorNames;
impFeatures{1,end+1}='nextDayLoad';
D=[];
for k=1:length(impFeatures)
   A=inputTable(:,impFeatures{k});
   B=varfun(@mean,A); % mean aka mu
   C=varfun(@std,A); %standard deviation aka sigma
   %standardize
   A{:,impFeatures{k}}=(A{:,impFeatures{k}}-B{1,1})/C{1,1};
   if(k==(length(impFeatures)-1)) %load case
      mu_load=B{1,1};
```

```
        sigma_load=C{1,1};
    elseif(k==length(impFeatures))%nextDayLoad case
         mu_nextDayLoad=B{1,1};
        sigma_nextDayLoad=C{1,1};
    end
    D=[D A];
end

inputTable=D;
end
end

% predictorNames = {'Month', 'Day', 'WeekDay', 'Hour', 'Weekend', 'temp', 'P_DG', 'P_HYD', 'P_inv',
'Vdc_bus', 'PPV', 'atmPressure', 'RHumidity', 'T2_temp', 'T2_Load', 'T1_temp', 'T1_Load', 'Load'};
predictors = inputTable(:, predictorNames);

response = inputTable.nextDayLoad;
% isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false];

% Train a regression model
% This code specifies all the model options and trains the model.
concatenatedPredictorsAndResponse = predictors;
concatenatedPredictorsAndResponse.nextDayLoad = response;



% Method = 'sgolay';
METHOD=varargin{1};
if strcmp(METHOD, 'best')
     Method = {'linear','interactions','robustLinear','stepwiseLinear',...
          'FineTree','MediumTree','CoarseTree','BaggedTree','BoostedTree',...
         'linearSVM','QuadraticSVM','CubicSVM',...
         'FineGSVM','MediumGSVM','CoarseGSVM',...
         'seGPR','MaternGPR','eGPR','rqGPR','neuralnet'};
elseif strcmp(METHOD, 'very best')
      Method = {'linear','interactions','robustLinear','stepwiseLinear',...
          'FineTree','MediumTree','CoarseTree','BaggedTree','BoostedTree',...
         'linearSVM','QuadraticSVM','CubicSVM',...
         'FineGSVM','MediumGSVM','CoarseGSVM',...
         'seGPR','MaternGPR','eGPR','rqGPR','neuralnet'};
else
    Method={METHOD};
end

varargout = cell(nargout,1);

for k=1:length(Method)

switch Method{k}
    case 'linear'

     linearModel = fitlm(...
    concatenatedPredictorsAndResponse, ...
    'linear', ...
    'RobustOpts', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
linearModelPredictFcn = @(x) predict(linearModel, x);
linearRegression_Model.predictFcn = @(x) linearModelPredictFcn(predictorExtractionFcn(x));
trainedModel=linearRegression_Model;
yfit=linearRegression_Model.predictFcn(TestData);
```

```
yfit(yfit<0)=0;


     %subplot_str = ' subplot(3,3,1)';


   case 'interactions'

      linearModel = fitlm(...
   concatenatedPredictorsAndResponse, ...
   'interactions', ...
   'RobustOpts', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
linearModelPredictFcn = @(x) predict(linearModel, x);
interactionsRegression_Model.predictFcn = @(x) linearModelPredictFcn(predictorExtractionFcn(x));
trainedModel=interactionsRegression_Model;
yfit=interactionsRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


     % subplot_str = 'subplot(3,3,2)';


   case 'robustLinear'

      linearModel = fitlm(...
   concatenatedPredictorsAndResponse, ...
   'linear', ...
   'RobustOpts', 'on');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
linearModelPredictFcn = @(x) predict(linearModel, x);
robustLinearRegression_model.predictFcn = @(x) linearModelPredictFcn(predictorExtractionFcn(x));
trainedModel=robustLinearRegression_model;
yfit=robustLinearRegression_model.predictFcn(TestData);
yfit(yfit<0)=0;


   case 'stepwiseLinear'

      linearModel = stepwiselm(...
   concatenatedPredictorsAndResponse, ...
   'linear', ...
   'Upper', 'interactions', ...
   'NSteps', 1000, ...
   'Verbose', 0);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
linearModelPredictFcn = @(x) predict(linearModel, x);
stepwiseLinearRegression_Model.predictFcn = @(x) linearModelPredictFcn(predictorExtractionFcn(x));
trainedModel=stepwiseLinearRegression_Model;
yfit=stepwiseLinearRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


   case 'FineTree'
   % Train a regression model
% This code specifies all the model options and trains the model.
regressionTree = fitrtree(...
```

```
    predictors, ...
    response, ...
    'MinLeafSize', 4, ...
    'Surrogate', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(regressionTree, x);
FineTreeRegression_Model.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));
trainedModel=FineTreeRegression_Model;
yfit=FineTreeRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'MediumTree'

        % Train a regression model
% This code specifies all the model options and trains the model.
regressionTree = fitrtree(...
    predictors, ...
    response, ...
    'MinLeafSize', 12, ...
    'Surrogate', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(regressionTree, x);
MediumTreeRegression_Model.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));
trainedModel=MediumTreeRegression_Model;
yfit=MediumTreeRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'CoarseTree'
        % Train a regression model
% This code specifies all the model options and trains the model.
regressionTree = fitrtree(...
    predictors, ...
    response, ...
    'MinLeafSize', 36, ...
    'Surrogate', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(regressionTree, x);
CoarseTreeRegression_Model.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));
trainedModel=CoarseTreeRegression_Model;
yfit=CoarseTreeRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;

     case 'BaggedTree'

        % Train a regression model
% This code specifies all the model options and trains the model.
template = templateTree(...
    'MinLeafSize', 8);
regressionEnsemble = fitrensemble(...
    predictors, ...
    response, ...
    'Method', 'Bag', ...
    'NumLearningCycles', 30, ...
    'Learners', template);
```

```
% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
ensemblePredictFcn = @(x) predict(regressionEnsemble, x);
BaggedTree_Model.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));
trainedModel=BaggedTree_Model;
yfit=BaggedTree_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'BoostedTree'

        % Train a regression model
% This code specifies all the model options and trains the model.
template = templateTree(...
    'MinLeafSize', 8);
regressionEnsemble = fitrensemble(...
    predictors, ...
    response, ...
    'Method', 'LSBoost', ...
    'NumLearningCycles', 30, ...
    'Learners', template, ...
    'LearnRate', 0.1);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
ensemblePredictFcn = @(x) predict(regressionEnsemble, x);
BoostedTreesRegression_Model.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));
trainedModel=BoostedTreesRegression_Model;
yfit=BoostedTreesRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;

    case 'linearSVM'

        % Train a regression model
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
    responseScale = 1.0;
end
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
    predictors, ...
    response, ...
    'KernelFunction', 'linear', ...
    'PolynomialOrder', [], ...
    'KernelScale', 'auto', ...
    'BoxConstraint', boxConstraint, ...
    'Epsilon', epsilon, ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
LinearSVMRegression_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=LinearSVMRegression_Model;
yfit=LinearSVMRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'QuadraticSVM'

        % Train a regression model
```

```
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
   responseScale = 1.0;
end
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
   predictors, ...
   response, ...
   'KernelFunction', 'polynomial', ...
   'PolynomialOrder', 2, ...
   'KernelScale', 'auto', ...
   'BoxConstraint', boxConstraint, ...
   'Epsilon', epsilon, ...
   'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
QSVM_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=QSVM_Model;
yfit=QSVM_Model.predictFcn(TestData);
yfit(yfit<0)=0;


   case 'CubicSVM'

      % Train a regression model
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
   responseScale = 1.0;
end
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
   predictors, ...
   response, ...
   'KernelFunction', 'polynomial', ...
   'PolynomialOrder', 3, ...
   'KernelScale', 'auto', ...
   'BoxConstraint', boxConstraint, ...
   'Epsilon', epsilon, ...
   'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
CubicSVMRegression_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=CubicSVMRegression_Model;
yfit=CubicSVMRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


   case 'FineGSVM'

      % Train a regression model
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
   responseScale = 1.0;
end
```

```
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
    predictors, ...
    response, ...
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 1.1, ...
    'BoxConstraint', boxConstraint, ...
    'Epsilon', epsilon, ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
FineGSVMRegression_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=FineGSVMRegression_Model;
yfit=FineGSVMRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'MediumGSVM'

        % Train a regression model
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
    responseScale = 1.0;
end
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
    predictors, ...
    response, ...
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 4.2, ...
    'BoxConstraint', boxConstraint, ...
    'Epsilon', epsilon, ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
MediumSVMGaussianRegression_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=MediumSVMGaussianRegression_Model;
yfit=MediumSVMGaussianRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'CoarseGSVM'

        % Train a regression model
% This code specifies all the model options and trains the model.
responseScale = iqr(response);
if ~isfinite(responseScale) || responseScale == 0.0
    responseScale = 1.0;
end
boxConstraint = responseScale/1.349;
epsilon = responseScale/13.49;
regressionSVM = fitrsvm(...
    predictors, ...
    response, ...
```

```
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 17, ...
    'BoxConstraint', boxConstraint, ...
    'Epsilon', epsilon, ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(regressionSVM, x);
CoarseGSVMRegression_Model.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));
trainedModel=CoarseGSVMRegression_Model;
yfit=CoarseGSVMRegression_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'seGPR'

        % Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
    'BasisFunction', 'constant', ...
    'KernelFunction', 'squaredexponential', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
squaredeGPR_Model.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));
trainedModel=squaredeGPR_Model;
yfit=squaredeGPR_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'MaternGPR'

        % Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
    'BasisFunction', 'constant', ...
    'KernelFunction', 'matern52', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
matern52gpModel.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));
trainedModel=matern52gpModel;
yfit=matern52gpModel.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'eGPR'

        % Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
```

```
    'BasisFunction', 'constant', ...
    'KernelFunction', 'exponential', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
eGPR_Model.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));
trainedModel=eGPR_Model;
yfit=eGPR_Model.predictFcn(TestData);
yfit(yfit<0)=0;

    case 'rqGPR'

        % Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
    'BasisFunction', 'constant', ...
    'KernelFunction', 'rationalquadratic', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
RationalQuadraticGPR_Model.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));
trainedModel=RationalQuadraticGPR_Model;
yfit=RationalQuadraticGPR_Model.predictFcn(TestData);
yfit(yfit<0)=0;


    case 'LSTM'

    case 'ARIMA'

    case 'neuralnet'

% Train a regression model
predictors= table2array(predictors)';
response = response';

TestData_Load=TestData.Load';
Test_Data=TestData(:,predictorNames);
Test_Data=table2array(Test_Data)';
% TestData=TestData';

% myTrainers={'trainlm','trainscg','trainbr','trainbfg','traincgb',...
%     'traincgf','traincgp','traingd','traingda','traingdm','traingdx','trainoss','trainrp'};
% trainFxn = num2str(cell2mat(myTrainers(1,kk)));  % Levenberg-Marquardt backpropagation.
trainFxn = 'trainlm';

% Create a Fitting Network
% hiddenLayerSize = 10*i;
hdnLSize = 10;
net = fitnet(hdnLSize,trainFxn);
%net = cascadeforwardnet(hdnLSize,trainFxn);
%net = feedforwardnet(hdnLSize,trainFxn);
% Setup Division of Data for Training, Validation, Testing
rng('default')
net.trainParam.epochs = 100;
net.trainParam.showWindow = 0;
```

```
net.divideFcn = 'dividerand';
% net.divideFcn = 'divideblock';
%net.divideFcn = 'divideint';

% use only with either dividerand / divideblock / divideint
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
%[net,tr] = train(net,x,t,'useParallel','yes');
NNnet_model = train(net,predictors,response,'useParallel','yes');

% Create the result struct with predict function

trainedModel=NNnet_model;

yfit = sim(NNnet_model, Test_Data)';
yfit(yfit<0)=0;


% % Test the Network
% y = NN_net(x);
% e = gsubtract(t,y);
% performance = perform(NN_net,t,y);
%
% clear y e



% [net,y,e] = adapt(net,x,t);

   otherwise
      warning('Unexpected plot type. No plot created.')
end

%------------ inside for loop -------------------
%--------------- error performance calculation --------------
ERROR=yfit-TestData.Load;
RMSE = sqrt(mean((ERROR).^2));
MAE=mean(abs(ERROR));
% MAPE=mean(abs((ERROR)./TestData.Load))*100;
% ------------- destandardization -------------------------
try
if strcmp(varargin{2}, 'standardize')
   disp('standardization opted!');
%destandardize
%yfit=yfit*sigma_nextDayLoad + mu_nextDayLoad;

end
catch

end
%--------------end of standardization routine ------------------

%-------------- concatenate results on each loop --------------------------
ResultData{k,1}=Method{k}; % Regression method name
ResultData{k,2}=trainedModel;
ResultData{k,3}=yfit; % prediction
ResultData{k,4}=RMSE;
ResultData{k,5}=MAE;
%ResultData{k,6}=MAPE;
ResultData{k,6}=predictorNames';
```

```
end

%-------------- outside for loop ---------------------------
% -------------output variable result ----------------------

if strcmp(METHOD, 'very best')
   ResultData=sortrows(ResultData,5,'ascend');
   ResultData=ResultData(1,:);
%    trainedModel_Name=ResultData{1,1};
%    trainedModel=ResultData{1,2};
%    yfit=ResultData{1,3};
%    RMSE=ResultData{1,4};
%    MAE=ResultData{1,5};
%    %MAPE=ResultData{1,6};
%    predictorNames=ResultData{1,6};
%
%  result={ResultData,trainedModel_Name,yfit,RMSE,MAE,trainedModel};
  result={ResultData};

for k=1:nargout
      varargout{k}=result{k};
end
%varargout=ResultData;
elseif strcmp(METHOD, 'best')
   ResultData=sortrows(ResultData,5,'ascend');
   trainedModel_Name=ResultData{1,1};
   trainedModel=ResultData{1,2};
   yfit=ResultData{1,3};
   RMSE=ResultData{1,4};
   MAE=ResultData{1,5};
   %MAPE=ResultData{1,6};
   predictorNames=ResultData{1,6};

 result={ResultData,trainedModel_Name,yfit,RMSE,MAE,trainedModel};

for k=1:nargout
      varargout{k}=result{k};
end
else
%    ResultData=sortrows(ResultData,5,'ascend');
   trainedModel_Name=METHOD;
   result={ResultData,trainedModel_Name,yfit,RMSE,MAE,trainedModel};

for k=1:nargout
      varargout{k}=result{k};
end

end

% ------------------------------------------------------------

end
```

linearRegression_Model

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'l
inear');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('LinearRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
```

```
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'l
inear','standardize');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('LinearRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

interactionsRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'i
nteractions');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('interactions Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

RobustLinearRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'r
obustLinear');
Warning: Iteration limit reached.
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('robustLinearRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

stepwiseLinearRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
stepwiseLinear');

plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('stepwiseLinearRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

FineTreeRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
FineTree');

plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('FineTreeRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
```

```
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

MediumTreeRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
MediumTree');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('MediumTreeRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

CourseTreeRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
CoarseTree');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('CoarseTreeRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

LinearSVM_Regression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'l
inearSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('LinearSVMRegression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

QSVM_Model

```
%% make sure model inputs are tables and not timetables
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
QuadraticSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Quadratic SVM Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

CubicSVM_Model

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
CubicSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Cubic SVM Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
```

```
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

FineGaussianSVMRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
FineGaussianSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Fine Gaussian SVM Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

MediumGaussianSVMRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
MediumGaussianSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Medium Gaussian SVM Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

CoarseGaussianSVMRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
CoarseGaussianSVM');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Coarse Gaussian SVM Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

BaggedTree_Model

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
BaggedTree');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('BaggedTree Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

BoostedTreesRegression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
BoostedTree');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Boosted Trees Regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
```

```
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

Squared Exponential Gaussian Process Regression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
seGPR');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Squared exponential GPR Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

Matern 5/2 Gaussian Process Regression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
MaternGPR');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('matern52GP Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

Exponential Gaussian Process Regression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'
exponentialGPR');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
str = sprintf('Exponential GPR Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

Rational quadratic Gaussian Process Regression

```
[ResultData,trainedModel_Name,yfit,RMSE,MAE]=RegressionLearner(myNNtable,TestData,predictorNames,'r
qGPR');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
RMSE = sqrt(mean((yfit-TestData.Load).^2));
MAE=mean(abs(yfit-TestData.Load));
MAE=mean(abs(yfit-TestData.Load));
MAE=sum(abs(yfit(:)-TestData.Load(:)))/numel(yfit)

str = sprintf('Rational quadratic GPR Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
```

Neural Network

```
[trainedModel_Name,trainedModel,yfit,RMSE,MAE,ResultData]=RegressionLearner(myNNtable,TestData,pre
dictorNames,'neuralnet');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',trainedModel_Name,RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
```

```
legend(["Observed" "Predicted"])
```

Best prediction model

```
% predictorNames = {'Month', 'Day', 'WeekDay', 'Hour', 'Weekend', 'temp', 'P_DG', 'P_HYD', 'P_inv',
'Vdc_bus', 'PPV', 'atmPressure', 'RHumidity', 'T2_temp', 'T2_Load', 'T1_temp', 'T1_Load', 'Load'};
% predictorNames = {'Month', 'Day', 'Hour', 'P_DG', 'P_HYD', 'P_inv', 'Vdc_bus', 'RHumidity', 'T2_Load',
'T1_Load', 'Load'};
% predictorNames = { 'Hour', 'P_DG', 'P_HYD', 'P_inv', 'Vdc_bus', 'RHumidity', 'Load'};
% predictorNames = { 'Hour', 'P_DG', 'P_HYD', 'P_inv',  'Load'};
predictorNames = {  'P_DG', 'P_HYD', 'P_inv',  'Load'};

[ResultData,trainedModel_Name,yfit,RMSE,MAE,MAPE]=RegressionLearner(myNNtable,TestData,predictor
Names,'best');
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',trainedModel_Name,RMSE,MAE);
title(str);
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])


%
% predictorNames = {  'P_DG', 'P_HYD', 'P_inv',  'Load'};
%
[trainedModel_Name,trainedModel,yfit,RMSE,MAE,ResultData]=RegressionLearner(myNNtable,TestData,pre
dictorNames,'linear');
for k=1:length(ResultData)
   if k<=9
 figure(1)
      subplot_str = sprintf('subplot(3,3,%0.0f)',k);
eval(subplot_str)
yfit=ResultData{k,3};
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s  (RMSE=%0.0f, MAE=%0.0f) ',ResultData{k,1},ResultData{k,4},ResultData{k,5});
title(str);
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
hold on
   elseif k>9 && k<=18
    figure(2)
      subplot_str = sprintf('subplot(3,3,%0.0f)',k-9);
eval(subplot_str)
yfit=ResultData{k,3};
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s  (RMSE=%0.0f, MAE=%0.0f) ',ResultData{k,1},ResultData{k,4},ResultData{k,5});
title(str);
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
hold on
if k==9 || k==18
      hold off
   end
   elseif k>18
     figure(3)
      subplot_str = sprintf('subplot(2,1,%0.0f)',k-18);
eval(subplot_str)
yfit=ResultData{k,3};
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s  Load Forecast (RMSE=%0.0f, MAE=%0.0f)
',ResultData{k,1},ResultData{k,4},ResultData{k,5});
title(str);
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
hold on
```

```
if k==9 || k==18
       hold off
   end
   end

end

hold off
```

Bargraph: Prediction model comparison by error

```
methods=ResultData(:,1);
methods=categorical(methods,methods);
Mae_bar=cell2mat(ResultData(:,5));
RMSE_bar=cell2mat(ResultData(:,4));
error_index=[Mae_bar,RMSE_bar];
bar(methods,error_index)
title('Prediction model comparison by error')
xlabel('Prediction model');ylabel('Error');
legend(["mae" "rmse"])
```

hybrid models

```
hybrid={};
p=0;
for n=1:length(ResultData)
m=n;
for k=1:length(ResultData)
   p=p+1;
A=ResultData{m,3};
B=ResultData{k,3}; %i.e B=A+1...next model in 'sliding window'
yyfit=[A';B'];
yfit_mean=mean(yyfit)';
yfit_max=max(yyfit)';
yfit_min=min(yyfit)';
RMSE_mean = sqrt(mean((yfit_mean-TestData.Load).^2));
MAE_mean=mean(abs(yfit_mean-TestData.Load));
RMSE_max = sqrt(mean((yfit_max-TestData.Load).^2));
MAE_max=mean(abs(yfit_max-TestData.Load));
RMSE_min = sqrt(mean((yfit_min-TestData.Load).^2));
MAE_min=mean(abs(yfit_min-TestData.Load));

   hybrid{p,1}=ResultData{m,1}; % model name
   hybrid{p,2}=ResultData{k,1}; % model name
   hybrid{p,3}=ResultData{m,4}; % rmse
   hybrid{p,4}=ResultData{m,5}; % mae
   hybrid{p,5}=ResultData{k,4}; % rmse
   hybrid{p,6}=ResultData{k,5}; % mae
   hybrid{p,7}=RMSE_mean; % mean hybrid rmse
   hybrid{p,8}=MAE_mean; % mean hybrid mae
   hybrid{p,9}=RMSE_max; % max hybrid rmse
   hybrid{p,10}=MAE_max; % max hybrid rmse
   hybrid{p,11}=RMSE_min; % max hybrid rmse
   hybrid{p,12}=MAE_min; % max hybrid rmse
end
end

hybrid=sortrows(hybrid,10,'ascend');

% create table
modelA=hybrid(:,1);
modelB=hybrid(:,2);
rmseA=hybrid(:,3);
```

```
maeA=hybrid(:,4);
rmseB=hybrid(:,5);
maeB=hybrid(:,6);
rmseMean=hybrid(:,7);
maeMean=hybrid(:,8);
rmseMax=hybrid(:,9);
maeMax=hybrid(:,10);
rmseMin=hybrid(:,11);
maeMin=hybrid(:,12);

hybrid=[table(modelA) table(modelB) table(rmseA) table(maeA) table(rmseB) table(maeB) ...
   table(rmseMean) table(maeMean) table(rmseMax) table(maeMax) table(rmseMin) table(maeMin)];

%remove duplicate rows
k=1;
while k<height(hybrid)
A=hybrid{k,1:2};
B=hybrid{k+1,1:2};
C=ismember(A,B);
if C(1)&C(2)
   hybrid(k+1,:)=[];
end
k=k+1;
end

%remove duplicate 1st and 2nd rows
k=1;
while k<=height(hybrid)
A=hybrid{k,1};
B=hybrid{k,2};
if strcmp(A,B)
   hybrid(k,:)=[];
end
k=k+1;
end

subplot(5,1,1)
plot(TestData.Time,TestData.Load,TestData.Time,A)
RMSE_A = sqrt(mean((A-TestData.Load).^2));
MAE_A=mean(abs(A-TestData.Load));
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',ResultData{1,1},RMSE_A,MAE_A);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])

subplot(5,1,2)
plot(TestData.Time,TestData.Load,TestData.Time,B)
RMSE_B = sqrt(mean((B-TestData.Load).^2));
MAE_B=mean(abs(B-TestData.Load));
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',ResultData{2,1},RMSE_B,MAE_B);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])

subplot(5,1,3)
plot(TestData.Time,TestData.Load,TestData.Time,yfit_mean)
RMSE_mean = sqrt(mean((yfit_mean-TestData.Load).^2));
MAE_mean=mean(abs(yfit_mean-TestData.Load));
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ','mean',RMSE_mean,MAE_mean);
title(str);
% title('Load Forecast');
```

```
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])

subplot(5,1,4)
plot(TestData.Time,TestData.Load,TestData.Time,yfit_max)
RMSE_max = sqrt(mean((yfit_max-TestData.Load).^2));
MAE_max=mean(abs(yfit_max-TestData.Load));
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ','max',RMSE_max,MAE_max);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
subplot(5,1,5)
plot(TestData.Time,TestData.Load,TestData.Time,yfit_min)
RMSE_min = sqrt(mean((yfit_min-TestData.Load).^2));
MAE_min=mean(abs(yfit_min-TestData.Load));
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ','max',RMSE_min,MAE_min);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])


%------- confidence band -------------------------
xData =datenum(TestData.Time);
SEM = std(xData)/sqrt(length(xData));
err=SEM*yfit;
hi=yfit+err;
lo=yfit-err;
lo(lo<0)=0; %prevent negative energy/watts
%

% calculate confidence interval
ci=0.95;
a=1-ci;
n=size(xData,1);
T_multiplier=tinv(1-a/2,n-1);
ci95 = T_multiplier*std(xData)/sqrt(n);

%apply confidence interval to data
err_ci=ci95*yfit;

%err=SEM*yData;
hi=yfit+err_ci;
lo=yfit-err_ci;
lo(lo<0)=0; %prevent negative energy/watts


%plot(TestData.Time,TestData.Load,TestData.Time,yfit,[hi,lo],'--m')

figure
plot(TestData.Time,yfit)
ax1 = gca; % current axes
ax1.YColor = 'r';

%

ax1_pos = ax1.Position; % position of first axes
ax2 = axes('Position',ax1_pos,...
   'XAxisLocation','top',...
   'YAxisLocation','right',...
   'Color','none');
hold on
```

```
plot(TestData.Time,[hi,lo],'Parent',ax2,'Color','k')
opt={'-s','MarkerSize',4,'MarkerEdgeColor','red','MarkerFaceColor','red','CapSize',12};


figure( 'Name', 'errorbar based on ci' );
errorbar(xData,yfit,err_ci,opt{:});



hold on
plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',trainedModel_Name,RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted" "Confidence band"])

legend(["Observed" "Predicted"])

% predictorNames = {'Month', 'Day', 'WeekDay', 'Hour', 'Weekend', 'temp', 'P_DG', 'P_HYD', 'P_inv',
'Vdc_bus', 'PPV', 'atmPressure', 'RHumidity', 'T2_temp', 'T2_Load', 'T1_temp', 'T1_Load', 'Load'};
predictorNames = {  'P_DG', 'P_HYD', 'P_inv',  'Load'};
[trainedModel_Name,trainedModel,yfit,RMSE,MAE,ResultData]=RegressionLearner(myNNtable,TestData,pre
dictorNames,'best');

plot(TestData.Time,TestData.Load,TestData.Time,yfit)
str = sprintf('%s regression Load Forecast (RMSE=%0.0f, MAE=%0.0f) ',trainedModel_Name,RMSE,MAE);
title(str);
% title('Load Forecast');
xlabel('Time');ylabel('Load [W]');
legend(["Observed" "Predicted"])
hold off

Result_Data=sortrows(ResultData,5,'ascend');
```

Appendix 3 : Blackout Forecast Python Source Code

```
import os
import winsound
from keras.regularizers import L1L2
# this line is important in order to suppress numpy and pandas import errors
os.environ['OPENBLAS_NUM_THREADS'] = '1'
import traceback
#from signal import signal, SIGPIPE, SIG_DFL
#Ignore SIG_PIPE and don't throw exceptions on it... (http://docs.python.org/library/signal.html)
#signal(SIGPIPE,SIG_DFL)
import numpy as np
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
#import pymysql
#from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from scipy.interpolate import UnivariateSpline
from scipy.signal import savgol_filter
import joblib
import datetime # to get current time for error logging
import time # to time routines/tasks durations
import pytz # to convert from server to Tanzania timezone

import statistics

# machine learning modules
from sklearn.svm import SVR
import sklearn.metrics as metrics
import math
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing, svm
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

def blackout_forecast_15min():
        tic = time.perf_counter()
        #allDays = convert_to_15min()
        #check_timer(tic)
        allDays = pd.read_pickle("./allDays15min_filled.pkl")
        #allDays = pd.read_pickle("./allDays15min.pkl")

        # fetch all dates again, this now includes valid dataframes
        all_dates=allDays['Date'].unique()
        #print("last day is ",len(allDays.loc[allDays['Date'] == allDays.iloc[-1].Date] )," records long!")
        #print("allDays\n",len(allDays.loc[allDays['Date'] == all_dates[-1]]))
#       print("last day \n",allDays.loc[allDays['Date'] == allDays.iloc[-
1].Date,('Date','hour','minute')].to_string(index=False) )
        #print("allDays\n",allDays)
        print("final all_dates \n",allDays['Date'].unique())
        print("total dataset days \n",len(allDays['Date'].unique()))


        trainData=allDays.loc[allDays.Date<='2021-03-31']
```

```
        testData=allDays.loc[allDays.Date >'2021-03-31']
        #print("trainData\n",trainData)
        #print("testData\n",testData)

        #------------ BLACKOUT PREDICTIONS ----------------------

        # BLACKOUT MODEL TRAINING ---------------------------------
        next_blackout=allDays.loc[allDays.Date>=all_dates[14] ].blackout
        #print("next_blackout startdate ",all_dates[14],"currentDay startdate ",all_dates[13])
        #next_blackout=allDays.loc[allDays.Date >= all_dates[14] ]
        #print("next_blackout\n",next_blackout)
        #next_blackout = next_blackout['Date','blackout']
        #print("start_date length is :",len(next_blackout))
        #print("start day index ",next_blackout.index.values[0])
        currentDay=allDays.loc[allDays['Date'] >= all_dates[13],: ]
        #print("currentDay\n",currentDay)
        #print("final all_dates \n",all_dates[13] )

        #print("day2 length is :",len(currentDay))
        currentDay.reset_index(drop=True, inplace=True)
        currentDay=currentDay.loc[0:len(next_blackout)-1]
        #print("tomorrow is ",all_dates[14],
        #"\nsimilar day 14 days back : ",(datetime.datetime.strptime(all_dates[14], '%Y-%m-%d')-
datetime.timedelta(days=14)).strftime('%Y-%m-%d'),
        #"\n the is also :",all_dates[0])

        #select relevant columns only
        desired_features = ['Date','weekday','hour','day','ac_voltage','frequency','blackout']
        currentDay = currentDay[desired_features]
        d1=allDays.loc[allDays.Date>=all_dates[12]].blackout
        d1.reset_index(drop=True, inplace=True)
        d1=d1.loc[0:len(next_blackout)-1]
        d2=allDays.loc[allDays.Date>=all_dates[11]].blackout
        d2.reset_index(drop=True, inplace=True)
        d2=d2.loc[0:len(next_blackout)-1]
        d7=allDays.loc[allDays.Date>=all_dates[7]].blackout
        d7.reset_index(drop=True, inplace=True)
        d7=d7.loc[0:len(next_blackout)-1]
        d14=allDays.loc[allDays.Date>=all_dates[0]].blackout
        d14.reset_index(drop=True, inplace=True)
        d14=d14.loc[0:len(next_blackout)-1]

        currentDay =
currentDay.assign(b14=d14.values,b7=d7.values,b2=d2.values,b1=d1.values,next_blackout=next_blackout.valu
es)
        currentDay =
currentDay[['Date','weekday','hour','day','ac_voltage','frequency','b14','b7','b2','b1','blackout','next_blackout']]
        #print("final dataset \n",currentDay.loc[0:47])
        #print("final dataset \n",currentDay)
        #print("final dataset \n",currentDay.info())



        predictorNames=currentDay.columns.values[1:-1]
        #print("predictors: ",predictorNames)
        #print("predictors length: ",len(predictorNames))
        #x=currentDay[predictorNames].values
        #y=currentDay['next_blackout'].values

        # ------------ train and test data split ---------------
        train_dates=currentDay['Date'].unique()
        #print("train dates\n",train_dates)
```

```
#print("currentDay\n",len(currentDay))
#print("currentDay\n",currentDay)
#get train 80% and 20% test
#cutoff=int(np.floor(len(train_dates)*0.8))
cutoff=int(np.floor(len(train_dates)*0.2))
cutoff=str(train_dates[cutoff])
#cutoff="2021-05-31"
print("cutoff date: ",cutoff)

x_train=currentDay[currentDay['Date']<=cutoff]

x_train=x_train[predictorNames].values
y_train=currentDay[currentDay['Date']<=cutoff]
y_train=y_train['next_blackout'].values

#x_test=currentDay[(currentDay['Date']>cutoff) & (currentDay['Date']< "2021-07-01")]
#x_test=currentDay[(currentDay['Date']>"2021-10-03") & (currentDay['Date']<="2021-04-01")]
x_test=currentDay[(currentDay['Date']>cutoff) ]
#print("x_test\n",x_test[0:100].to_string(index=False))
print("x_test\n",x_test['Date'].unique())



#x_test=x_test[predictorNames].values
#x_test=x_test[predictorNames]
y_test=currentDay[currentDay['Date']>cutoff]
y_test=y_test[{'next_blackout','Date'}]
#print("x_train is\n",x_train)
#print("y_train is\n",y_train)
#print("x_test is\n",x_test)
#print("y_test is\n",y_test)


blackout_pred_rf=[]
blackout_actual=[]
#observations=list()
maeScore_rf=list()
mapeScore_rf=list()
mseScore_rf=list()
r2Score_rf=list()



historyX=[x for x in x_train]
historyY=[y for y in y_train]


#Random forest model
blackoutmodel_rf = RandomForestRegressor(n_estimators=10)
#forward-walk-validation
test_dates=x_test['Date'].unique()
#print("all dates tail\n",test_dates)
#print("y_test\n",y_test)
#model_svr.fit(x_train, y_train)

#ytest=ytest.values
# oggi = currentDay.loc[currentDay.Date=="2021-09-14"]

blocker=0

dayLength = 24*4 #each hour has 4 quarters
start_index = 0
prev_amp = 0
```

```
	for k in range(start_index,len(test_dates)):
	#for k in range(3):
#	for k in range(len(test_dates)):
				print("\nOuter loop is ",k)
				print("date is ",test_dates[k])
				print("length of test_dates is ",len(test_dates))
				xtest=x_test[x_test['Date']==test_dates[k]]

				xtest.reset_index(drop=True, inplace=True)

				print("*** Length of day : ",len(xtest))

				#make sure you only fetch a day 24hrs long, if longer trim it to only 24hrs
				# length will be 96 for a 15min interval day
				if len(xtest)>dayLength:
					xtest = xtest.loc[0:dayLength-1]

				ytest=xtest['next_blackout'].values
				xtest=xtest.values[:,1:-1]


				#print("length of xtest is ",len(xtest),"\nxtest is\n",xtest)

				oggi = currentDay.loc[currentDay.Date==test_dates[k]]
				oggi.reset_index(drop=True, inplace=True)
				if len(oggi)>dayLength:
					oggi = oggi.loc[0:dayLength-1]

				minCount=0
				if sum(oggi.b14)>0 or sum(oggi.b7)>0 or sum(oggi.b2)>0 or sum(oggi.b1)>0:
					minCount=1

				# now count blackout events in the past 24hrs, assume they have high influence on
the next 24hrs
				maxCount=0
				for i, element in enumerate(oggi.blackout):
					if oggi.blackout[i]>0:
						maxCount+=1
				#if there was no blackout event, then allow only 1 positive blackout event in the next
24hrs
				if maxCount==0:
					maxCount=minCount

				# check max amplitude (blackout duration) in the past 2weeks, assume this to be the
max permissible prediction amplitude for next 24hrs prediction
				maxAmplitude=max(oggi.b14)
				if maxAmplitude < max(oggi.b7):
					maxAmplitude=max(oggi.b7)
				if maxAmplitude < max(oggi.b2):
					maxAmplitude=max(oggi.b2)
				if maxAmplitude < max(oggi.b1):
					maxAmplitude=max(oggi.b1)
				if maxAmplitude < max(oggi.blackout):
					maxAmplitude=max(oggi.blackout)
				maxAmplitude =
statistics.mean([max(oggi.b14),max(oggi.b7),max(oggi.b2),max(oggi.b1),max(oggi.blackout)])
				# pred_permit allows predictions or suppresses them
				pred_permit=1
				asd_interval1 = 0
				asd_interval2 = 0
				next_day = [0]*dayLength
				# make hourly prediction for current day before next outer incremented day loop
				for j in range(len(oggi)):
```

```
                                print("\ncurrent minute quarter is ",j)
                                #u=b14 v=b7 w=b2 x=b1 y=blackout z=next_blackout
                                u=oggi.b14[j]
                                v=oggi.b7[j]
                                w=oggi.b2[j]
                                x=oggi.b1[j]
                                y=oggi.blackout[j]


                                if (u>0 and v>0):
                                        if w>0 and x>0 and y>0:
                                                #print("oggi.Date ",oggi.Date[0])
                                                #print("nextday ",next_day," b14 ",oggi.b14," b7
",oggi.b7," b2 ",oggi.b2," b1 ",oggi.b1," blackout ", oggi.blackout)
                                                #next_day[j] = oggi.b14[j] + oggi.b7[j] + oggi.b2[j] +
oggi.b1[j] + oggi.blackout[j]
                                                next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] ,
oggi.b2[j] , oggi.b1[j] , oggi.blackout[j]] ,axis=0)
                                                print("sector 1")
                                        elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
                                                if (w>0 and x>0 ):
                                                        #next_day[j] = oggi.b14[j] + oggi.b7[j] +
oggi.b2[j] + oggi.b1[j]
                                                        next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] ,
oggi.b2[j] , oggi.b1[j] ] ,axis=0)
                                                        print("sector 2")
                                                if (w>0 and y>0):
                                                        #next_day[j] = oggi.b14[j] + oggi.b7[j] +
oggi.b2[j] + oggi.blackout[j]
                                                        next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] ,
oggi.b2[j] , oggi.blackout[j]] ,axis=0)
                                                        print("sector 3")
                                                if (x>0 and y>0):

                        #print("oggi.b14\n",oggi.b14,"oggi.b7\n",oggi.b7," oggi.b1 \n", oggi.b1,"oggi.blackout
\n",oggi.blackout)
                                                        #next_day[j] = oggi.b14[j] + oggi.b7[j] +
oggi.b1[j] + oggi.blackout[j]
                                                        next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j],
oggi.b1[j] , oggi.blackout[j]] ,axis=0)
                                                        print("sector 4")
                                        else:
                                                #next_day[j] = oggi.b14[j] + oggi.b7[j]
                                                next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j]] ,axis=0)
                                                print("sector 5")
                                elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
                                        if w>0 and x>0 and y>0:
                                                #next_day[j] =  oggi.b2[j] + oggi.b1[j] + oggi.blackout[j]
                                                next_day[j] = np.mean([ oggi.b2[j] , oggi.b1[j] ,
oggi.blackout[j]] ,axis=0)
                                                print("sector 6")
                                        elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
                                                if (w>0 and x>0 ):
                                                        next_day[j] =  oggi.b2[j] + oggi.b1[j]
                                                        next_day[j] = np.mean([  oggi.b2[j] ,
oggi.b1[j]],axis=0)
                                                        print("sector 7")
                                                if (w>0 and y>0):
                                                        #next_day[j] =  oggi.b2[j] + oggi.blackout[j]
                                                        next_day[j] = np.mean([oggi.b2[j] ,
oggi.blackout[j]] ,axis=0)
                                                        print("sector 8")
                                                if (x>0 and y>0):
```

```
                                                #next_day[j] =  oggi.b1[j] + oggi.blackout[j]
                                                next_day[j] = np.mean([  oggi.b1[j] ,
oggi.blackout[j]] ,axis=0)
                                                print("sector 9")
                                        else:
                                                next_day[j] =  oggi.blackout[j]
                                                print("sector 10")
                                        print("\npredicted: ",next_day[j], "actual:",oggi.next_blackout[j])
                                        # check and correct error for the respective hour
                                        if j==0:
                                                # use previous day blackout duration amplitude, if any
                                                if next_day[j] > 0 and prev_amp > 0:
                                                        next_day[j] = prev_amp

                                                predError_hr = oggi.next_blackout[j] - next_day[j]
                                                predError_hr = predError_hr/2
                                                # take note of previous blackout amplitude, next prediction
shouldn't exceed this
                                                prev_amp=oggi.next_blackout[j]
                                                if oggi.next_blackout[j]==0:
                                                        if next_day[j] > 0:
                                                                pred_permit = 0
                                                #if blackout has occured get amplitude and start counter for
interval to detect second blackout event
                                                elif oggi.next_blackout[j] > 0:
                                                        asd_amp1 = oggi.next_blackout[j]
                                                        # j value will also be used to mark next blackout, then
derive/deduce interval for 3rd blackout
                                                        asd_interval1 = j
                                        elif j>0:
                                                if pred_permit == 1:
                                                        #adjust predicted hr/quarter with previous quarter error
                                                        next_day[j] = next_day[j] + predError_hr
                                                        #ensure blackout prediction doesnt exceed previous
blackout value
                                                        if next_day[j] >= prev_amp:
                                                                next_day[j] = prev_amp
                                                        predError_hr = oggi.next_blackout[j] - next_day[j]
                                                        predError_hr = predError_hr/2
                                                        #allow positive prediction if maxCount has not runout
                                                        if next_day[j]>0 and maxCount>0:
                                                                maxCount-=1
                                                        elif next_day[j]>0 and maxCount==0 and prev_amp==0:
                                                                next_day[j]=0
                                                        elif next_day[j]>0 and maxCount==0 and prev_amp > 0:
                                                                next_day[j]=prev_amp
                                                        #make prediction if both 1st & 2nd blackout have occured
                                                        if asd_interval1 > 0 and asd_interval2 > 0:
                                                                #compute interval
                                                                asd_pred_interval =  asd_interval2 -
asd_interval1

                                                                #check current elapsed time, if it's time to allow
prediction

                                                                if (asd_interval2 + asd_pred_interval) == j:
                                                                        # prediction is mean of 1st and 2nd
blackout that occured prior

                                                                        next_day[j] =
round(np.mean([asd_amp1,asd_amp2]),2)

                                                                        currentPrediction = next_day[j]
                                                                        #reset/update trackers
                                                                        asd_interval1 = j-1
                                                                        asd_interval2 = asd_interval1 +
asd_pred_interval
```

```
                                                asd_amp1 = prev_amp
                                                asd_amp2 =  prev_amp
                                                print("@@@@@@@@@@@@
executed!")

                                        # take note of current blackout amplitude, next prediction
shouldn't exceed this
                                        if oggi.next_blackout[j]>0:
                                                prev_amp=oggi.next_blackout[j]
                                                pred_permit = 1
                                                if asd_interval1 == 0 and asd_interval2 == 0:
                                                        asd_interval1 = j
                                                        asd_amp1 = oggi.next_blackout[j]
                                                elif asd_interval1 > 0 and asd_interval2 == 0:
                                                        asd_interval2 = j
                                                        asd_amp2 = oggi.next_blackout[j]

                                        # revoke prediction ability if algo wrongly make a
prediction without it actually occuring
                                        if oggi.next_blackout[j]==0:
                                                if next_day[j] > 0:
                                                        pred_permit = 0
                                                        asd_interval1 = 0
                                                        asd_interval2 = 0


                                elif pred_permit == 0:
                                        next_day[j]=0 #assume no blackout
                                        # if blackout occurs, grant permission to make new
predictions, until the next incorrect prediction
                                        if oggi.next_blackout[j]>0:
                                                pred_permit = 1
                                                prev_amp=oggi.next_blackout[j]


                                #remove predictions greater than 1, or less than 0
                                if next_day[j] > maxAmplitude:
                                        next_day[j] = maxAmplitude
                                if next_day[j]<0:
                                        next_day[j]=0
                                #if next_day[j] > maxAmplitude:
                                #       next_day[j] = maxAmplitude
                        print(" \nk is now",k)
                        print("*************** progress complete:.....
",np.round((k/len(test_dates))*100,1),"% ***************************")
                        print("date is ",test_dates[k])
                        check_timer(tic)
                        #####---------- routine to estimate remaining time----------########
                        remaining_time(tic,k,len(test_dates))
                        if k == start_index:
                                print("oggi.next_blackout[j] is ",oggi.next_blackout[j]," next_day[j] is
",next_day[j])
                                bestMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                worstMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                bestMaeDate = oggi.Date[0]
                                worstMaeDate = oggi.Date[0]
                                observations = oggi.next_blackout
                                predError = oggi.next_blackout - next_day
                                next_day_list1 = next_day
                                next_day_list2 = next_day
                                maeScore_list1=list()
```

```
                                        maeScore_list2=list()
                                        mseScore_list2=list()
                                        rmseScore_list2=list()
                                        r2Score_list2=list()

        maeScore_list1.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))


        maeScore_list2.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))

        mseScore_list2.append(round(metrics.mean_squared_error(oggi.next_blackout,next_day),3))

        rmseScore_list2.append(round(math.sqrt(metrics.mean_squared_error(oggi.next_blackout,next_day)),3
))

        r2Score_list2.append(round(metrics.r2_score(oggi.next_blackout,next_day),3))
                                        #print("next_day\n",next_day)
                                        #print("prediction error\n",predError)
                                        #print("prediction error\n",predError/4)
                                        predError = predError/2
                                elif k>start_index :
                                        observations=np.hstack((observations,oggi.next_blackout))
                                        #if len(next_day)<24:
                                        #print("next_day original\n",next_day)
                                        next_day_list1=np.hstack((next_day_list1,next_day))

        maeScore_list1.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))


                                        predError_prev_day = oggi.next_blackout - next_day
                                        #next_day = next_day + predError

                                        next_day_list2=np.hstack((next_day_list2,next_day))

                                        #print("prediction error before correction\n",predError_prev_day)
                                        #print("actual DATA\n",oggi.next_blackout)
                                        #print("predicted DATA\n",next_day)
                                        #print("#k is ",k, "date is ",oggi.Date[0])


                                        if bestMae >
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3):
                                                bestMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                                bestMaeDate = oggi.Date[0]
                                        if worstMae <
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3):
                                                worstMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                                worstMaeDate = oggi.Date[0]

        maeScore_list2.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))

        mseScore_list2.append(round(metrics.mean_squared_error(oggi.next_blackout,next_day),3))

        rmseScore_list2.append(round(math.sqrt(metrics.mean_squared_error(oggi.next_blackout,next_day)),3
))

        r2Score_list2.append(round(metrics.r2_score(oggi.next_blackout,next_day),3))

                                        #print("k ",k," mae
",round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))
                                        #print("next_day error corrected\n",next_day)
```

```
                predError = oggi.next_blackout - next_day
                #print("prediction error after correction\n",predError)
                #print("prediction error\n",predError/4)
                predError = predError/2


        blk_ypred_rf = [0]*dayLength

        #print("*** input length ",len(xtest[0]))
        #print("*** xtest[j] ",np.array(xtest[0].reshape(1,10)).shape)
        #print("*** historyX ",np.array(historyX).shape)

        #check past days for max blackout events, past24hrs get all votes, other days max of
1 event ceiling
        #u=b14 v=b7 w=b2 x=b1 y=blackout z=next_blackout
        #minCount sets a 'floor' for the influence of days previous to 24hrs, their influence is
assumed low
        minCount=0
        if sum(oggi.b14)>0 or sum(oggi.b7)>0 or sum(oggi.b2)>0 or sum(oggi.b1)>0:
                minCount=1

        # now count blackout events in the past 24hrs, assume they have high influence on
the next 24hrs
        maxCount=0
        for i, element in enumerate(oggi.blackout):
                if oggi.blackout[i]>0:
                        maxCount+=1
        #if there was no blackout event, then allow only 1 positive blackout event in the next
24hrs
        if maxCount==0:
                maxCount=minCount
        # pred_permit allows predictions or suppresses them
        pred_permit=1
        rf_interval1 = 0
        rf_interval2 = 0
        prev_amp = 0

        for j in range(len(xtest)):
                activeDate = oggi.Date[0]
                #training with Random forest model
                blackoutmodel_rf.fit(historyX, historyY)
                #predict with RF model, [0] used to extract array element instead of
returning an array
                blk_ypred_rf[j] = blackoutmodel_rf.predict(xtest[j].reshape(1,
len(xtest[j])))[0]

                # check and correct error for the respective hour if prediction is still
permissible
                if j==0:
                        # predError_rf = ytest[j] - blk_ypred_rf[j]
                        # predError_rf = predError_rf/2
                        # take note of previous blackout amplitude, next prediction
shouldn't exceed this
                        prev_amp=ytest[j]
                        if ytest[j]==0:
                                if blk_ypred_rf[j] > 0:
                                        pred_permit = 0
                        #element can't be higher than maxAmplitude, for first prediction
                        if blk_ypred_rf[j]>maxAmplitude:
                                blk_ypred_rf[j]=maxAmplitude
                        #pass
                elif j>0:
                        if pred_permit == 1:
```

```
                                        if blk_ypred_rf[j] >= prev_amp:
                                                blk_ypred_rf[j] = prev_amp
                                        #allow positive prediction if maxCount has not runout
                                        if blk_ypred_rf[j]>0 and maxCount>0:
                                                maxCount-=1
                                        elif blk_ypred_rf[j]>0 and maxCount==0 and
prev_amp==0:
                                                blk_ypred_rf[j]=0
                                        elif blk_ypred_rf[j]>0 and maxCount==0 and prev_amp >
0:
                                                blk_ypred_rf[j] = prev_amp
                                        #make prediction if both 1st & 2nd blackout have occured
                                        if rf_interval1 > 0 and rf_interval2 > 0:
                                                #compute interval
                                                rf_pred_interval =  rf_interval2 - rf_interval1
                                                #check current elapsed time, if it's time to allow
prediction
                                                if (rf_interval2 + rf_pred_interval) == j:
                                                        # prediction is mean of 1st and 2nd
blackout that occured prior
                                                        blk_ypred_rf[j] =
round(np.mean([rf_amp1,rf_amp2]),2)

                                                        #reset/update trackers
                                                        rf_interval1 = j-1
                                                        rf_interval2 = rf_interval1 +
rf_pred_interval
                                                        rf_amp1 = prev_amp
                                                        rf_amp2 =  prev_amp
                                                        print("########### executed!")

                                        # take note of current blackout amplitude, next prediction
shouldn't exceed this
                                        if ytest[j] > 0:
                                                prev_amp=ytest[j]
                                                pred_permit = 1
                                                if rf_interval1 == 0 and rf_interval2 == 0:
                                                        rf_interval1 = j
                                                        rf_amp1 = ytest[j]
                                                elif rf_interval1 > 0 and rf_interval2 == 0:
                                                        rf_interval2 = j
                                                        rf_amp2 = ytest[j]
                                        if ytest[j]==0:
                                                if blk_ypred_rf[j] > 0:
                                                        pred_permit = 0
                                                        rf_interval1 = 0
                                                        rf_interval2 = 0

                                        # include routine to track blackout interval, and predict
third occurance based on the interval

                                        #adjust predicted hr with previous hr error
                                        # blk_ypred_rf[j] = blk_ypred_rf[j] + predError_rf
                                        # predError_rf = ytest[j] - blk_ypred_rf[j]
                                        # predError_rf = predError_rf/2
                                        #pass
                                elif pred_permit == 0:

                                        blk_ypred_rf[j]=0 #assume no blackout

                                        # if blackout occurs, grant permission to make new
predictions, until the next incorrect prediction
                                        if ytest[j] > 0:
```

```
                                        pred_permit = 1
                                        prev_amp=ytest[j]
                                actualPrediction = blk_ypred_rf[j]
                                actualBlackout = ytest[j]
                                #element can't be higher than 1
                                if blk_ypred_rf[j] > maxAmplitude:
                                        blk_ypred_rf[j] = maxAmplitude
                                #no element permited below zero
                                if blk_ypred_rf[j] < 0:
                                        blk_ypred_rf[j] = 0



                                #add actual observations to history for next loop
                                historyX=np.vstack((historyX,xtest[j]))
                                historyY=np.hstack((historyY,ytest[j]))

                        # ensure no element is greater than 1 or less than 0 or greater than sliding window
maxAmplitude



                                #mae_rf = round(metrics.mean_absolute_error(ytest,blk_ypred_rf),3)
                                mae_rf = round(metrics.mean_absolute_error(ytest,blk_ypred_rf),3)
                                mse_rf = round(metrics.mean_squared_error(ytest,blk_ypred_rf),3)
                                r2_rf = round(metrics.r2_score(ytest,blk_ypred_rf),3)
                                mape_rf =
round(metrics.mean_absolute_percentage_error(ytest,blk_ypred_rf),3)


                                #store all predictions in a list
                                #predictions.append(y_pred)
                                blackout_pred_rf=np.hstack((blackout_pred_rf,blk_ypred_rf[j]))
                                blackout_actual=np.hstack((blackout_actual,ytest[j]))
                                #observations.append(ytest)
                                #print("ytest is ",ytest)
                                #print("xtest is ",xtest)
                                pastday=historyY[len(historyY)-24:len(historyY)]

                                if k==start_index:
                                        pastdayLog = pastday

                                else:
                                        #print("°°° predictions_lstm1 is now
\n",predictions_lstm1,"\nypred_lstm1 is \n",predictions_lstm1)
                                        pastdayLog = np.hstack((pastdayLog,pastday))

                                #matokeo=round(model_svr.score(xtest,ytest),1)

                                maeScore_rf.append(mae_rf)
                                mseScore_rf.append(mse_rf)
                                r2Score_rf.append(r2_rf)
                                mapeScore_rf.append(mape_rf)


                        #print("SVR efficiency mae: ",mae_svr)


                                #print("RFefficiency mae: ",mae_rf)
                                #print("past day len",len(pastday))
                                #print("k is ",k)
                                mae_asd =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
```

```
                                        RF_ASD =[pd.DataFrame(next_day),pd.DataFrame(blk_ypred_rf)]
                                        RF_ASD=pd.concat(RF_ASD).groupby(level=0).mean()
                                        RF_ASD=RF_ASD.round(1)
                                        RF_ASD=np.array(RF_ASD.values)
                                        maeRFasd =
round(metrics.mean_absolute_error(oggi.next_blackout,RF_ASD),3)

                                if oggi.Date[0] == "2021-04-01" and blocker==1:
                                        worstday_15min = next_day

        worstday_15min=pd.DataFrame(worstday_15min,columns=['worstday_15min'])
                                        worstday_15min.to_pickle("./worstday_15min.pkl")
                                        dpi = 600
                                        px = 1/plt.rcParams['figure.dpi']
                                        #plt.figure(figsize=(10, 4))
                                        plt.figure(figsize=(dpi*px, dpi*px))
                                        plt.plot( next_day, color ='b')
                                        plt.plot( blk_ypred_rf, color ='r')
                                        plt.plot( RF_ASD, color ='g')
                                        plt.plot( oggi.next_blackout, color ='k',marker='o')
                                        plt.plot( oggi.blackout, color ='b',linestyle = ':')
                                        plt.xlabel('Time[hr]')
                                        plt.xticks(np.arange(0,24,2))
                                        plt.ylabel('blackout index')
                                        plt.title('blackout forecast' )
                                        plt.legend(['ASD mae: '+str(mae_asd),'Random Forest mae:
'+str(mae_rf),'RF-ASD mae: '+str(maeRFasd),'actual','yesterday'])
                                        plt.tight_layout()
                                        plt.show()
                                        #plt.savefig('img/blackout_forecast.png')
                                        plt.close()
                                        blocker=1

                                '"if bestMaeDate == "2021-10-13" and blocker==1:
                                        dpi = 600
                                        px = 1/plt.rcParams['figure.dpi']
                                        #plt.figure(figsize=(10, 4))
                                        plt.figure(figsize=(dpi*px, dpi*px))
                                        plt.plot( next_day, color ='b')
                                        plt.plot( blk_ypred_rf, color ='r')
                                        plt.plot( RF_ASD, color ='g')
                                        plt.plot( ytest, color ='k',marker='o')
                                        plt.plot( pastday, color ='b',linestyle = ':')
                                        plt.xlabel('Time')
                                        plt.xticks(np.arange(0,24,2))
                                        plt.ylabel('blackout index')
                                        plt.title('blackout forecast bestMaeDate'+oggi.Date[0])
                                        plt.legend(['ASD mae: '+str(mae_asd),'Random Forest mae:
'+str(mae_rf),'RF-ASD mae: '+str(maeRFasd),'actual','yesterday'])
                                        plt.tight_layout()
                                        plt.show()
                                        #plt.savefig('img/blackout_forecast.png')
                                        plt.close()
                                        blocker=2"'


        print("bestMaeDate ",bestMaeDate," bestMae ",bestMae,
        "\nworstMaeDate ",worstMaeDate,"worstMae ",worstMae)

        # #make prediction for nextday i.e current day and save
        # xtest_today=allDays.loc[allDays.Date==all_dates[-1] ]
        # #print("xtest_today length is :",len(xtest_today))
        # #print("xtest_today length is :",xtest_today)
```

```
        # xtest_today.reset_index(drop=True, inplace=True)
        # #select relevant columns only
        # xtest_today = xtest_today[desired_features]
        # #print("xtest_today \n",xtest_today)
        # d1=allDays.loc[allDays.Date==all_dates[-2]].blackout
        # d1.reset_index(drop=True, inplace=True)
        # d2=allDays.loc[allDays.Date==all_dates[-3]].blackout
        # d2.reset_index(drop=True, inplace=True)
        # #print("d2 \n",d2)
        # d3=allDays.loc[allDays.Date==all_dates[-4]].blackout
        # d3.reset_index(drop=True, inplace=True)
        # #print("d3 \n",d3)
        # d4=allDays.loc[allDays.Date==all_dates[-5]].blackout
        # d4.reset_index(drop=True, inplace=True)
        # #print("d4 \n",d4)
        # d5=allDays.loc[allDays.Date==all_dates[-6]].blackout
        # d5.reset_index(drop=True, inplace=True)
        # #print("d5 \n",d5)
        # #print("d5 full data\n",allDays.loc[allDays.Date==all_dates[-6]])
        # d6=allDays.loc[allDays.Date==all_dates[-7]].blackout
        # d6.reset_index(drop=True, inplace=True)
        # #print("d6 \n",d6)
        # #print("d6 full data\n",allDays.loc[allDays.Date==all_dates[-7]])
        # d7=allDays.loc[allDays.Date==all_dates[-8]].blackout
        # d7.reset_index(drop=True, inplace=True)
        # #print("d7 \n",d7)
        # #print("d7 full data\n",allDays.loc[allDays.Date==all_dates[-8]])

        # try:
        #     xtest_today =
xtest_today.assign(b7=d7.values,b6=d6.values,b5=d5.values,b4=d4.values,b3=d3.values,b2=d2.values,b1=d1.v
alues)
        # except Exception as error:
        #     print("error has occured!")
        #           #msg =msg+ str(ct)+" An exception occurred: {}".format(traceback.format_exc())



        #print("final dataset \n",xtest_today)

        #xtest=xtest_today.values[:,1:]
        #print("xtest dataset \n",xtest)
        #make prediction once a day
        #blk_ypred_svr = blackoutmodel_svr.predict(xtest)
        #blk_ypred_rf = blackoutmodel_rf.predict(xtest)
        #add predicted with yesterday value
        #get mean of predicted data and past 2 days data

        ypred_mean1 =[pd.DataFrame(next_day_list2),pd.DataFrame(blackout_pred_rf)]
        ypred_mean1=pd.concat(ypred_mean1).groupby(level=0).mean()
        ypred_mean1=ypred_mean1.round(1)
        ypred_mean1=np.array(ypred_mean1.values)
        mae_asd_rf = round(metrics.mean_absolute_error(blackout_actual,ypred_mean1),3)
        rmse_asd_rf = round(math.sqrt(metrics.mean_squared_error(blackout_actual,ypred_mean1)),3)
        mse_asd_rf = round(metrics.mean_squared_error(blackout_actual,ypred_mean1),3)
        r2_asd_rf = round(metrics.r2_score(blackout_actual,ypred_mean1),3)


        #get only mae & rmse for instances where blackout occured only
        true_blackouts =pd.concat([pd.DataFrame(blackout_actual,columns =
['blackout_actual']),pd.DataFrame(next_day_list2,columns = ['ASD']),pd.DataFrame(blackout_pred_rf,columns
= ['RF']),pd.DataFrame(ypred_mean1,columns = ['RF_ASD'])],axis=1)
        print("true blackouts\n",true_blackouts)
```

```
        true_blackouts = true_blackouts.loc[true_blackouts.blackout_actual > 0]
        print("true blackouts after\n",true_blackouts)
        print("ASD mae:
",round(metrics.mean_absolute_error(true_blackouts.blackout_actual.values,true_blackouts.ASD.values),3)," ASD rmse:
",round(math.sqrt(metrics.mean_squared_error(true_blackouts.blackout_actual.values,true_blackouts.ASD.values)),3))
        print("RF mae:
",round(metrics.mean_absolute_error(true_blackouts.blackout_actual.values,true_blackouts.RF.values),3), " RF rmse:
",round(math.sqrt(metrics.mean_squared_error(true_blackouts.blackout_actual.values,true_blackouts.RF.values)),3))
        print("RF-ASD mae:
",round(metrics.mean_absolute_error(true_blackouts.blackout_actual.values,true_blackouts.RF_ASD.values),3)," RF-ASD rmse:
",round(math.sqrt(metrics.mean_squared_error(true_blackouts.blackout_actual.values,true_blackouts.RF_ASD.values)),3))
        beeper()
        exit()
        #blackoutForecast=np.round(max(ypred_mean)*100,1)

        #print("blackout_actual length:",len(blackout_actual)," days: ",len(blackout_actual)/24)

        mean_list2_mae =round(np.mean(maeScore_list2),3)
        mean_list2_rmse =round(np.mean(rmseScore_list2),3)
        mean_list2_mse =round(np.mean(mseScore_list2),3)
        mean_list2_rmse =round(np.mean(rmseScore_list2),3)
        mean_list2_r2score =round(np.mean(r2Score_list2),3)

        mean_rf_mae =round(np.mean(maeScore_rf),3)
        mean_rf_rmse =round(math.sqrt(np.mean(mseScore_rf)),3)
        mean_rf_mse =round(np.mean(mseScore_rf),3)
        mean_rf_r2 =round(np.mean(r2Score_rf),3)

        print("RF: mae(",mean_rf_mae,") rmse(",mean_rf_rmse,") mse(",mean_rf_mse,") r2(",mean_rf_r2,")")
        print("ASD: mae(",mean_list2_mae,") rmse(",mean_list2_rmse,") mse(",mean_list2_mse,") r2(",mean_list2_r2score,")")
        print("RF-ASD: mae(",mae_asd_rf,") rmse(",rmse_asd_rf,") mse(",mse_asd_rf,") r2(",r2_asd_rf,")")

        print("first date ",x_test['Date'])
        #print("last date ",x_test['Date'][len(x_test)-1])

        check_timer(tic)
        #make audible tone
        beeper()
        plt.close()
        dpi = 1000
        px = 1/plt.rcParams['figure.dpi']
        #plt.figure(figsize=(10, 4))
        fig = plt.figure(figsize=(dpi*px, dpi*px))
        #plt.figure(figsize=(dpi*px, dpi*px))
        ax = plt.subplot(111)
        #plt.plot( next_day_list1, color ='b',marker='o')
        ax.plot( next_day_list2, color ='g',marker='s',lw=3)
        ax.plot( ypred_mean1, color ='r',marker='o',linestyle = '-.')
        ax.plot( blackout_pred_rf, color ='b',marker='P',linestyle = '--')
        ax.plot( blackout_actual, color ='k',marker='o')
        #ax.plot( pastdayLog, color ='r',marker='.',linestyle = ':')
        #plt.plot( next_day_list2, color ='m',marker='.')
        #plt.plot( ypred_mean1, color ='g',marker='o',linestyle = '-.')
        #plt.plot( blackout_pred_rf, color ='b',linestyle = '--')
        #plt.plot( blackout_actual, color ='k',marker='o')
        #plt.plot( pastdayLog, color ='r',marker='.',linestyle = ':')
```

```
        #plt.axvline(x=24,  color='k', linestyle='--')
        #plt.text(24,1,'d1 ',horizontalalignment='right')
        xposition = [(dayLength*x)-1 for x in range(1,(int(len(blackout_actual)/dayLength))+1,1)]
        for xc in xposition:
                plt.axvline(x=xc, color='k', linestyle=':',ymin=0, ymax=1)
                plt.text(xc,1,'d'+str(int(np.ceil(xc/dayLength)))+' ',horizontalalignment='right')
        plt.xlabel('Time [Hr]')
        plt.ylabel('Blackout Index')
        plt.title('Blackout Forecast')
        plt.ylim(-0.03, 1.05)
        #plt.legend(['SVR blackout','RF blackout','actual','yesterday'])
#       plt.legend([
#               #'SD1 (mae:'+str(mean_list1_mae)+')',
#       'ASD (mae:'+str(mean_list2_mae)+', rmse:'+str(mean_list2_rmse)+')',
#       'ASD-RF (mae: '+str(mae_asd_rf)+', rmse:'+str(rmse_asd_rf)+')',
#       'RF (mae: '+str(mean_rf_mae)+', rmse:'+str(mean_rf_rmse)+')',
#       'actual','previous day'],loc='upper center',ncol=2)
        ax.legend([
                #'SD1 (mae:'+str(mean_list1_mae)+')',
        'ASD (mae:'+str(mean_list2_mae)+')',
        'ASD-RF (mae: '+str(mae_asd_rf)+')',
        'RF (mae: '+str(mean_rf_mae)+')',

        'actual'],loc='upper center',bbox_to_anchor=(0.9, 0.95),ncol=1)
        #'actual','previous day'],loc='upper center',bbox_to_anchor=(0.5, -0.08),ncol=5)
        #matplotlib.rcParams['legend.fontsize'] = 20
        #plt.tight_layout()
        plt.show()

def blackout_15min_classifier():
        tic = time.perf_counter()
        #allDays = convert_to_15min()
        #check_timer(tic)

        allDays = pd.read_pickle("./allDays15min_filled.pkl")
        #allDays = pd.read_pickle("./allDays15min.pkl")

        # fetch all dates again, this now includes valid dataframes
        all_dates=allDays['Date'].unique()
        #print("last day is ",len(allDays.loc[allDays['Date'] == allDays.iloc[-1].Date] )," records long!")
        #print("allDays\n",len(allDays.loc[allDays['Date'] == all_dates[-1]]))
#       print("last day \n",allDays.loc[allDays['Date'] == allDays.iloc[-
1].Date,('Date','hour','minute')].to_string(index=False) )
        #print("allDays\n",allDays)
        print("final all_dates \n",allDays['Date'].unique())
        print("total dataset days \n",len(allDays['Date'].unique()))


        trainData=allDays.loc[allDays.Date<='2021-03-31']
        testData=allDays.loc[allDays.Date >'2021-03-31']
        #print("trainData\n",trainData)
        #print("testData\n",testData)

        #------------ BLACKOUT PREDICTIONS ----------------------

        # BLACKOUT MODEL TRAINING ---------------------------------
        next_blackout=allDays.loc[allDays.Date>=all_dates[14] ].blackout
        #print("next_blackout startdate ",all_dates[14],"currentDay startdate ",all_dates[13])
        #next_blackout=allDays.loc[allDays.Date >= all_dates[14] ]
        #print("next_blackout\n",next_blackout)
        #next_blackout = next_blackout['Date','blackout']
        #print("start_date length is :",len(next_blackout))
        #print("start day index ",next_blackout.index.values[0])
```

```
	currentDay=allDays.loc[allDays['Date'] >= all_dates[13],: ]
	#print("currentDay\n",currentDay)
	#print("final all_dates \n",all_dates[13] )

	#print("day2 length is :",len(currentDay))
	currentDay.reset_index(drop=True, inplace=True)
	currentDay=currentDay.loc[0:len(next_blackout)-1]
	#print("tomorrow is ",all_dates[14],
	#"\nsimilar day 14 days back : ",(datetime.datetime.strptime(all_dates[14], '%Y-%m-%d')-
datetime.timedelta(days=14)).strftime('%Y-%m-%d'),
	#"\n the is also :",all_dates[0])

	#select relevant columns only
	desired_features = ['Date','weekday','hour','day','ac_voltage','frequency','blackout']
	currentDay = currentDay[desired_features]
	d1=allDays.loc[allDays.Date>=all_dates[12]].blackout
	d1.reset_index(drop=True, inplace=True)
	d1=d1.loc[0:len(next_blackout)-1]
	d2=allDays.loc[allDays.Date>=all_dates[11]].blackout
	d2.reset_index(drop=True, inplace=True)
	d2=d2.loc[0:len(next_blackout)-1]
	d7=allDays.loc[allDays.Date>=all_dates[7]].blackout
	d7.reset_index(drop=True, inplace=True)
	d7=d7.loc[0:len(next_blackout)-1]
	d14=allDays.loc[allDays.Date>=all_dates[0]].blackout
	d14.reset_index(drop=True, inplace=True)
	d14=d14.loc[0:len(next_blackout)-1]

	currentDay =
currentDay.assign(b14=d14.values,b7=d7.values,b2=d2.values,b1=d1.values,next_blackout=next_blackout.valu
es)
	currentDay =
currentDay[['Date','weekday','hour','day','ac_voltage','frequency','b14','b7','b2','b1','blackout','next_blackout']]
	#print("final dataset \n",currentDay.loc[0:47])
	#print("final dataset \n",currentDay)
	#print("final dataset \n",currentDay.info())
	#//todo ensure all blackout values have only binary values, and not continuous values
	currentDay.loc[(currentDay.blackout > 0),('blackout')] = 1
	currentDay.loc[(currentDay.b14 > 0),('b14')] = 1
	currentDay.loc[(currentDay.b7 > 0),('b7')] = 1
	currentDay.loc[(currentDay.b2 > 0),('b2')] = 1
	currentDay.loc[(currentDay.b1 > 0),('b1')] = 1
	currentDay.loc[(currentDay.next_blackout > 0),('next_blackout')] = 1




	predictorNames=currentDay.columns.values[1:-1]
	#print("predictors: ",predictorNames)
	#print("predictors length: ",len(predictorNames))
	#x=currentDay[predictorNames].values
	#y=currentDay['next_blackout'].values

	#convert all blackout events into binary
	print("dataset\n",currentDay)


	# ------------ train and test data split ---------------
	train_dates=currentDay['Date'].unique()
	#print("train dates\n",train_dates)

	#print("currentDay\n",len(currentDay))
	#print("currentDay\n",currentDay)
```

```
#get train 80% and 20% test
#cutoff=int(np.floor(len(train_dates)*0.8))
cutoff=int(np.floor(len(train_dates)*0.2))
cutoff=str(train_dates[cutoff])
#cutoff="2021-03-01"
print("cutoff date: ",cutoff)
x_train=currentDay[currentDay['Date']<cutoff]

x_train=x_train[predictorNames]
#x_train.to_csv("./matlabData/x_train.csv")
x_train=x_train.values
y_train=currentDay[currentDay['Date']<cutoff]
y_train=y_train['next_blackout']
#y_train.to_csv("./matlabData/y_train.csv")
y_train=y_train.values

#x_test=currentDay[(currentDay['Date']>=cutoff) & (currentDay['Date']< "2021-04-01")]
#x_test=currentDay[(currentDay['Date']>"2021-10-03") & (currentDay['Date']<="2021-04-01")]
x_test=currentDay[(currentDay['Date']>=cutoff) ]
#x_test.to_csv("./matlabData/x_test.csv")
#print("x_test\n",x_test[0:100].to_string(index=False))
print("x_test\n",x_test['Date'].unique())



#x_test=x_test[predictorNames].values
#x_test=x_test[predictorNames]
y_test=currentDay[currentDay['Date']>=cutoff]
y_test=y_test[{'next_blackout','Date'}]
#print("x_train is\n",x_train)
#print("y_train is\n",y_train)
#print("x_test is\n",x_test)
#print("y_test is\n",y_test)


blackout_pred_rf=[]
blackout_actual=[]
#observations=list()
maeScore_rf=list()
mapeScore_rf=list()
mseScore_rf=list()
r2Score_rf=list()



historyX=[x for x in x_train]
historyY=[y for y in y_train]


#//todo Random forest classifier model instead of regressor
#blackoutmodel_rf = RandomForestRegressor(n_estimators=10)
blackout_rf_classifier = RandomForestClassifier(n_estimators=500,max_depth=2,random_state=0)
#forward-walk-validation
test_dates=x_test['Date'].unique()
#test_dates.to_csv("./matlabData/test_dates.csv")
#print("all dates tail\n",test_dates)
#print("y_test\n",y_test)
#model_svr.fit(x_train, y_train)

#ytest=ytest.values
# oggi = currentDay.loc[currentDay.Date=="2021-09-14"]

blocker=0
```

```
	dayLength = 24*4 #each hour has 4 quarters
	start_index = 0
	prev_amp = 0
	for k in range(start_index,len(test_dates)):
	#for k in range(2):
#	for k in range(len(test_dates)):
				print("\nOuter loop is ",k)
				print("*************** progress complete:.....
",np.round((k/len(test_dates))*100,1),"% ***************************")
				print("date is ",test_dates[k])
				print("length of test_dates is ",len(test_dates))
				xtest=x_test[x_test['Date']==test_dates[k]]

				xtest.reset_index(drop=True, inplace=True)

				print("*** Length of day : ",len(xtest))

				#make sure you only fetch a day 24hrs long, if longer trim it to only 24hrs
				# length will be 96 for a 15min interval day
				if len(xtest)>dayLength:
					xtest = xtest.loc[0:dayLength-1]

				ytest=xtest['next_blackout'].values
				xtest=xtest.values[:,1:-1]


				#print("length of xtest is ",len(xtest),"\nxtest is\n",xtest)

				oggi = currentDay.loc[currentDay.Date==test_dates[k]]
				oggi.reset_index(drop=True, inplace=True)
				if len(oggi)>dayLength:
					oggi = oggi.loc[0:dayLength-1]

				minCount=0
				if sum(oggi.b14)>0 or sum(oggi.b7)>0 or sum(oggi.b2)>0 or sum(oggi.b1)>0:
					minCount=1

				# now count blackout events in the past 24hrs, assume they have high influence on
the next 24hrs
				maxCount=0
				for i, element in enumerate(oggi.blackout):
					if oggi.blackout[i]>0:
						maxCount+=1
				#if there was no blackout event, then allow only 1 positive blackout event in the next
24hrs
				if maxCount==0:
					maxCount=minCount

				# check max amplitude (blackout duration) in the past 2weeks, assume this to be the
max permissible prediction amplitude for next 24hrs prediction
				maxAmplitude=max(oggi.b14)
				if maxAmplitude < max(oggi.b7):
					maxAmplitude=max(oggi.b7)
				if maxAmplitude < max(oggi.b2):
					maxAmplitude=max(oggi.b2)
				if maxAmplitude < max(oggi.b1):
					maxAmplitude=max(oggi.b1)
				if maxAmplitude < max(oggi.blackout):
					maxAmplitude=max(oggi.blackout)
				maxAmplitude =
statistics.mean([max(oggi.b14),max(oggi.b7),max(oggi.b2),max(oggi.b1),max(oggi.blackout)])
				# pred_permit allows predictions or suppresses them
```

```
pred_permit=1
asd_interval1 = 0
asd_interval2 = 0
next_day = [0]*dayLength
# make hourly prediction for current day before next outer incremented day loop
for j in range(len(oggi)):
	print("\ncurrent minute quarter is ",j)
	#u=b14 v=b7 w=b2 x=b1 y=blackout z=next_blackout
	u=oggi.b14[j]
	v=oggi.b7[j]
	w=oggi.b2[j]
	x=oggi.b1[j]
	y=oggi.blackout[j]


	if (u>0 and v>0):
		if w>0 and x>0 and y>0:
			#print("oggi.Date ",oggi.Date[0])
			#print("nextday ",next_day," b14 ",oggi.b14," b7 ",oggi.b7," b2 ",oggi.b2," b1 ",oggi.b1," blackout ", oggi.blackout)
			#next_day[j] = oggi.b14[j] + oggi.b7[j] + oggi.b2[j] + oggi.b1[j] + oggi.blackout[j]
			next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] , oggi.b2[j] , oggi.b1[j] , oggi.blackout[j]] ,axis=0)
			print("sector 1")
		elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
			if (w>0 and x>0 ):
				#next_day[j] = oggi.b14[j] + oggi.b7[j] + oggi.b2[j] + oggi.b1[j]
				next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] , oggi.b2[j] , oggi.b1[j] ] ,axis=0)
				print("sector 2")
			if (w>0 and y>0):
				#next_day[j] = oggi.b14[j] + oggi.b7[j] + oggi.b2[j] + oggi.blackout[j]
				next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j] , oggi.b2[j] , oggi.blackout[j]] ,axis=0)
				print("sector 3")
			if (x>0 and y>0):

	#print("oggi.b14\n",oggi.b14,"oggi.b7\n",oggi.b7," oggi.b1 \n", oggi.b1,"oggi.blackout \n",oggi.blackout)
				#next_day[j] = oggi.b14[j] + oggi.b7[j] + oggi.b1[j] + oggi.blackout[j]
				next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j], oggi.b1[j] , oggi.blackout[j]] ,axis=0)
				print("sector 4")
		else:
			#next_day[j] = oggi.b14[j] + oggi.b7[j]
			next_day[j] = np.mean([ oggi.b14[j] , oggi.b7[j]] ,axis=0)
			print("sector 5")
	elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
		if w>0 and x>0 and y>0:
			#next_day[j] =  oggi.b2[j] + oggi.b1[j] + oggi.blackout[j]
			next_day[j] = np.mean([ oggi.b2[j] , oggi.b1[j] , oggi.blackout[j]] ,axis=0)
			print("sector 6")
		elif (w>0 and x>0 )or (w>0 and y>0) or (x>0 and y>0):
			if (w>0 and x>0 ):
				next_day[j] =  oggi.b2[j] + oggi.b1[j]
				next_day[j] = np.mean([  oggi.b2[j] , oggi.b1[j]],axis=0)
				print("sector 7")
```

```
                                        if (w>0 and y>0):
                                                #next_day[j] =  oggi.b2[j] + oggi.blackout[j]
                                                next_day[j] = np.mean([oggi.b2[j] ,
oggi.blackout[j]] ,axis=0)
                                                print("sector 8")
                                        if (x>0 and y>0):
                                                #next_day[j] =  oggi.b1[j] + oggi.blackout[j]
                                                next_day[j] = np.mean([  oggi.b1[j] ,
oggi.blackout[j]] ,axis=0)
                                                print("sector 9")
                                else:
                                        next_day[j] =  oggi.blackout[j]
                                        print("sector 10")
                                print("\npredicted: ",next_day[j], "actual:",oggi.next_blackout[j])
                                # check and correct error for the respective hour
                                if j==0:
                                        # use previous day blackout duration amplitude, if any
                                        if next_day[j] > 0 and prev_amp > 0:
                                                next_day[j] = prev_amp

                                        predError_hr = oggi.next_blackout[j] - next_day[j]
                                        predError_hr = predError_hr/2
                                        # take note of previous blackout amplitude, next prediction
shouldn't exceed this
                                        prev_amp=oggi.next_blackout[j]
                                        if oggi.next_blackout[j]==0:
                                                if next_day[j] > 0:
                                                        pred_permit = 0
                                        #if blackout has occured get amplitude and start counter for
interval to detect second blackout event
                                        elif oggi.next_blackout[j] > 0:
                                                asd_amp1 = oggi.next_blackout[j]
                                                # j value will also be used to mark next blackout, then
derive/deduce interval for 3rd blackout
                                                asd_interval1 = j
                                elif j>0:
                                        if pred_permit == 1:
                                                #adjust predicted hr/quarter with previous quarter error
                                                next_day[j] = next_day[j] + predError_hr
                                                #ensure blackout prediction doesnt exceed previous
blackout value
                                                if next_day[j] >= prev_amp:
                                                        next_day[j] = prev_amp
                                                predError_hr = oggi.next_blackout[j] - next_day[j]
                                                predError_hr = predError_hr/2
                                                #allow positive prediction if maxCount has not runout
                                                if next_day[j]>0 and maxCount>0:
                                                        maxCount-=1
                                                elif next_day[j]>0 and maxCount==0 and prev_amp==0:
                                                        next_day[j]=0
                                                elif next_day[j]>0 and maxCount==0 and prev_amp > 0:
                                                        next_day[j]=prev_amp
                                                #make prediction if both 1st & 2nd blackout have occured
                                                if asd_interval1 > 0 and asd_interval2 > 0:
                                                        #compute interval
                                                        asd_pred_interval =  asd_interval2 -
asd_interval1

                                                        #check current elapsed time, if it's time to allow
prediction

                                                        if (asd_interval2 + asd_pred_interval) == j:
                                                                # prediction is mean of 1st and 2nd
blackout that occured prior
```

```
                                                next_day[j] =
round(np.mean([asd_amp1,asd_amp2]),2)
                                                currentPrediction = next_day[j]
                                                #reset/update trackers
                                                asd_interval1 = j-1
                                                asd_interval2 = asd_interval1 +
asd_pred_interval
                                                asd_amp1 = prev_amp
                                                asd_amp2 =  prev_amp
                                                print("@@@@@@@@@@@@
executed!")

                                        # take note of current blackout amplitude, next prediction
shouldn't exceed this
                                        if oggi.next_blackout[j]>0:
                                                prev_amp=oggi.next_blackout[j]
                                                pred_permit = 1
                                                if asd_interval1 == 0 and asd_interval2 == 0:
                                                        asd_interval1 = j
                                                        asd_amp1 = oggi.next_blackout[j]
                                                elif asd_interval1 > 0 and asd_interval2 == 0:
                                                        asd_interval2 = j
                                                        asd_amp2 = oggi.next_blackout[j]

                                        # revoke prediction ability if algo wrongly make a
prediction without it actually occuring
                                        if oggi.next_blackout[j]==0:
                                                if next_day[j] > 0:
                                                        pred_permit = 0
                                                        asd_interval1 = 0
                                                        asd_interval2 = 0


                                elif pred_permit == 0:
                                        next_day[j]=0 #assume no blackout
                                        # if blackout occurs, grant permission to make new
predictions, until the next incorrect prediction
                                        if oggi.next_blackout[j]>0:
                                                pred_permit = 1
                                                prev_amp=oggi.next_blackout[j]


                                #force all predictions greater than 0 to 1
                                if next_day[j] > 0:
                                        next_day[j] = 1
                                if next_day[j]<0:
                                        next_day[j]=0
                                #if next_day[j] > maxAmplitude:
                                #       next_day[j] = maxAmplitude
                        print(" \nk is now",k)
                        print("*************** progress complete:.....
",np.round((k/len(test_dates))*100,1),"% **************************")
                        print("date is ",test_dates[k])
                        check_timer(tic)
                        #####---------- routine to estimate remaining time----------########
                        remaining_time(tic,k,len(test_dates))
                        if k == start_index:
                                print("oggi.next_blackout[j] is ",oggi.next_blackout[j]," next_day[j] is
",next_day[j])
                                bestMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                worstMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
```

```
                                        bestMaeDate = oggi.Date[0]
                                        worstMaeDate = oggi.Date[0]
                                        observations = oggi.next_blackout
                                        predError = oggi.next_blackout - next_day
                                        next_day_list1 = next_day
                                        next_day_list2 = next_day
                                        maeScore_list1=list()
                                        maeScore_list2=list()
                                        mseScore_list2=list()
                                        rmseScore_list2=list()
                                        r2Score_list2=list()

        maeScore_list1.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))


        maeScore_list2.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))

        mseScore_list2.append(round(metrics.mean_squared_error(oggi.next_blackout,next_day),3))

        rmseScore_list2.append(round(math.sqrt(metrics.mean_squared_error(oggi.next_blackout,next_day)),3
))

        r2Score_list2.append(round(metrics.r2_score(oggi.next_blackout,next_day),3))
                                        #print("next_day\n",next_day)
                                        #print("prediction error\n",predError)
                                        #print("prediction error\n",predError/4)
                                        predError = predError/2
                                elif k>start_index :
                                        observations=np.hstack((observations,oggi.next_blackout))
                                        #if len(next_day)<24:
                                        #print("next_day original\n",next_day)
                                        next_day_list1=np.hstack((next_day_list1,next_day))

        maeScore_list1.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))


                                        predError_prev_day = oggi.next_blackout - next_day
                                        #next_day = next_day + predError

                                        next_day_list2=np.hstack((next_day_list2,next_day))

                                        #print("prediction error before correction\n",predError_prev_day)
                                        #print("actual DATA\n",oggi.next_blackout)
                                        #print("predicted DATA\n",next_day)
                                        #print("#k is ",k, "date is ",oggi.Date[0])


                                        if bestMae >
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3):
                                                bestMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                                bestMaeDate = oggi.Date[0]
                                        if worstMae <
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3):
                                                worstMae =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                                worstMaeDate = oggi.Date[0]

        maeScore_list2.append(round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))

        mseScore_list2.append(round(metrics.mean_squared_error(oggi.next_blackout,next_day),3))
```

```
	rmseScore_list2.append(round(math.sqrt(metrics.mean_squared_error(oggi.next_blackout,next_day)),3
))

	r2Score_list2.append(round(metrics.r2_score(oggi.next_blackout,next_day),3))

				#print("k ",k," mae
",round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3))
				#print("next_day error corrected\n",next_day)
				predError = oggi.next_blackout - next_day
				#print("prediction error after correction\n",predError)
				#print("prediction error\n",predError/4)
				predError = predError/2


			blk_ypred_rf = [0]*dayLength

			#print("*** input length ",len(xtest[0]))
			#print("*** xtest[j] ",np.array(xtest[0].reshape(1,10)).shape)
			#print("*** historyX ",np.array(historyX).shape)

			#check past days for max blackout events, past24hrs get all votes, other days max of
1 event ceiling
			#u=b14 v=b7 w=b2 x=b1 y=blackout z=next_blackout
			#minCount sets a 'floor' for the influence of days previous to 24hrs, their influence is
assumed low
			minCount=0
			if sum(oggi.b14)>0 or sum(oggi.b7)>0 or sum(oggi.b2)>0 or sum(oggi.b1)>0:
				minCount=1

			# now count blackout events in the past 24hrs, assume they have high influence on
the next 24hrs
			maxCount=0
			for i, element in enumerate(oggi.blackout):
				if oggi.blackout[i]>0:
					maxCount+=1
			#if there was no blackout event, then allow only 1 positive blackout event in the next
24hrs
			if maxCount==0:
				maxCount=minCount
			# pred_permit allows predictions or suppresses them
			pred_permit=1
			rf_interval1 = 0
			rf_interval2 = 0
			prev_amp = 0

			for j in range(len(xtest)):
				activeDate = oggi.Date[0]
				#training with Random forest classifier model
				blackout_rf_classifier.fit(historyX, historyY)
				#predict with RF model, [0] used to extract array element instead of
returning an array
				blk_ypred_rf[j] = blackout_rf_classifier.predict(xtest[j].reshape(1,
len(xtest[j])))[0]

				# check and correct error for the respective hour if prediction is still
permissible
				if j==0:
					# predError_rf = ytest[j] - blk_ypred_rf[j]
					# predError_rf = predError_rf/2
					# take note of previous blackout amplitude, next prediction
shouldn't exceed this
					prev_amp=ytest[j]
```

```
                                        if ytest[j]==0:
                                                if blk_ypred_rf[j] > 0:
                                                        pred_permit = 0
                                        #element can't be higher than maxAmplitude, for first prediction
                                        if blk_ypred_rf[j]>maxAmplitude:
                                                blk_ypred_rf[j]=maxAmplitude
                                        #pass
                                elif j>0:
                                        if pred_permit == 1:
                                                if blk_ypred_rf[j] >= prev_amp:
                                                        blk_ypred_rf[j] = prev_amp
                                                #allow positive prediction if maxCount has not runout
                                                if blk_ypred_rf[j]>0 and maxCount>0:
                                                        maxCount-=1
                                                elif blk_ypred_rf[j]>0 and maxCount==0 and
prev_amp==0:
                                                        blk_ypred_rf[j]=0
                                                elif blk_ypred_rf[j]>0 and maxCount==0 and prev_amp >
0:
                                                        blk_ypred_rf[j] = prev_amp
                                                #make prediction if both 1st & 2nd blackout have occured
                                                if rf_interval1 > 0 and rf_interval2 > 0:
                                                        #compute interval
                                                        rf_pred_interval =  rf_interval2 - rf_interval1
                                                        #check current elapsed time, if it's time to allow
prediction
                                                        if (rf_interval2 + rf_pred_interval) == j:
                                                                # prediction is mean of 1st and 2nd
blackout that occured prior
                                                                blk_ypred_rf[j] =
round(np.mean([rf_amp1,rf_amp2]),2)

                                                        #reset/update trackers
                                                        rf_interval1 = j-1
                                                        rf_interval2 = rf_interval1 +
rf_pred_interval
                                                        rf_amp1 = prev_amp
                                                        rf_amp2 =  prev_amp
                                                        print("########### executed!")

                                                # take note of current blackout amplitude, next prediction
shouldn't exceed this
                                                if ytest[j] > 0:
                                                        prev_amp=ytest[j]
                                                        pred_permit = 1
                                                        if rf_interval1 == 0 and rf_interval2 == 0:
                                                                rf_interval1 = j
                                                                rf_amp1 = ytest[j]
                                                        elif rf_interval1 > 0 and rf_interval2 == 0:
                                                                rf_interval2 = j
                                                                rf_amp2 = ytest[j]
                                                if ytest[j]==0:
                                                        if blk_ypred_rf[j] > 0:
                                                                pred_permit = 0
                                                                rf_interval1 = 0
                                                                rf_interval2 = 0

                                                # include routine to track blackout interval, and predict
third occurance based on the interval

                                                #adjust predicted hr with previous hr error
                                                # blk_ypred_rf[j] = blk_ypred_rf[j] + predError_rf
                                                # predError_rf = ytest[j] - blk_ypred_rf[j]
```

```
                                        # predError_rf = predError_rf/2
                                        #pass
                                elif pred_permit == 0:

                                        blk_ypred_rf[j]=0 #assume no blackout

                                        # if blackout occurs, grant permission to make new
predictions, until the next incorrect prediction
                                        if ytest[j] > 0:
                                                pred_permit = 1
                                                prev_amp=ytest[j]
                                actualPrediction = blk_ypred_rf[j]
                                actualBlackout = ytest[j]
                                #element can't be higher than 1
                                if blk_ypred_rf[j] > 0:
                                        blk_ypred_rf[j] = 1
                                #no element permited below zero
                                if blk_ypred_rf[j] < 0:
                                        blk_ypred_rf[j] = 0



                                #add actual observations to history for next loop
                                historyX=np.vstack((historyX,xtest[j]))
                                historyY=np.hstack((historyY,ytest[j]))

                        # ensure no element is greater than 1 or less than 0 or greater than sliding window
maxAmplitude



                                #mae_rf = round(metrics.mean_absolute_error(ytest,blk_ypred_rf),3)
                                mae_rf = round(metrics.mean_absolute_error(ytest,blk_ypred_rf),3)
                                mse_rf = round(metrics.mean_squared_error(ytest,blk_ypred_rf),3)
                                r2_rf = round(metrics.r2_score(ytest,blk_ypred_rf),3)
                                mape_rf =
round(metrics.mean_absolute_percentage_error(ytest,blk_ypred_rf),3)


                                #store all predictions in a list
                                #predictions.append(y_pred)
                                blackout_pred_rf=np.hstack((blackout_pred_rf,blk_ypred_rf[j]))
                                blackout_actual=np.hstack((blackout_actual,ytest[j]))
                                #observations.append(ytest)
                                #print("ytest is ",ytest)
                                #print("xtest is ",xtest)
                                pastday=historyY[len(historyY)-24:len(historyY)]

                                if k==start_index:
                                        pastdayLog = pastday

                                else:
                                        #print("°°° predictions_lstm1 is now
\n",predictions_lstm1,"\nypred_lstm1 is \n",predictions_lstm1)
                                        pastdayLog = np.hstack((pastdayLog,pastday))

                                #matokeo=round(model_svr.score(xtest,ytest),1)

                                maeScore_rf.append(mae_rf)
                                mseScore_rf.append(mse_rf)
                                r2Score_rf.append(r2_rf)
                                mapeScore_rf.append(mape_rf)
```

```
                        #print("SVR efficiency mae: ",mae_svr)


                                #print("RFefficiency mae: ",mae_rf)
                                #print("past day len",len(pastday))
                                #print("k is ",k)
                                mae_asd =
round(metrics.mean_absolute_error(oggi.next_blackout,next_day),3)
                                RF_ASD =[pd.DataFrame(next_day),pd.DataFrame(blk_ypred_rf)]
                                RF_ASD=pd.concat(RF_ASD).groupby(level=0).mean()
                                RF_ASD=RF_ASD.round(1)
                                RF_ASD=np.array(RF_ASD.values)
                                maeRFasd =
round(metrics.mean_absolute_error(oggi.next_blackout,RF_ASD),3)

                        #if oggi.Date[0] == "2021-04-01" and blocker==1:
                        if blocker==1:
                                worstday_15min = next_day

        worstday_15min=pd.DataFrame(worstday_15min,columns=['worstday_15min'])
                                worstday_15min.to_pickle("./worstday_15min.pkl")
                                dpi = 600
                                px = 1/plt.rcParams['figure.dpi']
                                #plt.figure(figsize=(10, 4))
                                plt.figure(figsize=(dpi*px, dpi*px))
                                #plt.plot( next_day, color ='b')
                                plt.plot( blk_ypred_rf, color ='r',marker='o')
                                #plt.plot( RF_ASD, color ='g')
                                plt.plot( oggi.next_blackout, color ='k',marker='o')
                                #plt.plot( oggi.blackout, color ='b',linestyle = ':')
                                plt.xlabel('Time[hr]')
                                plt.xticks(np.arange(0,dayLength,2))
                                plt.ylabel('blackout index')
                                plt.title('blackout forecast' )
                                plt.legend(['Random Forest mae: '+str(mae_rf),'actual'])
                                #plt.legend(['ASD mae: '+str(mae_asd),'Random Forest mae:
'+str(mae_rf),'RF-ASD mae: '+str(maeRFasd),'actual','yesterday'])
                                plt.tight_layout()
                                plt.show()
                                #plt.savefig('img/blackout_forecast.png')
                                plt.close()
                                #blocker=1

                        '"if bestMaeDate == "2021-10-13" and blocker==1:
                                dpi = 600
                                px = 1/plt.rcParams['figure.dpi']
                                #plt.figure(figsize=(10, 4))
                                plt.figure(figsize=(dpi*px, dpi*px))
                                plt.plot( next_day, color ='b')
                                plt.plot( blk_ypred_rf, color ='r')
                                plt.plot( RF_ASD, color ='g')
                                plt.plot( ytest, color ='k',marker='o')
                                plt.plot( pastday, color ='b',linestyle = ':')
                                plt.xlabel('Time')
                                plt.xticks(np.arange(0,24,2))
                                plt.ylabel('blackout index')
                                plt.title('blackout forecast bestMaeDate'+oggi.Date[0])
                                plt.legend(['ASD mae: '+str(mae_asd),'Random Forest mae:
'+str(mae_rf),'RF-ASD mae: '+str(maeRFasd),'actual','yesterday'])
                                plt.tight_layout()
                                plt.show()
                                #plt.savefig('img/blackout_forecast.png')
```

```
                                plt.close()
                                blocker=2'''


        print("bestMaeDate ",bestMaeDate," bestMae ",bestMae,
        "\nworstMaeDate ",worstMaeDate,"worstMae ",worstMae)

        # #make prediction for nextday i.e current day and save
        # xtest_today=allDays.loc[allDays.Date==all_dates[-1] ]
        # #print("xtest_today length is :",len(xtest_today))
        # #print("xtest_today length is :",xtest_today)
        # xtest_today.reset_index(drop=True, inplace=True)
        # #select relevant columns only
        # xtest_today = xtest_today[desired_features]
        # #print("xtest_today \n",xtest_today)
        # d1=allDays.loc[allDays.Date==all_dates[-2]].blackout
        # d1.reset_index(drop=True, inplace=True)
        # d2=allDays.loc[allDays.Date==all_dates[-3]].blackout
        # d2.reset_index(drop=True, inplace=True)
        # #print("d2 \n",d2)
        # d3=allDays.loc[allDays.Date==all_dates[-4]].blackout
        # d3.reset_index(drop=True, inplace=True)
        # #print("d3 \n",d3)
        # d4=allDays.loc[allDays.Date==all_dates[-5]].blackout
        # d4.reset_index(drop=True, inplace=True)
        # #print("d4 \n",d4)
        # d5=allDays.loc[allDays.Date==all_dates[-6]].blackout
        # d5.reset_index(drop=True, inplace=True)
        # #print("d5 \n",d5)
        # #print("d5 full data\n",allDays.loc[allDays.Date==all_dates[-6]])
        # d6=allDays.loc[allDays.Date==all_dates[-7]].blackout
        # d6.reset_index(drop=True, inplace=True)
        # #print("d6 \n",d6)
        # #print("d6 full data\n",allDays.loc[allDays.Date==all_dates[-7]])
        # d7=allDays.loc[allDays.Date==all_dates[-8]].blackout
        # d7.reset_index(drop=True, inplace=True)
        # #print("d7 \n",d7)
        # #print("d7 full data\n",allDays.loc[allDays.Date==all_dates[-8]])

        # try:
        #     xtest_today =
xtest_today.assign(b7=d7.values,b6=d6.values,b5=d5.values,b4=d4.values,b3=d3.values,b2=d2.values,b1=d1.v
alues)
        # except Exception as error:
        #     print("error has occured!")
        #           #msg =msg+ str(ct)+" An exception occurred: {}".format(traceback.format_exc())



        #print("final dataset \n",xtest_today)

        #xtest=xtest_today.values[:,1:]
        #print("xtest dataset \n",xtest)
        #make prediction once a day
        #blk_ypred_svr = blackoutmodel_svr.predict(xtest)
        #blk_ypred_rf = blackoutmodel_rf.predict(xtest)
        #add predicted with yesterday value
        #get mean of predicted data and past 2 days data

        ypred_mean1 =[pd.DataFrame(next_day_list2),pd.DataFrame(blackout_pred_rf)]
        ypred_mean1=pd.concat(ypred_mean1).groupby(level=0).mean()
        ypred_mean1=ypred_mean1.round(1)
        ypred_mean1.columns = ['data']
```

```
        blackout_actual=pd.DataFrame(blackout_actual,columns=['blackout_actual'])
        blackout_pred_rf=pd.DataFrame(blackout_pred_rf,columns=['blackout_pred_rf'])
        blackout_actual.to_pickle("./blackout_actual.pkl")
        blackout_pred_rf.to_pickle("./blackout_pred_rf.pkl")
        blackout_actual = pd.read_pickle("./blackout_actual.pkl")
        blackout_actual['blackout_actual'] = blackout_actual['blackout_actual'].apply(np.ceil)
        blackout_pred_rf = pd.read_pickle("./blackout_pred_rf.pkl")
        blackout_pred_rf['blackout_pred_rf'] = blackout_pred_rf['blackout_pred_rf'].apply(np.ceil)
        blackout_actual    = blackout_actual.blackout_actual.values
        blackout_pred_rf = blackout_pred_rf.blackout_pred_rf.values

        print("ypred_mean1\n",ypred_mean1)
#currentDay.loc[(currentDay.blackout > 0),('blackout')] = 1
        ypred_mean1.loc[ypred_mean1.data > 0 ,('data')] = 1
        ypred_mean1.to_pickle("./results15min/ypred_mean1.pkl")
        ypred_mean1=np.array(ypred_mean1.data.values)
        mae_asd_rf = round(metrics.mean_absolute_error(blackout_actual,ypred_mean1),3)
        rmse_asd_rf = round(math.sqrt(metrics.mean_squared_error(blackout_actual,ypred_mean1)),3)
        mse_asd_rf = round(metrics.mean_squared_error(blackout_actual,ypred_mean1),3)
        r2_asd_rf = round(metrics.r2_score(blackout_actual,ypred_mean1),3)

        #blackoutForecast=np.round(max(ypred_mean)*100,1)
        #print("blackout_actual length:",len(blackout_actual)," days: ",len(blackout_actual)/24)

        mean_list2_mae =round(np.mean(maeScore_list2),3)
        mean_list2_rmse =round(np.mean(rmseScore_list2),3)
        mean_list2_mse =round(np.mean(mseScore_list2),3)
        mean_list2_rmse =round(np.mean(rmseScore_list2),3)
        mean_list2_r2score =round(np.mean(r2Score_list2),3)

        mean_rf_mae =round(np.mean(maeScore_rf),3)
        mean_rf_rmse =round(math.sqrt(np.mean(mseScore_rf)),3)
        mean_rf_mse =round(np.mean(mseScore_rf),3)
        mean_rf_r2 =round(np.mean(r2Score_rf),3)

        print("RF: mae(",mean_rf_mae,") rmse(",mean_rf_rmse,") mse(",mean_rf_mse,") r2(",mean_rf_r2,")")
        print("ASD: mae(",mean_list2_mae,") rmse(",mean_list2_rmse,") mse(",mean_list2_mse,")
r2(",mean_list2_r2score,")")
        print("RF-ASD: mae(",mae_asd_rf,") rmse(",rmse_asd_rf,") mse(",mse_asd_rf,") r2(",r2_asd_rf,")")

        print("first date ",x_test['Date'])
        #print("last date ",x_test['Date'][len(x_test)-1])

        check_timer(tic)
        #make audible tone
        beeper()



        cm = confusion_matrix(blackout_actual,blackout_pred_rf)
        print("confusion matrix\n",cm)
        cm_df = pd.DataFrame(cm,
             index = ['Power_ON','Blackout'],
             columns = ['Power_ON','Blackout'])
        print("confusion matrix dataframe\n",cm_df)
        print("RF accuracy: ",accuracy_score(blackout_actual,blackout_pred_rf))


        plt.close()
        dpi = 1200
        px = 1/plt.rcParams['figure.dpi']
        #plt.figure(figsize=(10, 4))
```

```
        #plt.figure(figsize=(dpi*px, dpi*px))
        #f,(ax1,ax2,ax3) = plt.subplots(1,3,sharey=True,figsize=(dpi*px, dpi*px))
        f,(ax1,ax2,ax3, axcb) = plt.subplots(1,4,
       gridspec_kw={'width_ratios':[1,1,1,0.08]},figsize=(10, 4))
        ax1.get_shared_y_axes().join(ax2,ax3)
        labels = [np.round(value*100,1) for value in

        cm_df.values/np.sum(cm_df.values)]
        a =[f"{value}%" for value in labels[0]]
        a = [np.array([f"TN\n{cm_df.values[0][0]}\n{a[0]}", f"FP\n{cm_df.values[0][1]}\n{a[1]}"])]
        b =[f"{value}%" for value in labels[1]]
        b = [np.array([f"FN\n{cm_df.values[1][0]}\n{b[0]}", f"TP\n{cm_df.values[1][1]}\n{b[1]}"])]
        labels = np.concatenate((a, b), axis=0)
        labels = labels.reshape(2,2)
        g1 = sns.heatmap(cm_df, annot=labels, annot_kws={'size': 15} ,cmap='YlGnBu',
fmt='',cbar=False,ax=ax1)
        g1.set_title('RF confusion matrix')
        g1.set_ylabel('Actual values',fontsize=13,fontweight='bold')
        g1.set_xlabel('')
        g1.set_yticklabels(cm_df.index,fontsize=13)

        next_day_list2 = np.ceil(next_day_list2)
        pd.DataFrame(next_day_list2).to_pickle("./results15min/next_day_list2.pkl")
        cm = confusion_matrix(blackout_actual,next_day_list2)
        print("confusion matrix\n",cm)
        cm_df = pd.DataFrame(cm,
             index = ['Power_ON','Blackout'],
             columns = ['Power_ON','Blackout'])
        print("confusion matrix dataframe\n",cm_df)
        print("ASD accuracy: ",accuracy_score(blackout_actual,next_day_list2))
        labels = [np.round(value*100,1) for value in

        cm_df.values/np.sum(cm_df.values)]
        a =[f"{value}%" for value in labels[0]]
        a = [np.array([f"TN\n{cm_df.values[0][0]}\n{a[0]}", f"FP\n{cm_df.values[0][1]}\n{a[1]}"])]
        b =[f"{value}%" for value in labels[1]]
        b = [np.array([f"FN\n{cm_df.values[1][0]}\n{b[0]}", f"TP\n{cm_df.values[1][1]}\n{b[1]}"])]
        labels = np.concatenate((a, b), axis=0)
        labels = labels.reshape(2,2)
        g2 = sns.heatmap(cm_df, annot=labels,  annot_kws={'size': 15} ,cmap='YlGnBu',
fmt='',cbar=False,ax=ax2)
        g2.set_title('ASD confusion matrix')
        g2.set_ylabel('')
        g2.set_xlabel('Predicted values', fontsize=13,fontweight='bold')
        g2.set_yticks([])

        cm = confusion_matrix(blackout_actual,ypred_mean1)
        print("confusion matrix\n",cm)
        cm_df = pd.DataFrame(cm,
             index = ['Power_ON','Blackout'],
             columns = ['Power_ON','Blackout'])
        print("confusion matrix dataframe\n",cm_df)
        print("RF-ASD accuracy: ",accuracy_score(blackout_actual,ypred_mean1))
        print("RF-ASD recall:
",(cm_df.Blackout['Blackout']/(cm_df.Blackout['Blackout']+cm_df.Power_ON['Blackout']))*100)
        labels = [np.round(value*100,1) for value in

        cm_df.values/np.sum(cm_df.values)]
        a =[f"{value}%" for value in labels[0]]
        a = [np.array([f"TN\n{cm_df.values[0][0]}\n{a[0]}", f"FP\n{cm_df.values[0][1]}\n{a[1]}"])]
        b =[f"{value}%" for value in labels[1]]
        b = [np.array([f"FN\n{cm_df.values[1][0]}\n{b[0]}", f"TP\n{cm_df.values[1][1]}\n{b[1]}"])]
        labels = np.concatenate((a, b), axis=0)
```

```
        labels = labels.reshape(2,2)
        g3 = sns.heatmap(cm_df, annot=labels , annot_kws={'size': 15},cmap='YlGnBu',
fmt='',ax=ax3,cbar_ax=axcb)
        g3.set_title('RF-ASD confusion matrix')
        g3.set_ylabel('')
        g3.set_xlabel('')
        g3.set_yticks([])
        plt.tight_layout()
        plt.savefig('img/confusion_matrix.png',dpi=1200)
        plt.show()

 ##########################################################################
        exit()

        #confusion matrix plot for pure RF
        #print('',blackout_actual\n",blackout_actual,"blackout_pred_rf\n",blackout_pred_rf)
        cm = confusion_matrix(blackout_actual,blackout_pred_rf)
        print("confusion matrix\n",cm)
        cm_df = pd.DataFrame(cm,
             index = ['Power_ON','Blackout'],
             columns = ['Power_ON','Blackout'])
        print("confusion matrix dataframe\n",cm_df)

        plt.close()
        dpi = 1200
        px = 1/plt.rcParams['figure.dpi']
        #plt.figure(figsize=(10, 4))
        plt.figure(figsize=(dpi*px, dpi*px))
        plt.subplot(131)
        sns.heatmap(cm_df, annot=True, annot_kws={'size': 15} ,cmap='Blues', fmt='g')#fmt='g' ensures
values are printed as they are without using scientific exponent notation
        #plt.suptitle(month+' Blackout Heatmap', fontsize=30)
        plt.title('Confusion Matrix',fontsize=16)
        plt.ylabel('Actual Values',fontsize=16)
        plt.xlabel('Predicted Values',fontsize=16)
        #plt.tight_layout()
        #plt.savefig('img/confusion_matrix_rfclf.png')
        #plt.show()

        #confusion matrix plot for ASD
        #print('',blackout_actual\n",blackout_actual,"blackout_pred_rf\n",blackout_pred_rf)
        next_day_list2 = np.ceil(next_day_list2)
        pd.DataFrame(next_day_list2).to_pickle("./results15min/next_day_list2.pkl")
        #next_day_list2.to_pickle(".results15min/next_day_list2.pkl")
        cm = confusion_matrix(blackout_actual,next_day_list2)
        print("confusion matrix\n",cm)
        cm_df = pd.DataFrame(cm,
             index = ['Power_ON','Blackout'],
             columns = ['Power_ON','Blackout'])
        print("confusion matrix dataframe\n",cm_df)

        #plt.close()
        #dpi = 1200
        #px = 1/plt.rcParams['figure.dpi']
        #plt.figure(figsize=(10, 4))
        #plt.figure(figsize=(dpi*px, dpi*px))
        plt.subplot(132)
        sns.heatmap(cm_df, annot=True,  annot_kws={'size': 15} ,cmap='Blues', fmt='g')#fmt='g' ensures
values are printed as they are without using scientific exponent notation
        #plt.suptitle(month+' Blackout Heatmap', fontsize=30)
        plt.title('ASD Confusion Matrix',fontsize=16)
        plt.ylabel('Actual Values',fontsize=16)
        plt.xlabel('Predicted Values',fontsize=16)
```

```
#plt.tight_layout()
#plt.savefig('img/confusion_matrix_ASDclf.png')
#plt.show()

#confusion matrix plot for pure RF-ASD
#print('',blackout_actual\n",blackout_actual,"blackout_pred_rf\n",blackout_pred_rf)

cm = confusion_matrix(blackout_actual,ypred_mean1)
print("confusion matrix\n",cm)
cm_df = pd.DataFrame(cm,
     index = ['Power_ON','Blackout'],
     columns = ['Power_ON','Blackout'])
print("confusion matrix dataframe\n",cm_df)

#plt.close()
#dpi = 1200
#px = 1/plt.rcParams['figure.dpi']
#plt.figure(figsize=(10, 4))
#plt.figure(figsize=(dpi*px, dpi*px))
plt.subplot(133)
sns.heatmap(cm_df, annot=True , annot_kws={'size': 15},cmap='Blues', fmt='g')#fmt='g' ensures
values are printed as they are without using scientific exponent notation
#plt.suptitle(month+' Blackout Heatmap', fontsize=30)
plt.title('RF-ASD Confusion Matrix',fontsize=16)
plt.ylabel('Actual Values',fontsize=16)
plt.xlabel('Predicted Values',fontsize=16)
plt.tight_layout()
plt.savefig('img/confusion_matrix_RFASDclf.png')
plt.show()

#line plot
plt.close()
dpi = 1200
px = 1/plt.rcParams['figure.dpi']
#plt.figure(figsize=(10, 4))
fig = plt.figure(figsize=(dpi*px, dpi*px))
#plt.figure(figsize=(dpi*px, dpi*px))
ax = plt.subplot(111)
#plt.plot( next_day_list1, color ='b',marker='o')
ax.plot( next_day_list2, color ='g',marker='s',lw=3)
ax.plot( ypred_mean1, color ='r',marker='o',linestyle = '-.')
ax.plot( blackout_pred_rf, color ='b',marker='P',linestyle = '--')
ax.plot( blackout_actual, color ='k',marker='o')
#ax.plot( pastdayLog, color ='r',marker='.',linestyle = ':')
#plt.plot( next_day_list2, color ='m',marker='.')
#plt.plot( ypred_mean1, color ='g',marker='o',linestyle = '-.')
#plt.plot( blackout_pred_rf, color ='b',linestyle = '--')
#plt.plot( blackout_actual, color ='k',marker='o')
#plt.plot( pastdayLog, color ='r',marker='.',linestyle = ':')
#plt.axvline(x=24,  color='k', linestyle='--')
#plt.text(24,1,'d1 ',horizontalalignment='right')
xposition = [(dayLength*x)-1 for x in range(1,(int(len(blackout_actual)/dayLength))+1,1)]
for xc in xposition:
     plt.axvline(x=xc, color='k', linestyle=':',ymin=0, ymax=1)
     plt.text(xc,1,'d'+str(int(np.ceil(xc/dayLength)))+' ',horizontalalignment='right')
plt.xlabel('Time [Hr]')
plt.ylabel('Blackout Index')
plt.title('Blackout Forecast')
plt.ylim(-0.03, 1.05)
#plt.legend(['SVR blackout','RF blackout','actual','yesterday'])
#     plt.legend([
#          #'SD1 (mae:'+str(mean_list1_mae)+')',
#     'ASD (mae:'+str(mean_list2_mae)+', rmse:'+str(mean_list2_rmse)+')',
```

```
#           'ASD-RF (mae: '+str(mae_asd_rf)+', rmse:'+str(rmse_asd_rf)+')',
#           'RF (mae: '+str(mean_rf_mae)+', rmse:'+str(mean_rf_rmse)+')',
#           'actual','previous day'],loc='upper center',ncol=2)
ax.legend([
        #'SD1 (mae:'+str(mean_list1_mae)+')',
'ASD (mae:'+str(mean_list2_mae)+')',
'ASD-RF (mae: '+str(mae_asd_rf)+')',
'RF (mae: '+str(mean_rf_mae)+')',

'actual'],loc='upper center',bbox_to_anchor=(0.9, 0.95),ncol=1)
#'actual','previous day'],loc='upper center',bbox_to_anchor=(0.5, -0.08),ncol=5)
#matplotlib.rcParams['legend.fontsize'] = 20
plt.tight_layout()
plt.savefig('img/blackout_15min_classifier.png')
plt.show()
```

Appendix 4: Smart Meter Arduino Code

```
// Include the libraries:
#include <PZEM004Tv30.h>
//#include <Wire.h> // Library for I2C communication, used with lcd and sending data to wifi module
//#define i2c_address 6
#define baudrate 9600
String received_i2c_data;
#define datasize 200 // ample datasize to accomodate all characters sent from arduino


#include <LiquidCrystal_I2C.h> // Library for LCD
#include "RTClib.h"
RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
// Wiring: SDA pin is connected to A4 and SCL pin to A5.
// Connect to LCD via I2C, default address 0x27 (A0-A2 not jumpered)
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x3F, 20, 4); // Change to (0x27,16,2) for 16x2 LCD.
#include "DHT.h"
#define DHTPIN 7     // Digital pin connected to the DHT sensor
// Uncomment whatever type you're using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors.  This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 8

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

// configuration for bluetooth module ---------------------------------------------
#include <SoftwareSerial.h>
#define rxPin 18
#define txPin 22

#define baudrate 57600
SoftwareSerial bluetooth(rxPin ,txPin);
String bluetooth_data; //bluetooth_data buffer
char character;

#include <ArduinoJson.h>
//~~~~~~~~~~~~~~~~~~~~ configuration parameters ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ //

// configuration for sim800L gsm module ---------------------------------------------
SoftwareSerial mySerial(42, 44); ////SIM800L Tx & Rx is connected to Arduino #10 & #11
//~~~~~~~~~~~~~~~~~~~~ configuration parameters ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ //
// These are parameters that can be adjusted/tuned
// constants won't change :
```

```
const long interval = 10000;          // 30s interval at which to show time and check pir pin
(milliseconds)***************************
const long interval_pir = 1;          // 5min interval at which to check pir pin (minutes)
unsigned long interval_log = 1;          // interval at the end of the day, after sheet data has been cleared
float offsetVoltage = 2338.9896875;//2395.930859375 ;//2508.7084375;//2412;//2377.92;//2363;//2533.85; // ac
current sensor offset voltage
int sampling = 500;          // read 500 samples of voltage & current sensor analog pins
float OFFset_inv2 = 0;//offset for inverter voltage value
const long poll_pir_interval = 1000;     // check pir & dht22 status every 2s
//
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~ //
const int wifiModule_enable = 10;
//---- parameters for load3 --------------------------
const int load1 = 32; // photocopier
const int load2 = 30; // cctv
const int load3 = 28;  // lights , epson printer             // choose the pin for the LED/load3
bool LightON = HIGH, LightOFF = LOW;
String load1_status="1",load2_status="1",load3_status="1", inverter_fan="0",loadControl_flag;
String inverter_status="1",pv_charger_status="1",inverter_temp="0",tanesco_charger_status="1";

//---- parameters for PIR --------------------------
const int inputPin = 3;              // choose the input pin (for PIR sensor)


//const byte interruptPin = 2;    // choose the input pin (for PIR sensor)
int motion = 0, person = 0, backlight_timer=0;                  // variable for reading the PIR pin status
 // flag for pir wait time, this ensures that the mcu has waited at least 10minutes of
 //inactivity before turning off appliances
bool wait_time = false, wait_more_time = false, appliance = true, clear_sheet_data = false ;

//---- parameters for RTC --------------------------
// Generally, you should use "unsigned long" for variables that hold time
// The motionue will quickly become too large for an int to store

//String ds18b20_temp,dht22_temp,hic,dht22_humidity,load1,load2,load3,ac_voltage,currentvalue,frequency
,pv_volt,pv_current,charged_percent,charging_status ,bat_volt ,person,dc_fan_status ,inv_fan_status ,inv_temp
,inv_status ,power_blackout ,reed_switch_indicator , mega2nodemcu;

const long interval_nodemcu_post = 0;
unsigned long previousMillis_nodemcu_post = 0;
unsigned long previous_pir_poll = 0;        // will store last time LED was updated
unsigned long previousMillis = 0;        // will store last time LED was updated
unsigned long previousMillis_pir = 0;        // will store last time LED was updated
unsigned long currentMillis_pir = 0;
unsigned long previous_tracker_log = 0;        // will store last time LED was updated
  unsigned long current_tracker_log = 0;

//---- parameters for Battery & PV sensor ---------------------------------
const int charging_status = 24 ; //pwm on / off charging indicator
const int bat_volt_sensor = A5;//A4 is possibly damaged, it's giving false adc values
float bat_offset = 0.2;

float pvCounter = 0;
float PVavg = 0;

const int pv_current_sensor = A3;
const int pv_currentSensor = 5;
float pv_current =0;
float pv_value = 0;
float pv_current_peak_value = 0;
float pv_current_min_value = 0;
const int pv_volt_sensor = A2;
```

```
float adc_value = 0, vin = 0, bat_volt=0, pv_volt = 0;
float R1 = 100000, R2 = 10000;
const int pv_pwm = 6;
int pv_charging_counter = 0, pv_charger_flag=1,charged_percent = 0, eveningFlag = 0;
const int dc_fan = 38;
int chon = 60; // charging time duration
int choff = 65; // charging OFF time limit
float setpoint = 14; // battery max charge set point
float difference = 0; // var for storing difference between setpoint and actual battery charge

float OFFset_inv1 = 2319;//2368.1640625;// since 220V produces 751 adc motion, then 220/751=0.292943

const int inv_enable = 36;
const int inv_current = A8;
float inv_current_peak_value = 0;
float inv_current_min_value = 0;
float inv_value = 0;
float inv_current_value = 0;

float iCurrentCounter = 0;
float iCurrentavg = 0;

//---------- parameters for ac current , voltage sensor, & frequency counter ------------------------------
const int currentPin = A0;
const int currentSensor = 9;
float current_peak_value = 0;
float current_min_value = 0;
int sensitivity = 66;//66 100 185;
float currentvalue = 0;
float current_buffer =0;
float current_avg=0;

const int ac_sensor = A1;
float ac_value =0;
float ac_voltage=0;
float acVoltage = 0;
float ac_avg = 0;
 float OFFset_ac1 = 0.315417256;// since 220V produces 751 adc motion, then 220/751=0.292943
 float currentOffset = 0.5;//0.16; //obtained by 1.3k=0.73 ...0.73A was the true value, while 1.3A was the
recorded value
 //hence each adc motionue multiplied by 0.29 will give an approximate ac voltage
 float invCurrentOffset = 0.125;
 float POWER = 0;

 float FreqOffset =0.494;
const int frequency_count_pin = 2;
int acPowerStatus = 34;
unsigned long previous_acPower_poll = 0;
const long poll_acPower_interval = 5000;
int power_blackout = 1;
int blackout_flag = 0;
//---- reed switch  ---------------------------------
const int reed_switch = 26;                // show if office door is open/close
const int heartBeatpin =40;
bool heartbeat = 0;

String mega2nodemcu="0";
float ds18b20_temp,dht22_temp,heat_index,dht22_humidity;
int post = 1;
uint16_t tmr1 = 0;
float period, frequency;

#include <SPI.h>
```

```
#include <SD.h>

File myFile;
int epoch=0;
String myFileName = "";
//String FileName = "";
String dataString = "";
Sd2Card card;

SdVolume volume;

SdFile root;
float SDstatus = 0;
float cardStatus = 0;
//SoftwareSerial nodeSerial(15, 14); // RX, TX

PZEM004Tv30 pzem(11, 12); // Software Serial pin 11 (RX) & 12 (TX)
String received_uart_data;
//********************** functions ****************************************
void nodemcu2arduino(){
    // if wifimodule is off turn it ON
    Serial.print("wifiEnable pin is ");Serial.println(digitalRead(wifiModule_enable));
  if(digitalRead(wifiModule_enable)== 0){
  digitalWrite(wifiModule_enable, HIGH); // Now turn wifimodule ON
  }
  received_uart_data = ""; //clear serial comm receiver first
  //Begin serial communication with Arduino and nodemcu
while (Serial3.available() > 0) {
  // read the incoming byte from nodemcu:
    //String str = Serial3.readString();
     String str = Serial3.readStringUntil('\n');
    Serial.print("received: ");Serial.println(str);
     received_uart_data.concat(str);
    //detect last character
    /*if(str.substring(0) == "\n"){
      break; // end loop
      }*/
}
//cleanup received data, remove mangled characters
int index2 = received_uart_data.indexOf("ʔ",3);
if(received_uart_data.indexOf("ʔ",3)>0){
  received_uart_data = "";//bad data received, fetch another stream
  }
int index1 = received_uart_data.indexOf("{",3);

 if(index1>0){
 Serial.print("start index: ");Serial.println(index1);
received_uart_data = received_uart_data.substring(index1);
Serial.print("*clean data: ");Serial.println(received_uart_data);
}
else
return; // no need to continue(bad data received) break out of function



DynamicJsonDocument doc(2048);
 deserializeJson(doc, received_uart_data);



//StaticJsonDocument<1024> doc;

//DeserializationError error = deserializeJson(doc, Serial3);
```

```
 // root.prettyPrintTo(Serial);
 //serializeJsonPretty(doc,Serial) ;
 //Serial.println("");

    String ld1=doc["load1"];
    String ld2=doc["load2"];
    String ld3=doc["load3"];
    String inv=doc["inverter"];
    String tanesco=doc["tanescoCharger"];
    String pv=doc["pvCharger"];
    String inv_fan=doc["inverter_fan"];
    String inv_temp=doc["inverter_temp"];
    String load_flag=doc["loadControl_flag"];

     // update arduino mega global variables with data from wifi module
     load1_status=ld1;
    load2_status=ld2;
     load3_status=ld3;
     inverter_status=inv;
     tanesco_charger_status=tanesco;
     pv_charger_status=pv;
     inverter_fan=inv_fan;
     inverter_temp=inv_temp;
     loadControl_flag=load_flag;

Serial.print("data received from nodemcu is ");
Serial.println(" "+load1_status+" "+load2_status+" "+load3_status+" "+inverter_status+"
"+tanesco_charger_status+" "+pv_charger_status+" "+inverter_fan+" "+inverter_temp);
return;

// Now control arduino mega I/O based on data received from wifi module
  // if wifimodule is off turn it ON
  if(loadControl_flag=="reset_wifi"){
  digitalWrite(wifiModule_enable, LOW); // Now turn wifimodule ON
  delay(3000); // wait 3seconds
  digitalWrite(wifiModule_enable, HIGH); // Now turn wifimodule ON
  loadControl_flag="load1_on";
  }
if(loadControl_flag=="load1_on"){ digitalWrite(load1, LOW); }
  else if(loadControl_flag=="load1_off"){digitalWrite(load1, HIGH);}
  else if(loadControl_flag=="load2_on"){digitalWrite(load2, HIGH);}
  else if(loadControl_flag=="load2_off"){digitalWrite(load2, LOW);}
  else if(loadControl_flag=="load3_on"){digitalWrite(load3, LightON);}
  else if(loadControl_flag=="load3_off"){digitalWrite(load3, LightOFF);}
  else if(loadControl_flag=="pv_on"){ pv_charger_flag=1;}
  else if(loadControl_flag=="pv_off"){pv_charger_flag=0;}
  else{
    if(pv_charger_status=="1"){
  pv_charger_flag=1;
  Serial.println("**** pv_charger_status is now ON ");
  }
else if(pv_charger_status=="0"){
  pv_charger_flag=0;
   Serial.println("**** pv_charger_status is now OFF ");
  }
  if( load1_status=="1"){
  digitalWrite(load1, LOW); // turn ON load1, load1 uses inverted logic where off is HIGh and on is LOW
       Serial.println("turning load1 ON ");
       }
       else if( load1_status=="0"){
          digitalWrite(load1, HIGH); // turn OFF load1
```

```
          Serial.println("turning load1 OFF ");
        }
     if( load2_status=="1"){
  digitalWrite(load2, HIGH); // turn ON load1
       Serial.println("turning load2 ON ");
       }
       else if( load2_status=="0"){
          digitalWrite(load2, LOW); // turn OFF load1
           Serial.println("turning load2 OFF ");
        }
         if( load3_status=="1" ){
  digitalWrite(load3, LightON); // turn ON load1
       Serial.println("turning load3 ON ");
       }
       else if( load3_status=="0"){
          digitalWrite(load3, LightOFF); // turn OFF load1
           Serial.println("turning load3 OFF ");
        }
   }

  post = 1;
  }

void arduino2nodemcu(){
  // if wifimodule is off turn it ON
  Serial.print("wifiEnable pin is ");Serial.println(digitalRead(wifiModule_enable));
  if(digitalRead(wifiModule_enable)== 0){
  digitalWrite(wifiModule_enable, HIGH); // Now turn wifimodule ON
  }
  unsigned long currentMillis_nodemcu_post = millis();
  if (currentMillis_nodemcu_post - previousMillis_nodemcu_post >= interval_nodemcu_post) {
    // save the last time you processed this block
    previousMillis_nodemcu_post = currentMillis_nodemcu_post;

//StaticJsonDocument<1024> doc;
DynamicJsonDocument doc(2048);
 doc["ds18b20_temp"] = String(ds18b20_temp);
  doc["dht22_temp"] = String(dht22_temp);
  doc["heat_index"] = String(heat_index);
  doc["dht22_humidity"] = String(dht22_humidity);
  doc["load1"] = load1_status;
  doc["load2"] = String(digitalRead(load2));
  doc["load3"] = String(!digitalRead(inv_enable)); // this corresponds to load 3 being high
  doc["ac_voltage"] = String(ac_voltage);
  doc["currentvalue"] = String(currentvalue);
  doc["frequency"] = String(frequency);
  doc["inv_current"] = String(inv_current_value);
  doc["pv_volt"] = String(pv_volt);
  doc["pv_current"] = String(pv_current);
  doc["charged_percent"] = String(charged_percent);
  doc["charging_status"] = String(digitalRead(charging_status));
  doc["bat_volt"] = String(bat_volt);
  doc["person"] = String(person);
  doc["dc_fan_status"] = String(digitalRead(dc_fan));
  doc["inv_fan_status"] = inverter_fan;
  doc["inv_temp"] = inverter_temp;
  doc["inv_status"] = inverter_status;
  doc["power_blackout"] = String(power_blackout);
  doc["reed_switch_indicator"] = String(digitalRead(reed_switch));
   doc["cardStatus"] = String(cardStatus);
//mega2nodemcu = "ds18b20_temp=" +  String(ds18b20_temp) + "&dht22_temp=" + String(dht22_temp)+
"&dht22_heatIndex_temp=" + String(heat_index)+  "&dht22_humidity=" +
String(dht22_humidity)+"&load1="+load1_status+"&load2="+load2_status+"&load3="+load3_status+
```

```
"&ac_voltage=" + String(ac_voltage)+ "&ac_current=" + String(currentvalue)+
"&frequency="+String(frequency) + "&inv_current=" + String(inv_current_value)+"&pv_voltage=" +
String(pv_volt)+  "&pv_current=" + String(pv_current)+  "&soc=" + String(charged_percent)+
"&pv_bat_charging=" + String(digitalRead(charging_status))+  "&battery_voltage=" + String(bat_volt)+
"&pir_status=" + String(person)+  "&dc_fan_status=" + String(digitalRead(dc_fan))+  "&inv_fan_status=" +
inverter_status+  "&inverter_temp=" + inverter_temp+  "&inverter_status=" + inverter_status+  "&blackout=" +
String(power_blackout)+  "&door_status=" + String(digitalRead(reed_switch))+  "&cardStatus=" +
String(cardStatus);

// now encode and send data to nodemcu
serializeJson(doc, Serial3);
//Serial.print("mega2nodemcu data: ");Serial.println(mega2nodemcu);
// deserialize data which was jsonified

//serializeJson(doc, mega2nodemcu);
//Serial.print("mega2nodemcu data: ");Serial.println(mega2nodemcu);
Serial.println("data sent from arduino to nodemcu...");
/*
  String payload =  "{\"ds18b20_temp\":\""+String(ds18b20_temp)+ "\"," +
  "\"dht22_temp\":\""+String(dht22_temp)+ "\",";
 // "\"}\n";
//payload = "I am Benson Mbuya a fighter";
// send data from arduino via serial communication
send2nodemcu(payload);
payload="\"heat_index\":\""+String(heat_index)+ "\"," +
"\"dht22_humidity\":\""+String(dht22_humidity)+"\",";
send2nodemcu(payload);
payload="\"load1_status\":\""+load1_status+ "\"," +
"\"load2_status\":\""+load2_status+"\","+
"\"load3_status\":\""+load3_status+"}";
send2nodemcu(payload);
send2nodemcu("\n");
*/
 /* String payload =  "{\"ds18b20_temp\":"+String(ds18b20_temp)+ "\"," +
  "\"dht22_temp\":\""+String(dht22_temp)+ "\"," +
  "\"heat_index\":\""+String(heat_index)+ "\"," +
  "\"dht22_humidity\":\""+String(dht22_humidity)+"\","+
  "\"load1_status\":\""+load1_status+"\","+load2_status+"\","+load3_status+ "\"," + String(ac_voltage)+ "\","+
String(currentvalue)+  "\","+String(frequency)+ "\","  + String(pv_volt)  + "\"}\n";
*/
  post = 0;
  }
  }

// function below was abandoned
void send2nodemcu(String payload){
  char buf[datasize];
  //sprintf(buf,"%s", "Hello I am nodemcu\n");//this has the same effect as using toCharArray method
  payload.toCharArray(buf, datasize);
  Serial3.write(buf);//Serial.write only sends characters not strings convert to character first
  // No need to read data from nodemcu since signal volts is 3.3v it is received mangled
  }

// function that executes whenever data is received from master
void receiveEvent(int howMany) {
  received_i2c_data = ""; //clear buffer
 while (0 <Wire.available()) {
    char character = Wire.read();      /* receive byte as a character */
    Serial.print(character);          /* print the character */
    received_i2c_data.concat(character);
  }
 Serial.println();            /* to newline */
}
```

```
// function that executes whenever data is requested from master
void requestEvent() {
//String mega2nodemcu =  "my name is Benson"+ "," + " Mbuya";
    //String payload = "Hello I am nodemcu\n";
    // String payload = String(load1_status)+String(", Mbuya\n");
   // String payload =  String(ds18b20_temp) + "," + String(dht22_temp)+ "," + String(heat_index)+  "," +
String(dht22_humidity)+","+load1_status+","+load2_status+","+load3_status+ "," + String(ac_voltage)+ "," +
String(currentvalue)+  ","+String(frequency) + "," + String(inv_current_value)+"," + String(pv_volt)+  "," +
String(pv_current)+  "," + String(charged_percent)+  "," + String(charging_status)+  "," + String(bat_volt)+  ","
+ String(person)+  "," + String(dc_fan)+  "," + inverter_status+  "," + inverter_temp+  "," + inverter_status+  ","
+ String(power_blackout)+  "," + String(reed_switch)+  "," + String(cardStatus)+"\n";
 // you can only transmit 11 items...safer to only transmit 10 items
// "{\"benson\":\"mbuya\"}";
 // String payload = "{\"ds18b20_temp\":\""+ String(ds18b20_temp) + "\"}\n";
 // a response to nodemcu to indicate that data it sent was received
   String payload = "{\"response\":\""+ String("ok") + "\"}\n";
 char buf[datasize];
  //sprintf(buf,"%s", "Hello I am nodemcu\n");//this has the same effect as using toCharArray method
  payload.toCharArray(buf, datasize);
  //sprintf(buf,"%s\n",mega2nodemcu);//this has the same effect as using toCharArray method
 Wire.write(buf);  /* sends string to arduino */
// Wire.write("Hello I am arduino uno\n");  /*send string on request, important to include end of line escape */
// Wire.write(printTime);  /* sends string to arduino */
}

void wattmeter(){
   float min_max=0,vpp=0,voltage_peak_value=0,voltage_min_value=0;
int avg=0;
  /// clear variables
  current_peak_value=0;current_min_value=0;
    pv_current_peak_value=0;pv_current_min_value=0;
     inv_current_peak_value=0;inv_current_min_value=0;
  voltage_peak_value=0; ac_voltage =0;

  // ********** PV voltage measurement ********************
  adc_value = analogRead(pv_volt_sensor);
  //Serial.print("pv adc: ");  Serial.println(adc_value,3);
   vin = (adc_value*5)/1024;
  // Serial.print("pv vin: ");  Serial.println(vin,3);
   pv_volt = vin / (R2/(R1+R2)); // formula for calculating voltage in i.e. GND
   if (pv_volt<0.09)//condition
       pv_volt=0.00;//statement to quash undesired reading !

        pvCounter = pvCounter + 1;
        PVavg = PVavg + pv_volt;
        pv_volt = PVavg/pvCounter;

  // change the analog out value:
 // Serial.print("battery voltage: ");   Serial.println(bat_volt,1);
 //Serial.print("PV voltage: ");   Serial.println(pv_volt,1);
  lcd.setCursor(6, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
   lcd.print("PV"); lcd.print(pv_volt,1);  lcd.print("V"); // Battery voltage


  // ********** AC voltage , Current, & frequency measurement ********************


  //---- current sensor operations -------------------------
  for (int k = 0; k < sampling; k++) {
  adc_value = analogRead(currentPin);
  //Serial.print("%%%%% ac current adc: ");Serial.println(adc_value);
 ac_value = analogRead(ac_sensor);
```

```
 inv_value = analogRead(inv_current);

pv_value = analogRead( pv_current_sensor);

if(ac_value>voltage_peak_value)
voltage_peak_value=ac_value;
if(ac_value<voltage_min_value)
voltage_min_value=ac_value;
if(adc_value > 300)
 {
  current_peak_value+=adc_value;
   avg=avg+1;
 }
 if(adc_value<current_min_value)
 current_min_value=adc_value;
 if(pv_value>pv_current_peak_value)
  pv_current_peak_value=pv_value;
 if(pv_value<pv_current_min_value)
 pv_current_min_value=pv_value;
  if(inv_value>inv_current_peak_value)
  inv_current_peak_value=inv_value;
 if(inv_value<inv_current_min_value)
 inv_current_min_value=inv_value;
    delayMicroseconds(200); // let ADC settle before next sample 3ms

  }

 //Serial.print("****adc value: ");Serial.println(adc_value);
  min_max = current_peak_value/avg;
        //    Serial.print("%%%%% current_peak_value: ");Serial.println(current_peak_value);
        //   Serial.print("%%%%% avg: ");Serial.println(avg);
         //           Serial.print("%%%%% ac current adc: ");Serial.println(min_max);
  if(min_max>480){
  vpp = (min_max* 5000)/1024.0; // currentvalue = (( offsetVoltage-adcVoltage) / sensitivity);
  currentvalue = abs((vpp-offsetVoltage)/sensitivity); //sensitivity(mVperAmp);
   //currentvalue = (vpp-offsetVoltage)/sensitivity;
   // currentvalue = (offsetVoltage-vpp)/sensitivity; //sensitivity(mVperAmp);
  // Serial.print("####ac current: ");Serial.println(currentvalue);
   if(currentvalue>1)
   currentvalue = currentvalue * currentOffset;
   else if (currentvalue<1 && currentvalue>0.1){
    currentvalue = currentvalue;
    }
  }
  else{
  // Serial.print("####resulting current is too small ... less than 0.1A ");
  }//currentvalue = currentvalue * currentOffset;

  //currentvalue=abs(currentvalue);
  currentvalue = pzem.current();

  //current_avg = current_avg + 1;
  //current_buffer = current_buffer + currentvalue;
  //currentvalue=current_buffer /current_avg;


   // inv_current_value = ((((inv_current_peak_value-inv_current_min_value)* 5000)/1024.0)-
673.828125)/sensitivity;
     inv_current_value = (((inv_current_peak_value* 5000)/1024.0)-OFFset_inv1)/sensitivity;
  //     Serial.print("%%%%% inv adc: ");Serial.println(inv_current_peak_value);
  // Serial.print("####inv current: ");Serial.println(inv_current_value);
      //if(tanesco_charger_status=="1")
      inv_current_value = abs(inv_current_value*currentOffset);
```

```
      //inv_current_value = abs(inv_current_value);
 //Serial.print("$$$$$ offset inv current: ");Serial.println(inv_current_value);
  // Serial.print("***** tanesco_charger_status: ");Serial.println(tanesco_charger_status);
         // iCurrentCounter = iCurrentCounter + 1;
        //iCurrentavg = iCurrentavg + inv_current_value;
        //inv_current_value = iCurrentavg/iCurrentavg;

  if(inv_current_value<0.005)
   inv_current_value=0;

     pv_current = ((((pv_current_peak_value-pv_current_min_value)* 5000)/1024.0)-offsetVoltage)/sensitivity;
//sensitivity(mVperAmp);
   if(pv_current<0.005)
   pv_current=0;

    //analogWrite(dc_fan, 250);        // turn cooling fan ON
    //digitalWrite(dc_fan, HIGH);
   if(pv_current > 1 ||ds18b20_temp > 28 )
   {
    digitalWrite(dc_fan, HIGH);  // turn cooling fan ON
    //analogWrite(dc_fan, 150);        // turn cooling fan ON
    }
    else
    {
     digitalWrite(dc_fan, LOW);  // turn cooling fan OFF
     // analogWrite(dc_fan,0);
    }

 if(currentvalue<0.005 || currentvalue==NAN )
   currentvalue=0;

       lcd.setCursor(7, 2); //Set the cursor on the third column and the second row (counting starts at 0!).
   lcd.print("PV"); lcd.print(pv_current,1);  lcd.print("A "); // Battery voltage

  //ac_voltage = ((voltage_peak_value-voltage_min_value) * OFFset_ac1)-OFFset_ac2;
 ac_voltage = voltage_peak_value * OFFset_ac1;
 ac_voltage = pzem.voltage();
 //ac_avg = ac_avg + 1;
 //acVoltage = acVoltage + ac_voltage;
 //ac_voltage = acVoltage/ac_avg;

if(ac_voltage<100 || ac_voltage==NAN ){
 ac_voltage=0;
      digitalWrite(currentSensor, LOW); // turn current sensor module OFF
 }
 else{
   digitalWrite(currentSensor, HIGH); // turn current sensor module ON
  }

     if(heartbeat==1){
lcd.setCursor(7,3);  lcd.print(currentvalue,2); lcd.print("A   ");
 lcd.setCursor(0,3);  lcd.print(frequency,1); lcd.print("Hz ");
  }
    else{
       lcd.setCursor(7, 3); //Set the cursor on the third column and the second row (counting starts at 0!).
   lcd.print("i"); lcd.print(inv_current_value,2);  lcd.print("A "); // Battery voltage
     lcd.setCursor(0,3);  lcd.print(ac_voltage,1); lcd.print("V ");
     }

//  Serial.print("ac sensor adc value:");
//Serial.println(voltage_peak_value);
//Serial.print("ac voltage value:");
```

```
// Serial.println(ac_voltage);
//analogWrite(pv_pwm, 255);
digitalWrite(pv_currentSensor, HIGH);
  if( pv_charger_flag==1){
    //tanesco_charger_status="0";
 // digitalWrite(LED_BUILTIN, HIGH); // turn ON load1
  //      Serial.println("turning pv_charger_status ON ");
       if(pv_charging_counter<chon){

   if((bat_volt < 12.5)&&(bat_volt > 0)  && (pv_volt > bat_volt)){
  //analogWrite(pv_pwm, 242.25);        // boost charging
  analogWrite(pv_pwm, 255);        // boost charging
  digitalWrite(charging_status,HIGH);
      digitalWrite(pv_currentSensor, HIGH);  // turn current sensor module ON
      }
else if((bat_volt < 13.5)&&(bat_volt > 12.5)  && (pv_volt > bat_volt)){
  analogWrite(pv_pwm, 255);    // float charging
  //analogWrite(pv_pwm, 255);
  digitalWrite(charging_status,HIGH);
      digitalWrite(pv_currentSensor, HIGH);  // turn current sensor module ON
      }
      else if((bat_volt < 14.5)&&(bat_volt > 13.5)  && (pv_volt > bat_volt)){
  analogWrite(pv_pwm, 5);    // float charging
  //analogWrite(pv_pwm, 255);
  digitalWrite(charging_status,HIGH);
      digitalWrite(pv_currentSensor, HIGH);  // turn current sensor module ON
      }
else if ((bat_volt > 14.5) or (pv_volt < bat_volt) or eveningFlag==1)
{
  analogWrite(pv_pwm,0);
//analogWrite(pv_pwm, 255);
digitalWrite(charging_status,LOW); // green LED will off as no charging is done during this time
digitalWrite(pv_currentSensor, LOW);
 pv_current=0;
}

}
if (pv_charging_counter>chon && pv_charging_counter<choff){
   analogWrite(pv_pwm,0);
   digitalWrite(pv_currentSensor, LOW);
   digitalWrite(charging_status,LOW);
            pv_current=0;
            ac_avg = 0;
            acVoltage = 0;
            pvCounter = 0;
            PVavg = 0;
            current_buffer =0;
            current_avg=0;
            iCurrentCounter = 0;
            iCurrentavg = 0;
    measure_battery();

  }
  else if (pv_charging_counter>choff){
   pv_charging_counter=0; // reset counter

    }
       }
       else if( pv_charger_flag==0){
       //    digitalWrite(LED_BUILTIN, LOW); // turn OFF load1
    //        Serial.println("turning pv_charger_status OFF ");
            analogWrite(pv_pwm,0);
            digitalWrite(pv_currentSensor, LOW);
```

```
            digitalWrite(charging_status,LOW);
            pv_current=0;
            measure_battery();

//digitalWrite(charging_status,LOW); // green LED will off as no charging is done during this time
//digitalWrite(pv_currentSensor, LOW);
          }


  POWER = ac_voltage * currentvalue;
//    Serial.print(" POWER = ");
//  Serial.println(POWER);
  //lcd.setCursor(0,1);
  //lcd.print("Current =       ");
 // lcd.clear();

   //   Serial.print("chon :");Serial.print(chon);Serial.print(" choff : ");Serial.println(choff);

  }

//******************* end of functions *****************************************
void setup() {
    // initialize digital pin LED_BUILTIN as an output. It'll act as heartbeat to indicate program is running
smoothly

  pinMode(wifiModule_enable, OUTPUT); // turns on/off wifimodule at startup
   digitalWrite(wifiModule_enable, HIGH); // Turn wifi module off to permit mega to go on first
  pinMode(acPowerStatus, INPUT);
  pinMode(heartBeatpin, OUTPUT);
  pinMode(load3, OUTPUT);      // lights, epson printer, radio amplifier
  pinMode(load2, OUTPUT);      // cctv
  pinMode(load1, OUTPUT);      // tanesco loads: canon photocopier, microwave machine
  pinMode(charging_status, OUTPUT);      // declare LED as output
  pinMode(currentSensor, OUTPUT);      // declare current sensor 5V wire as output
  pinMode(pv_currentSensor, OUTPUT);
  pinMode(pv_pwm, OUTPUT);
  pinMode(dc_fan,OUTPUT);
   pinMode(inv_enable,OUTPUT);
   digitalWrite(inv_enable,LOW); // allow inverter power through
  pinMode(inputPin, INPUT);     // declare PIR sensor on interrupt pin 2 as input
  pinMode(ac_sensor,INPUT); // set pin a1 as input pin
  pinMode(currentPin,INPUT); // set pin a1 as input pin
  pinMode(bat_volt_sensor,INPUT); // set pin a1 as input pin
   pinMode(reed_switch, INPUT);        //Sets digital pin 13 as output pin
    pinMode(inv_current,INPUT); // set pin a1 as input pin
     pinMode(pv_current_sensor,INPUT); // set pin a1 as input pin
      pinMode(pv_volt_sensor,INPUT); // set pin a1 as input pin

  attachInterrupt(digitalPinToInterrupt(inputPin), pir_isr, RISING);

  //bluetooth.begin(baudrate);
  Serial.begin(baudrate);
  Serial.println("starting...");
  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(baudrate);
//Begin serial communication with Arduino and nodemcu
 Serial3.begin(baudrate);
 //nodeSerial.begin(9600);
 // attachInterrupt(digitalPinToInterrupt(rxPin), bluetooth_interrupt, CHANGE);

  // frequency measurement setup
  //  attachInterrupt(digitalPinToInterrupt(frequency_count_pin), frequency_counter, RISING);
    // Timer1 module configuration  for frequency measurement
```

```
  TCCR1A = 0;
  TCCR1B = 2;   // enable Timer1 module with 1/8 prescaler ( 2 ticks every 1 us)
  TCNT1  = 0;   // Set Timer1 preload value to 0 (reset)
  TIMSK1 = 1;   // enable Timer1 overflow interrupt
  EIFR |= 1;  // clear INT0 flag
  attachInterrupt(digitalPinToInterrupt(frequency_count_pin), frequency_isr, FALLING);  // enable external
interrupt (INT0)

   delay(1000); // wait for console opening

  dht.begin();

  // Start up the dallas ds18b20 temperature library
  sensors.begin();

// initialize i2c for sending data to and from wifi module
/*   Wire.begin(i2c_address);   //join i2c bus with address 8
// function that executes whenever data is received from master (nodemcu)
 Wire.onReceive(receiveEvent); // register receive event
 Wire.onRequest(requestEvent); */

  // Initiate the LCD:
  delay(1000);
  lcd.init();
  lcd.backlight();
    // Print 'Hello World!' on the first line of the LCD:
  lcd.setCursor(0, 0); // Set the cursor on the first column and first row.
  lcd.print("Smart Energy Manager"); // Print the string "Hello World!"
  lcd.setCursor(2, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
  lcd.print("Version 1.0");

/* mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+255757541412\"");//change ZZ with country code and xxxxxxxxxxx with
phone number to sms
  updateSerial();
  mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content
  updateSerial();
  mySerial.write(26);*/

  delay(1000);
  lcd.clear();
      if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
   // while (1);
    }
// synchronize with PC time
 // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

measure_battery();
 Serial.println("Finished setup...");
  digitalWrite(wifiModule_enable, HIGH); // Now turn wifimodule ON
}


void frequency_isr() {
  tmr1 = TCNT1;
  TCNT1  = 0;   // reset Timer1
}
```

```
ISR(TIMER1_OVF_vect) {  // Timer1 interrupt service routine (ISR)
  tmr1 = 0;             // related to frequency counter routine
}

void frequency_counter(){
   unsigned long current_acPower_poll = millis();
  if ( digitalRead(acPowerStatus) == LOW && ac_voltage>170) {            // check if the ac power is ON
    // Serial.println("AC power line is ON");
     // Serial.println(ac_voltage);
    //digitalWrite(load3, HIGH);                  // turn ON indicator light
     power_blackout = 0;
     blackout_flag = 0;
     previous_acPower_poll = current_acPower_poll;
  }
  else
  {
     if(blackout_flag>10)
     power_blackout = 1;
    blackout_flag++;


    }
// if ac power has been off for more than 1sec then indicate that it's trully off
  if( current_acPower_poll-previous_acPower_poll >= poll_acPower_interval && power_blackout == 1){

    //  Serial.println("AC power line is OFF");
    previous_acPower_poll = current_acPower_poll;
    }
  // save current Timer1 value
  uint16_t value = tmr1;
  // calculate signal period in milliseconds
  // 8.0 is Timer1 prescaler and 16000 = MCU_CLK/1000
  period = 8.0 * value/16000;
  // calculate signal frequency which is = 1/period ; or = MCU_CLK/(Prescaler * Timer_Value)
  if(value>20000){
    frequency = 16000000.0/(8UL*value);
    // Serial.print("tmr1 value:"); Serial.println(tmr1);
    //  Serial.print("period:"); Serial.println(period);
    // Serial.print("frequency:");Serial.println(frequency);
    if(ac_voltage<10 && frequency > 60){
      frequency=frequency*1.38;//this measures frequency of tanesco and inverter
      frequency = pzem.frequency(); // this only measures tanesco frequency
      }

  }
   value = 0; period = 0; //frequency = 0;

  }
void loop() {
  // constant communication between nodemcu and arduino
  //Serial.println("main loop...");
 load_controller();
 arduino2nodemcu();
 nodemcu2arduino();
/*if(post==1)
nodemcu2arduino();
else if(post==0)
arduino2nodemcu(); */
 // bluetooth_interrupt();
    load_controller();


  unsigned long currentMillis = millis();
```

```
  unsigned long currentMillis_pir = millis(); // track pir...ensure enough time elapsed without motion before
turning off applieances
   unsigned long current_pir_poll = millis();
  if( current_pir_poll-previous_pir_poll >= poll_pir_interval){
    measure_battery();
    heart_beat();
    showTime();
    pir();
    wattmeter();
    weather_info();
    frequency_counter();
     arduino2nodemcu();
 nodemcu2arduino();
/*if(post==1)
nodemcu2arduino();
else if(post==0)
arduino2nodemcu(); */
    pv_charging_counter++;
   // updateSerial();
   previous_pir_poll = current_pir_poll;

    }
    // ---------------------- interval for data logging ----------------------------------
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    //data2sdCard();
    previousMillis = currentMillis;
    showTime();
   // wattmeter();
    pir();
  //updateSerial();

      // leave enough time to elapse then check if room is vacant then turn off appliances
   DateTime pir_tracker = rtc.now();
   currentMillis_pir = pir_tracker .minute();
 if (currentMillis_pir - previousMillis_pir >= interval_pir && wait_time == true) {
 previousMillis_pir = currentMillis_pir;
 lcd.noBacklight();
 if(wait_more_time==true)
{
  wait_time=true;     // wait 5 more minutes
  wait_more_time = false; // clear waiting period flag
}
else if(wait_more_time==false)
{
  wait_time = false;
appliance = false;
 lcd.noBacklight();
}
 // Serial.println("TURN LOAD ON/OFF");
      //pir();
  }
  }


}
//########################## end of void() ##################################################

 void load_controller(){
  // if wifimodule is off turn it ON
  if(loadControl_flag=="reset_wifi"){
  digitalWrite(wifiModule_enable, LOW); // Now turn wifimodule ON
  delay(3000); // wait 3seconds
```

```
  digitalWrite(wifiModule_enable, HIGH); // Now turn wifimodule ON
  loadControl_flag="load1_on";
  }
 if(loadControl_flag=="load1_on"){ digitalWrite(load1, LOW); }
  else if(loadControl_flag=="load1_off"){digitalWrite(load1, HIGH);}
  else if(loadControl_flag=="load2_on"){digitalWrite(load2, HIGH);}
  else if(loadControl_flag=="load2_off"){digitalWrite(load2, LOW);}
  else if(loadControl_flag=="load3_on"){digitalWrite(load3, LightON);}
  else if(loadControl_flag=="load3_off"){digitalWrite(load3, LightOFF);}
  else if(loadControl_flag=="pv_on"){ pv_charger_flag=1;}
  else if(loadControl_flag=="pv_off"){pv_charger_flag=0;}
  else{
    if(pv_charger_status=="1"){
  pv_charger_flag=1;
  //Serial.println("**** pv_charger_status is now ON ");
  }
else if(pv_charger_status=="0"){
  pv_charger_flag=0;
  // Serial.println("**** pv_charger_status is now OFF ");
  }
  if( load1_status=="1"){
  digitalWrite(load1, LOW); // turn ON load1, load1 uses inverted logic where off is HIGh and on is LOW
      //  Serial.println("turning load1 ON ");
        }
        else if( load1_status=="0"){
           digitalWrite(load1, HIGH); // turn OFF load1
        //    Serial.println("turning load1 OFF ");
         }
      if( load2_status=="1"){
  digitalWrite(load2, HIGH); // turn ON load1
      //  Serial.println("turning load2 ON ");
        }
        else if( load2_status=="0"){
           digitalWrite(load2, LOW); // turn OFF load1
        //     Serial.println("turning load2 OFF ");
          }
           if( load3_status=="1" ){
  digitalWrite(load3, LightON); // turn ON load1
    //    Serial.println("turning load3 ON ");
        }
        else if( load3_status=="0"){
           digitalWrite(load3, LightOFF); // turn OFF load1
        //    Serial.println("turning load3 OFF ");
         }
    }

    }


void pir_isr() {
  motion = digitalRead(inputPin);  // read input motionue
// clear counter flag for turning appliances OFF ... i.e wait 10 more minutes from now
  previousMillis_pir = currentMillis_pir;
}

void pir(){
 // Serial.print("reed_switch status: ");Serial.println( digitalRead(reed_switch));  // check input again
  motion = digitalRead(inputPin);  // check input again

  if ( motion == HIGH || digitalRead(reed_switch)) {            // check if the input is HIGH
    if(!digitalRead(load3))         // if load3 are currently OFF
    {
   //digitalWrite(load3, HIGH);  // turn load3/Lights ON
```

```
   person = 1;                // Room occupied
    wait_time = true;          // flag to help check room occupation for 5min interval
   }
   // Serial.println("Motion detected!");
      //lcd.clear();
       person = 1;               // Room occupied

      if(wait_time==true)
      wait_more_time = 1;   // flag to show motion was detected during waiting period

 }

   if(!digitalRead(reed_switch)||motion == LOW && wait_time==false && appliance == false){

     lcd.noBacklight();
    // Serial.println("Motion ended!");
    // Serial.println(digitalRead(currentSensor));
     //lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("P_OFF"); // print message at (0, 1)
     person = 0;          // room is vacant i.e no person detected in past 5 min
     //lcd.clear();
     }
      //if(digitalRead(reed_switch))
 if( person==1 || digitalRead(reed_switch))
 {
      lcd.setCursor(0, 0);         // move cursor to   (0, 1)
      lcd.print("PR ON "); // print message at (0, 1)
      if(backlight_timer<120 && person==1){
      lcd.backlight();
      backlight_timer++;
      }
      else if (backlight_timer>=120 && person==1)
      lcd.noBacklight();

      load3_status="1";
   }
 // else if(!digitalRead(reed_switch))
   else if(person==0 || !digitalRead(reed_switch))
   {
     backlight_timer=0;
current_peak_value=0;
     lcd.setCursor(0, 0);
     lcd.print("PR OFF"); // print message at (0, 1)
     load3_status="0";

     }


}


  void measure_battery(){
      // ********** Battery voltage measurement **********
 adc_value = analogRead(bat_volt_sensor);
// Serial.print("***Battery adc: ");  Serial.println(adc_value,3);
  vin = (adc_value*5)/1024;
  ///Serial.print("---Battery vin: ");  Serial.println(vin,3);
  bat_volt = vin / (R2/(R1+R2)); // formula for calculating voltage in i.e. GND
  bat_volt = bat_volt - bat_offset;
//   Serial.print("battery voltage: "); Serial.println(bat_volt,1);
  if (bat_volt<0.09)//condition
       bat_volt=0.00;//statement to quash undesired reading !
```

```
  // Serial.print("PWM value: "); Serial.println(100);
  lcd.setCursor(0, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
 //lcd.print("B");
 lcd.print(bat_volt,1);  lcd.print("V "); // Battery voltage
//Serial.print("bat_volt: ");Serial.println(bat_volt);
 charged_percent=map(bat_volt*10, 115 , 130, 0, 100);
 if(charged_percent<0)
 charged_percent=0;
 else if(charged_percent>100)
 charged_percent=100;
 //Serial.print("charged_percent: ");Serial.println(charged_percent);
  lcd.setCursor(0, 2); //Set the cursor on the third column and the second row (counting starts at 0!).
 lcd.print("soc"); lcd.print(charged_percent,1);   lcd.print("%");
    }

  void showTime(){
   DateTime now = rtc.now();
//    Serial.println("Current Date & Time: ");
//    Serial.print(now.year(), DEC);
    //lcd.clear();
    lcd.setCursor(6, 0);
    //lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);
//    lcd.print(now.year()-2000);
    char printTime[17];
    sprintf (printTime, "%02d %02d:%02d",now.day(), now.hour(), now.minute());
    lcd.print(printTime);
    //lcd.print(now.second());

// stop charging in the evening and turnoff

if(now.hour()>=19)
    eveningFlag=1;
    else
    eveningFlag=0;

  }
void weather_info(){
  // Wait a few seconds between measurements.
  //delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
   dht22_humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
    dht22_temp = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
 // float f = dht.readTemperature(true);
  // Compute heat index in Celsius (isFahreheit = false)
   heat_index = dht.computeHeatIndex(dht22_temp, dht22_humidity, false);
  // Check if any reads failed and exit early (to try again).
  if (isnan(dht22_humidity) || isnan(dht22_temp)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    dht.begin();
    //return;
  }
//  Serial.print(F("Humidity: "));
//  Serial.print(h);
//  Serial.print(F("%  Temperature: "));
//  Serial.print(t);
//  Serial.print(F("°C "));
//  Serial.print(f);
//  Serial.print(F("°F  Heat index: "));
//  Serial.print(hic);
```

```
//  Serial.print(F("°C "));

 sensors.requestTemperatures(); // Send the command to get ds18b20 temperatures
  // After we got the temperatures, we can print them here.
  // We use the function ByIndex, and as an example get the temperature from the first sensor only.
  ds18b20_temp = sensors.getTempCByIndex(0);

  lcd.setCursor(15, 0); lcd.print(dht22_humidity,1);  lcd.print("%");
  lcd.setCursor(14, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
  lcd.print(ds18b20_temp,1); lcd.print(char(223)); lcd.print("C"); // ds18b20 temperature
   lcd.setCursor(14, 2);lcd.print(dht22_temp,1); lcd.print(char(223)); lcd.print("C");  // dht22 temperature
   lcd.setCursor(14, 3);lcd.print(heat_index,1);  lcd.print(char(223));lcd.print("C"); // dht22 heat index
temperature

  }

 void heart_beat()  {
    heartbeat = ! heartbeat; // toggle the builtin led (pin 13) variable
   // digitalWrite(LED_BUILTIN, heartbeat);   // toggle the builtin led (pin 13)
   // use pwm since LED at pin 13 has no limiting resistor
    if(heartbeat==1)
    digitalWrite(heartBeatpin,1);
    else
    digitalWrite(heartBeatpin,0);
    }


void data2sdCard(){
   SDstatus = 0;
// ********** begin sd card code section ****************************

// first check if sd card is not full before writing to it
Serial.println("\nInitializing SD card...");
  if (!card.init(SPI_HALF_SPEED, 4)) {
    Serial.println("initialization failed!");
   // while (1);
  }

// check sd card size
if (!volume.init(card)) {
    Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've formatted the card");
    //while (1);
  }
  // determine size of the card
  uint32_t volumesize;
  volumesize = volume.blocksPerCluster();    // clusters are collections of blocks
  volumesize *= volume.clusterCount();       // we'll have a lot of clusters
  volumesize /= 2;                           // SD card blocks are always 512 bytes (2 blocks are 1KB)
//  Serial.print("Volume size (Kb):  ");
//  Serial.println(volumesize);
  Serial.print("Volume size (Mb):  ");
  volumesize /= 1024;
  Serial.println(volumesize);
// uncomment the section below to see all the files in the SD card
//  Serial.println("\nFiles found on the card (name, date and size in bytes): ");
  //root.openRoot(volume);
  // list all files in the card with date and size
  //root.ls(LS_R | LS_DATE | LS_SIZE);
 // root.ls(LS_R | LS_SIZE);
 // root.close();

if(!SD.begin(4)){
   Serial.println("initialization failed. Things to check:");
```

```
  }
  else{
myFile = SD.open("/");
SDstatus=printDirectory(myFile, 0);
Serial.print("Total file size is (bytes): ");Serial.println(SDstatus);
SDstatus/=1024; //convert to Kb
Serial.print("Total file size is (Kb): ");Serial.println(SDstatus);
SDstatus=SDstatus/1024; // convert to MB
Serial.print("Total file size is (MB): ");Serial.println(SDstatus);
// if card is more than 90% full then wipe it...delete all files
cardStatus = (SDstatus/volumesize)*100;
Serial.print("card is  ");Serial.print(cardStatus,5 );Serial.println("% full ...");

if((float)SDstatus/volumesize > 0.9){
  myFile = SD.open("/");
  SDwipe(myFile, 0);
}
//SDwiper(myFile, 0);
myFile.close();
  }

// sd card can now be written to after confirming its not full or wiping it
   DateTime now = rtc.now();
// create file name corresponding to today's date
    char FileName[17];
    sprintf (FileName, "%04d%02d%02d.csv", now.year(), now.month(), now.day());
 //   Serial.print("file name is ");Serial.println(String(FileName));

//if (!SD.begin(4)) {
//    Serial.println("initialization failed. Things to check:");
//    //while (1);
//  }
//    // re-open the file for checking file size:
//   myFile = SD.open(String(FileName));
//    // check current file size
//  Serial.print("file size (Bytes) is ");Serial.println(myFile.size());
//  Serial.print("file size (kB) is ");Serial.println(float(myFile.size()/1024));
//  Serial.print("file size (MB) is ");Serial.println(float(myFile.size()/(1024*1024)));
//myFile.close();
// create time stamp in order to insert along with record
    char timeStamp[17];
    sprintf (timeStamp, "%04d%02d%02d%02d%02d%02d", now.year(), now.month(), now.day(), now.hour(),
now.minute(), now.second());
 // prep data to be recorded
dataString =  String(timeStamp) +String(ds18b20_temp) + ","+ String(dht22_temp)+ "," + String(heat_index)+
"," + String(dht22_humidity)+","+load1_status+","+load2_status+","+load3_status+ "," + String(ac_voltage)+
"," + String(currentvalue)+ "," +String(frequency) + "," + String(inv_current_value)+ "," + String(pv_volt)+  ","
+ String(pv_current)+  ","+ String(charged_percent)+  "," + String(digitalRead(charging_status))+  "," +
String(bat_volt)+  "," + String(person)+  ","+ String(digitalRead(dc_fan))+  "," + inverter_status+  "," +
inverter_temp+  "," + inverter_status+  "," + String(power_blackout)+  "," + String(digitalRead(reed_switch));
  if (!SD.begin(4)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your shield or module?");
    //while (1);
  }
  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
   myFile = SD.open(String(FileName), FILE_WRITE);
  // if the file opened okay, write to it:
  if (myFile) {
    myFile.println(dataString);
```

```
    // close the file:
    myFile.close();
    Serial.println("done writing to "+String(FileName)+" ...");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error writing to "+String(FileName));
  }


// code to read content of recorded file
//  if (myFile) {
//    // read from the file until there's nothing else in it:
//    while (myFile.available()) {
//      Serial.write(myFile.read());
//    }
//    // close the file:
//    myFile.close();
//  } else {
//    // if the file didn't open, print an error:
//    Serial.println("error reading "+ String(FileName));
//  }

}

int printDirectory(File dir, int numTabs){
  int fileSize = 0;
  while (true)
  {
    File entry = dir.openNextFile();
    if (! entry)
    {
      if (numTabs == 0){
        Serial.println("******* finished checking sd occupied space size **********");
        return fileSize;
        }
      return;
    }

    for (uint8_t i = 0; i < numTabs; i++)
    Serial.print('\t');
    Serial.println(entry.name());
    if (entry.isDirectory())
    {
      printDirectory(entry, numTabs + 1);
    }
    else
    {
      fileSize+=int(entry.size());
    }
    entry.close();
  }
}


  void SDwipe(File dir, int numTabs){
  while (true)
  {
    File entry = dir.openNextFile();
    if (! entry)
    {
      if (numTabs == 0){
        Serial.println("*****************sd card wipe finished! *****************");
        }
```

```
      return;
    }

    for (uint8_t i = 0; i < numTabs; i++)
      Serial.print('\t');
   if(String(entry.name())!="SYSTEM~1" && String(entry.name())!="WPSETT~1.DAT" &&
String(entry.name())!="INDEXE~1"){
         // delete the file
          Serial.println("deleted "+String(entry.name()))  ;
    SD.remove(String(entry.name()));
      }
    entry.close();
  }
}




//************** end of sd card code ******************************

/*void updateSerial(){
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
  }
}*/
/* void bluetooth_interrupt(){
while(bluetooth.available()){
    Serial.println("bluetooth routine");
       character = bluetooth.read();
    bluetooth_data.concat(character);
  // Serial.print(bluetooth_data);
   if(character == '\n'){ // if end of bluetooth_data received
     bluetooth_data.trim(); // remove whitespaces
     Serial.println(bluetooth_data); //display bluetooth_data and
     //Serial.print(bluetooth_data.length());
     if(bluetooth_data == "ben")
     {
      // take action 1
      Serial.println("bluetooth action 1");
     //digitalWrite(LED_BUILTIN, HIGH);
     }
     else if(bluetooth_data == "mbuya")
     {
      // take action 1
      Serial.println("bluetooth action 2");
     //digitalWrite(LED_BUILTIN, LOW);
     }
     bluetooth_data = ""; //clear buffer
     //Serial.println();
  }
  }
  }*/
```

Wifi module source code

```
#include <WiFiClient.h>
#include <WiFiClientSecure.h>
```

```
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFiMulti.h>

#include <SoftwareSerial.h>
SoftwareSerial esp2mega(D6,D5);// // (Rx, Tx) In NodeMCU we shall make pin 5 as Tx and pin 6 as Rx.
#include <ArduinoJson.h>
ESP8266WiFiMulti wifiMulti;
// --------  station mode --------------------
const char* ssid1 = "BENnet";    // Enter SSID here
const char* password1 = "Mbuya2020";  // Enter Password here
const char* ssid2 = "Techcentral";    // Enter SSID here
const char* password2 = "mbuya2020";  // Enter Password here
const char* host = "https://iot.techcentrall.com";

const int httpsPort = 443;

//SHA1 finger print of certificate use web browser to view and copy
const char fingerprint[] PROGMEM = "E1:09:9B:EB:10:AE:36:C7:5F:0F:3B:4C:09:0E:4E:28:6E:B1:FB:13";

//----- for thingsboard -------
#include <ESP8266WiFi.h>
#include <ThingsBoard.h>
#define TOKEN "sURL0pLpenFPWcGiLHtX"
char thingsboardServer[] = "demo.thingsboard.io";
int status = WL_IDLE_STATUS;

// headers for gsheet communication
#include "HTTPSRedirect.h"
#include "DebugMacros.h"

const char* spreadsheethost = "script.google.com";
const char *GScriptId = "AKfycbw6jZw5oo9HQL7WTZDQqRRCeglRoImS6ejAQ6-
rSs6SvSID9uRd9S_rJfLnynw5Cn8H"; // Replace with your own google script id

// echo | openssl s_gsclient -connect script.google.com:443 |& openssl x509 -fingerprint -noout
//const char* fingerprint = "";

String url = String("/macros/s/") + GScriptId + "/exec?value=ds18b20_temp";  // Write Teperature to Google
Spreadsheet at cell A1
// Fetch Google Calendar events for 1 week ahead
String url2 = String("/macros/s/") + GScriptId + "/exec?";  // Write to Cell A continuosly

//replace with sheet name not with spreadsheet file name taken from google
String append_payload_base =  "{\"command\": \"appendRow\",\"sheet_name\": \"emsLog\",\"values\": ";
String update_payload_base =  "{\"command\": \"update_row\",\"sheet_name\": \"dsm\", \"values\": ";
String append_payload = "";
String update_payload = "";

const int RSSI_MAX =-50;// define maximum straighten of signal in dBm
const int RSSI_MIN =-100;// define minimum strength of signal in dBm

const long interval = 300;
unsigned long previousMillis = 0;
const long interval_post = 1;
unsigned long previousMillis_post = 0;
const long interval_fetch = 1;
unsigned long previousMillis_fetch = 0;
uint8_t LEDpin1 = D4;//D4;
//uint8_t load1pin = D0;
uint8_t load1 = D1;
uint8_t load2 = D0;
uint8_t load3 = D2;
```

```
bool LightON = HIGH, LightOFF = LOW;
bool LEDstatus = LOW;
bool heartbeat = LOW;
String load1_status="1",load2_status="1", load3_status="1", inverter_status="1",
tanesco_charger_status="1", pv_charger_status="1",
inverter_fan="0",inverter_temp="0",postData,pir_status,door_status;

//String
ds18b20_temp,dht22_temp,heat_index,dht22_humidity,load1,load2,load3,ac_voltage,currentvalue,frequency,in
v_current ,pv_volt,pv_current,charged_percent,charging_status ,bat_volt ,person,dc_fan_status ,inv_fan_status
,inv_temp ,inv_status ,power_blackout ,reed_switch_indicator,cardStatus , postData;
//  String
ds18b20_temp,dht22_temp,heat_index,dht22_humidity,load1,load2,load3,ac_voltage,currentvalue,frequency,in
v_current ,pv_volt,pv_current,charged_percent,charging_status ,bat_volt ,person,dc_fan_status ,inv_fan_status
,inv_temp ,inv_status ,power_blackout ,reed_switch_indicator,cardStatus , postData;
int post = 1,reset_counter=0,reset_flag=0;
String dataUploaded,loadControl_flag;

WiFiClientSecure client;

//i2c one-wire is used to send/receive data from arduino mega with higher reliability than normal serial
communication
#include <Wire.h>
#define datasize 500 // ample datasize to accomodate all characters sent from arduino
#define i2c_address 6 // Slave in this case is arduino
#define baudrate 57600
String received_i2c_data;
String received_uart_data;


void setup() {
 Serial.begin(baudrate);
 //initialize wifi module for i2c communication i.e one wire communication
esp2mega.begin(baudrate); // serial to receive/send data from/to arduino
  Wire.begin(D1, D2); /* join i2c bus with SDA=D1 and SCL=D2 of NodeMCU */
  pinMode(LEDpin1, OUTPUT);
  pinMode(load3, OUTPUT);      // lights, epson printer, radio amplifier
  pinMode(load2, OUTPUT);      // cctv
  pinMode(load1, OUTPUT);      // tanesco loads: canon photocopier, microwave machine
   digitalWrite(LEDpin1, HIGH);
   digitalWrite(load1, HIGH);


   wifi_setup();
   client.setInsecure();
//uncomment if you want to get a msg everytime a reset occurs

  delay(1000);
}


void wifi_setup(){
   // Register multi WiFi networks
  wifiMulti.addAP(ssid1, password1);
  wifiMulti.addAP(ssid2, password2);
 // wifiMulti.addAP("ssid_from_AP_3", "your_password_for_AP_3");

 WiFi.mode(WIFI_STA);
//WiFi.begin(ssid1,password1);
 // while(WiFi.status()!= WL_CONNECTED){ //waiting for device to be connected to the network
 // Wait for the Wi-Fi to connect: scan for Wi-Fi networks, and connect to the strongest of the networks above
 while (wifiMulti.run() != WL_CONNECTED) {
  Serial.print(".");
```

```
 delay(500);
}
Serial.print("CONNECTED TO WIFI : ");  Serial.println(WiFi.SSID());
Serial.print("IP:");Serial.println(WiFi.localIP());

// print the received signal strength
  long rssi = WiFi.RSSI();
  Serial.print("Signal strength (RSSI):");
  Serial.print(rssi);
  Serial.print(" dBm ( ");
        Serial.print(dBmtoPercentage(rssi));//Signal strength in %
      Serial.println("% )");

  }
void loop(){
    ESP.wdtFeed();yield();
   load_controller();
  // Serial.println("void loop");
heart_beat();
 wifi_setup();
  refreshSerial();

if(post==1){
dataUploaded=uploadData();
delay(3000);
//post=2;
}
else if(post==2){
  Serial.println("data fetch routine...");
fetchData();
}
else if(post==3){
thingsboard();
  gsheetCom(dataUploaded);
  }

  }
void json_sender(String data_sent){
    char jsonData[datasize];
    data_sent.toCharArray(jsonData, datasize);
    //Serial.print("json is ");Serial.println(jsonData);
esp2mega.write(jsonData);
  }
 void fetchData(){
// begin by fetching data from server first before sending it to arduino mega
unsigned long currentMillis_fetch = millis();

  if (currentMillis_fetch - previousMillis_fetch >= interval_fetch) {

    // save the last time you blinked the LED
    previousMillis_fetch = currentMillis_fetch;
       ESP.wdtFeed();yield();
     wifi_setup();

 HTTPClient http;    //Declare object of class HTTPClient
 WiFiClientSecure client;
client.setInsecure(); //the magic line, use with caution
    Serial.print("Connect to: ");Serial.print(host);Serial.println(" and FETCH data");
client.connect(host, httpsPort);

       String serverPath = String("")+host+"/getUpdates.php";

      // Your Domain name with URL path or IP address with path
```

```
     http.begin(client, serverPath);

  int httpResponseCode = http.GET();   //Send the request
Serial.println(String(httpResponseCode)+" "+http.getString());
 ESP.wdtFeed();yield();
 if (httpResponseCode!=200){
    post = 3;
 return ;
 }

 DynamicJsonDocument doc(2048);
    deserializeJson(doc, http.getString());
    String ld1=doc["load1"];
    String ld2=doc["load2"];
    String ld3=doc["load3"];
    String inv=doc["inverter"];
    String tanesco=doc["tanescoCharger"];
    String pv=doc["pvCharger"];
    String inv_fan=doc["inverter_fan"];
    String inv_temp=doc["inverter_temp"];
    String load_flag=doc["loadControl_flag"];
    String pirStatus=doc["pir_status"];
    String doorStatus=doc["door_status"];

     load1_status=ld1;
    load2_status=ld2;
     load3_status=ld3;
     inverter_status=inv;
     tanesco_charger_status=tanesco;
     pv_charger_status=pv;
     inverter_fan=inv_fan;
     inverter_temp=inv_temp;
     loadControl_flag=load_flag;
     pir_status=pirStatus;
     door_status=doorStatus;

//String payload = load1_status + "," + load2_status + "," + load3_status+ "," + inverter_status+ "," +
tanesco_charger_status+ "," + pv_charger_status+ "," + inverter_fan+ "," + inverter_temp+ "," +
loadControl_flag+ "," + pir_status+ "," + door_status;

  Serial.print("server load1 status is : ");Serial.println(load1_status);
  Serial.print("server load2 status is : ");Serial.println(load2_status);
  Serial.print("server load3 status is : ");Serial.println(load3_status);
  Serial.print("server inverter status is : ");Serial.println(inverter_status);
  Serial.print("server tanesco_charger status is : ");Serial.println(tanesco_charger_status);
  Serial.print("server pv_charger status is : ");Serial.println(pv_charger_status);
  Serial.print("server inverter_fan status is : ");Serial.println(inverter_fan);
   Serial.print("server inverter_temp is : ");Serial.println(inverter_temp);
   Serial.print("loadControl_flag is : ");Serial.println(loadControl_flag);

ESP.wdtFeed();yield();

// ############## SEND DATA TO ARDUINO MEGA ###################################
    // now send data to arduino mega by serial communication
//String data_sent =
"{\"load1\":\"1\",\"load2\":\"1\",\"load3\":\"1\",\"inverter\":\"1\",\"tanescoCharger\":1,\"pvCharger\":\"1\",\"inver
ter_fan\":\"0\",\"inverter_temp\":\"36.44\",\"loadControl_flag\":\"tanesco_on\"}";
//json_sender(data_sent);
// send 10x to guarantee transmission or wait to receive confirmation 'ok' while keep sending
int messenger = 0;
while(messenger < 10){
  String data_sent = "{\"load1\":\""+String(load1_status)+"\",";
  //serializeJson(Data_sent, esp2mega);
```

```
  //send one data at a time
  json_sender(data_sent);
  data_sent = "\"load2\":\""+String(load2_status)+"\",";
  json_sender(data_sent);
  data_sent = "\"load3\":\""+String(load3_status)+"\",";
  json_sender(data_sent);
  data_sent = "\"inverter\":\""+String(inverter_status)+"\",";
  json_sender(data_sent);
  data_sent = "\"tanescoCharger\":\""+String(tanesco_charger_status)+"\",";
  json_sender(data_sent);
  data_sent = "\"pvCharger\":\""+String(pv_charger_status)+"\",";
  json_sender(data_sent);
  data_sent = "\"inverter_fan\":\""+String(inverter_fan)+"\",";
  json_sender(data_sent);
  data_sent = "\"inverter_temp\":\""+String(inverter_temp)+"\",";
  json_sender(data_sent);
  data_sent = "\"loadControl_flag\":\""+String(loadControl_flag)+"\"}\n";
  json_sender(data_sent);
  messenger++;
delay(1000);
  }
//return ;
// prepare data to send to arduino mega board
//StaticJsonDocument<1024> doc;
/*DynamicJsonDocument doc2(2048);
  doc2["load1_status"] = load1_status;
  doc2["load2_status"] = load2_status;
  doc2["load3_status"] = load3_status;
  doc2["inverter_status"] = inverter_status;
  doc2["tanesco_charger_status"] = tanesco_charger_status;
  doc2["pv_charger_status"] = pv_charger_status;
  doc2["inverter_fan"]= inverter_fan;
  doc2["inverter_temp"]= inverter_temp;
  doc2["loadControl_flag"]= loadControl_flag;
  //send atleast 10 times
// usual routine for sending data to arduino mega through serial communication TX & RX
  serializeJson(doc2, esp2mega);
  //deserializeJson(doc, Serial);//uncomment to see what is being sent to arduino mega board
  */
  http.end();  //Close connection
  }
  post = 3;
  }

 void load_controller(){
     ESP.wdtFeed();yield();
 if(loadControl_flag=="load1_on"){ digitalWrite(load1, LOW); load1_status="1";}
  else if(loadControl_flag=="load1_off"){digitalWrite(load1, HIGH);load1_status="0";}
  else if(loadControl_flag=="load2_on"){digitalWrite(load2, HIGH);load2_status="1";}
  else if(loadControl_flag=="load2_off"){digitalWrite(load2, LOW);load2_status="0";}
  else if(loadControl_flag=="load3_on"){digitalWrite(load3, LightON);load3_status="1";}
  else if(loadControl_flag=="load3_off"){digitalWrite(load3, LightOFF);load3_status="0";}
  //else if(loadControl_flag=="pv_on"){ pv_charger_flag=1;}
  //else if(loadControl_flag=="pv_off"){pv_charger_flag=0;}
  else{
    if(pv_charger_status=="1"){
 // pv_charger_flag=1;

  }
else if(pv_charger_status=="0"){
 // pv_charger_flag=0;

  }
```

```
  if( load1_status=="1" ){
  digitalWrite(load1, LOW); // turn ON load1, load1 uses inverted logic where off is HIGh and on is LOW
         ESP.wdtFeed();yield();
        Serial.println("turning load1 ON ");
        }
        else if( load1_status=="0"){
           digitalWrite(load1, HIGH); // turn OFF load1
            Serial.println("turning load1 OFF ");
          }
      if( load2_status=="1"){
  digitalWrite(load2, HIGH); // turn ON load1
        Serial.println("turning load2 ON ");
        }
        else if( load2_status=="0"){
           digitalWrite(load2, LOW); // turn OFF load1
            Serial.println("turning load2 OFF ");
          }
           if( load3_status=="1" || door_status=="1" ){
  digitalWrite(load3, LightON); // turn ON load1
        Serial.println("turning load3 ON ");
        }
        else if( load3_status=="0" || door_status=="0"){
           digitalWrite(load3, LightOFF); // turn OFF load1
            Serial.println("turning load3 OFF ");
          }
    }
         ESP.wdtFeed();yield();
    }

  //function to send sensor data
String uploadData() {
  //------------ RECEIVE DATA FROM ARDUINO AND UPLOAD TO CLOUD ------------------------
Serial.println("data upload routine inside...");
    ESP.wdtFeed();yield();

// receive data from arduino via serial communication
while (esp2mega.available() > 0) {
  //clear buffer first
     received_uart_data=""; //clear buffer
        // read the incoming byte:
           String str  = esp2mega.readString();
          // String str = esp2mega.readStringUntil('\n');
          Serial.print("*received data: ");Serial.println(str);
        //if received data is mangled i.e has '?', then discard
            // substring(index) looks for the substring from the index position to the end:
        if (received_uart_data.substring(0) == "?") {
          received_uart_data=""; //clear buffer
        }
           //concat subsequent characters
          received_uart_data.concat(str);
        //Serial.print("arduino inside: ");Serial.println(received_uart_data);
        //mega2node.write("be of good courage! never give up");
}
 //Serial.println();// go to line below
//Serial.print("arduino outside: ");Serial.println(received_uart_data);
int index2 = received_uart_data.indexOf("?",3);
if(index2>0){
   return "nothing";//bad data received, fetch another stream
  }
int index1 = received_uart_data.indexOf("{",3);
 Serial.print("start index: ");Serial.println(index1);
 if(index1>0){
received_uart_data = received_uart_data.substring(index1);
```

```
Serial.print("*cleaned data: ");Serial.println(received_uart_data);
}
else{
    String data_sent = "{\"loadControl_flag\":\"wifi_reset\"}";
  //serializeJson(Data_sent, esp2mega);
  //send one data at a time
  json_sender(data_sent);
 return "nothing";
 }
 delay(5000);


  /*   char jsonData[datasize];
    received_uart_data.toCharArray(jsonData, datasize);
    Serial.print("json is ");Serial.println(jsonData);*/

Serial.println("deserializing received cleaned data...");
 DynamicJsonDocument doc(2048);
 // DeserializationError error = deserializeJson(doc, esp2mega);
 //deserializeJson(doc, jsonData);
 deserializeJson(doc, received_uart_data);
 // JsonObject object = doc.as<JsonObject>();

 //deserializeJson(doc, esp2mega);

String  ds18b20_temp = doc["ds18b20_temp"];
String  dht22_temp = doc["dht22_temp"];
String  heat_index = doc["heat_index"];
String  dht22_humidity=doc["dht22_humidity"];
String  load1=doc["load1"];
String  load2=doc["load2"];
String  load3=doc["load3"];
String  ac_voltage=doc["ac_voltage"];
String  currentvalue=doc["currentvalue"];
String  frequency=doc["frequency"];
String  inv_current=doc["inv_current"];
String  pv_volt=doc["pv_volt"];
String  pv_current=doc["pv_current"];
String  charged_percent=doc["charged_percent"];
String  charging_status=doc["charging_status"];
String  bat_volt=doc["bat_volt"];
String  person=doc["person"];
String  dc_fan_status=doc["dc_fan_status"];
String  inv_fan_status=inverter_fan;
String  inv_temp=doc["inv_temp"];
String  inv_status=inverter_status;
String  power_blackout=doc["power_blackout"];
String  reed_switch_indicator=doc["reed_switch_indicator"];
//String tanescoCharger=tanesco_charger_status;
String  cardStatus=doc["cardStatus"];

pir_status=person;
door_status=reed_switch_indicator;

String postData = "ds18b20_temp=" +  ds18b20_temp + "&dht22_temp=" + dht22_temp+
"&dht22_heatIndex_temp=" + heat_index+  "&dht22_humidity=" +
dht22_humidity+"&load1="+load1+"&load2="+load2+"&load3="+load3+ "&ac_voltage=" + ac_voltage+
"&ac_current=" + currentvalue+  "&frequency=" + frequency+ "&inv_current="+inv_current+"&pv_voltage="
+ pv_volt+  "&pv_current=" + pv_current+  "&soc=" + charged_percent+  "&pv_bat_charging=" +
charging_status+  "&battery_voltage=" + bat_volt+  "&pir_status=" + person+  "&fan1_status=" +
dc_fan_status+  "&fan2_status=" + inv_fan_status+  "&inverter_temp=" + inv_temp+  "&inverter_status=" +
inv_status+  "&blackout=" + power_blackout+  "&door_status=" + reed_switch_indicator+
"&cardStatus="+cardStatus+"&tanesco_charger_status="+tanesco_charger_status;
```

```
//String postData = "{\"command\":\"" +  ds18b20_temp + "," + dht22_temp+ "," + heat_index+  "," +
dht22_humidity+","+load1+","+load2+","+load3+ "," + ac_voltage+ "," + currentvalue+  "," + frequency+
","+inv_current+"," + pv_volt+  "," + pv_current+  "," + charged_percent+  "," + charging_status+  "," +
bat_volt+  "," + person+  "," + dc_fan_status+  "," + inv_fan_status+  "," + inv_temp+  "," + inv_status+  "," +
power_blackout+  "," + reed_switch_indicator+  ","+cardStatus+","+tanesco_charger_status+"\"}";
//String postData = "{\"command\":\"" +  25 + "," + 26+ "," + 27+  "," + 50+","+1+","+1+","+1+ "," + 220+ ","
+ 0.5+  "," + 50+ ","+0.5+"," + 19+  "," + 0.5+  "," + 100+  "," + 1+  "," + 13+  "," + 1+  "," + 1+  "," + 0+  "," +
25+  "," + 1+  "," + 0+  "," + 1+  ","+0+","+1+"\"}";
String upload_data =   ds18b20_temp + "," + dht22_temp+ "," + heat_index+  "," +
dht22_humidity+","+load1+","+load2+","+load3+ "," + ac_voltage+ "," + currentvalue+  "," + frequency+
","+inv_current+"," + pv_volt+  "," + pv_current+  "," + charged_percent+  "," + bat_volt+  "," + person+   "," +
inv_temp+  "," + inv_status+  "," + power_blackout+  "," + reed_switch_indicator+ ","+tanesco_charger_status;
     Serial.print("postData: "); Serial.println(postData);
//if(ds18b20_temp.toInt())return "nothing";
if (ds18b20_temp == NULL)return "";
if (ds18b20_temp.isEmpty())  return "";

Serial.print("ds18b20_temp is "+ds18b20_temp+" temp "+String(ds18b20_temp.toFloat())+" isnan output:
");Serial.println(isnan(ds18b20_temp.toFloat()));
if(isnan(ds18b20_temp.toFloat())||ds18b20_temp.toFloat()<10)return "";

//post = 2;//################### remember to disable this command ####################
//return "nothing"; //################### remember to disable this command ####################

  unsigned long currentMillis_post = millis();
  if (currentMillis_post - previousMillis_post >= interval_post) {

    // save the last time you blinked the LED
    previousMillis_post = currentMillis_post;
     ESP.wdtFeed();yield();

 HTTPClient http;    //Declare object of class HTTPClient
 WiFiClientSecure client;

     if(reset_counter<1 && reset_flag==0){
      client.setFingerprint(fingerprint);
  client.setTimeout(15000); // 15 Seconds
  Serial.print("Connect to: ");Serial.print(host);Serial.println(" and post data");
client.connect(host, httpsPort);
     String serverPath = String("")+host+"/postData.php";
      http.begin(client, serverPath);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");    //Specify content-type header
      http.addHeader("Content-Length",String(postData.length()));
      //ESP.wdtFeed();
        int httpResponseCode = http.POST(postData);
       // Serial.print("HTTP code: ");
        Serial.println(String(httpResponseCode)+" "+http.getString());
      yield();
        if (httpResponseCode!=200){
          reset_counter++;
        }
    }

    if(reset_counter>=1 && reset_flag==0){
           client.setFingerprint(fingerprint);
  client.setTimeout(15000); // 15 Seconds
  Serial.print("Connect to: ");Serial.print(host);Serial.println(" and post data");
client.connect(host, httpsPort);
     String serverPath = String("")+host+"/getpostData.php?";
     // http.begin(client, serverPath);
  http.begin(client, serverPath+postData);
       int httpResponseCode = http.GET();   //Send the request
       // Serial.print("HTTP code: ");
```

```
      Serial.println(String(httpResponseCode)+" "+http.getString());
     yield();
      if (httpResponseCode!=200){
       //ESP.reset();
       reset_counter++;
      reset_flag=1;
      }
     }

     if(reset_counter>=1 && reset_flag==1){
               client.setInsecure(); //the magic line, use with caution
  Serial.print("Connect to: ");Serial.print(host);Serial.println(" and post data");
client.connect(host, httpsPort);
     String serverPath = String("")+host+"/getpostData.php?";
  http.begin(client, serverPath+postData);
      int httpResponseCode = http.GET();   //Send the request
      Serial.println(String(httpResponseCode)+" "+http.getString());
     yield();
      if (httpResponseCode!=200){
           reset_counter=0;
          reset_flag=0;
           ESP.reset();
        //return "";//if upload failed exit now}
        // instruct arduino mega to do a hard reset of nodemcu
           String data_sent = "{\"loadControl_flag\":\"wifi_reset\"}";
  //serializeJson(Data_sent, esp2mega);
  //send one data at a time
  json_sender(data_sent);
       }

    //ESP.restart();
    }


 client.flush();

 http.end();  //Close connection
}

    post = 2;


    Serial.print("upload data ....: ");Serial.println(upload_data);
     return upload_data;
}

void refreshSerial(){
    Serial.begin(baudrate);//it's important to put the serial comm here to avoid null errors in the data received
 //esp2mega.begin(baudrate);
  //Wire.begin(D5, D6); /* join i2c bus with SDA=D5 and SCL=D6 of NodeMCU */
  }
void gsheetCom(String input){
HTTPSRedirect* gsclient = nullptr;
    ESP.wdtFeed();yield();
    static int error_count = 0;
  static int connect_count = 0;
  const unsigned int MAX_CONNECT = 20;
  static bool flag = false;
  append_payload =append_payload_base + "\""+input+ + "\"}";

  Serial.print("data to be appended is:");Serial.println(append_payload);
  if(append_payload=="nothing"){
    post=1;
```

```
  return;
  }
   post=1;

  // ******* now post to spreadsheet ******************
    // Use HTTPSRedirect class to create a new TLS connection
  gsclient = new HTTPSRedirect(httpsPort);
  gsclient->setInsecure();
  gsclient->setPrintResponseBody(true);
  gsclient->setContentTypeHeader("application/json");
  Serial.print("Connecting to ");
  Serial.println(spreadsheethost);          //try to connect with "script.google.com"

ESP.wdtFeed();
  // Try to connect for a maximum of 5 times then exit
   flag = false;
  for (int i = 0; i < 5; i++) {
    int retval = gsclient->connect(spreadsheethost, httpsPort);
      ESP.wdtFeed();yield();
    if (retval == 1) {
      flag = true;
      break;
    }
    else
      return;//Serial.println("Connection failed. Retrying...");
  }

  if (!flag) {
    Serial.print("Could not connect to server: ");
    Serial.println(spreadsheethost);
    Serial.println("Exiting...");
    return;
  }
  Serial.println("\nWrite into cell 'A2'");
  Serial.println("------>");
  // fetch spreadsheet data
    ESP.wdtFeed();yield();
  gsclient->GET(url, spreadsheethost);
 ESP.wdtFeed();
  // delete HTTPSRedirect object
  delete gsclient;
  gsclient = nullptr;
 ESP.wdtFeed();
    // Use HTTPSRedirect class to create a new TLS connection
  gsclient = new HTTPSRedirect(httpsPort);
  gsclient->setInsecure();
  gsclient->setPrintResponseBody(true);
  gsclient->setContentTypeHeader("application/json");
  Serial.print("Connecting to ");
  Serial.println(spreadsheethost);          //try to connect with "script.google.com"
ESP.wdtFeed();
  if (!flag) {
     ESP.wdtFeed();yield();
    gsclient = new HTTPSRedirect(httpsPort);
    gsclient->setInsecure();
    flag = true;
    gsclient->setPrintResponseBody(true);
    gsclient->setContentTypeHeader("application/json");
  }
ESP.wdtFeed();
  if (gsclient != nullptr) {
     ESP.wdtFeed();yield();
    if (!gsclient->connected()) {
```

```
      ESP.wdtFeed();yield();
    gsclient->connect(spreadsheethost, httpsPort);
    gsclient->POST(url2, spreadsheethost, append_payload, false);
    Serial.print(".... ");  Serial.println(append_payload);
   }
  }
  else {
   DPRINTLN("Error creating gsclient object!");
   error_count = 5;
  }


ESP.wdtFeed();
   Serial.print("data to be appended is:");Serial.println(append_payload);
  if (gsclient) {
   int httpResponseCode =gsclient->POST(url2, spreadsheethost, append_payload);
   ESP.wdtFeed();yield();
   Serial.print("*** httpResponseCode:");Serial.println(httpResponseCode);
    Serial.println(gsclient->getStatusCode());
    Serial.println(gsclient->getReasonPhrase());
    Serial.println(gsclient->getResponseBody());
    ESP.wdtFeed();yield();
    //no need to deserialize
    /*
       DynamicJsonDocument doc(2048);
    deserializeJson(doc, gsclient->getResponseBody());
    String load1=doc["load1"];
    String load2=doc["load2"];
    String load3=doc["load3"];
    String inverter=doc["inverter"];
    String tanescoCharger=doc["tanescoCharger"];
    String pvCharger=doc["pvCharger"];*/
  delete gsclient;

  /*
  // there is no need to send control updates since namecheap server is already being updated
       httpResponseCode =gsclient->POST(url2, host, update_payload);
   Serial.print("*** httpResponseCode:");Serial.println(httpResponseCode);
    Serial.println(gsclient->getStatusCode());
    Serial.println(gsclient->getReasonPhrase());
    Serial.println(gsclient->getResponseBody());
   */

 }
 ESP.reset();

  }

void thingsboard(){
   ESP.wdtFeed();yield();

  WiFiClient wifiClient;
ThingsBoard tb(wifiClient);

// reconnect to thingsboard with our token
   Serial.print("Connecting to ThingsBoard node ...");
    tb.connect(thingsboardServer, TOKEN);

 ESP.wdtFeed();yield();
 // Prepare a JSON payload string
DynamicJsonDocument doc(2048);
  DeserializationError error = deserializeJson(doc, esp2mega);
  // Test if parsing succeeds.
```

```
 if (error) {
   return ;
 }
// JsonObject object = doc.as<JsonObject>();

 //deserializeJson(doc, esp2mega);
    Serial.println("JSON received and parsed data from arduino mega");
String  ds18b20_temp = doc["ds18b20_temp"];
String  dht22_temp = doc["dht22_temp"];
String  heat_index = doc["heat_index"];
String  dht22_humidity=doc["dht22_humidity"];
String  load1=load1_status;
String  load2=load2_status;
String  load3=load3_status;
String  ac_voltage=doc["ac_voltage"];
String  currentvalue=doc["currentvalue"];
String  frequency=doc["frequency"];
String  inv_current=doc["inv_current"];
String  pv_volt=doc["pv_volt"];
String  pv_current=doc["pv_current"];
String  charged_percent=doc["charged_percent"];
String  charging_status=doc["charging_status"];
String  bat_volt=doc["bat_volt"];
String  person=doc["person"];
String  dc_fan_status=doc["dc_fan_status"];
String  inv_fan_status=inverter_fan;
String  inv_temp=doc["inv_temp"];
String  inv_status=inverter_status;
String  power_blackout=doc["power_blackout"];
String  reed_switch_indicator=doc["reed_switch_indicator"];
//String tanescoCharger=tanesco_charger_status;
String  cardStatus=doc["cardStatus"];

   ESP.wdtFeed();yield();
   if(isnan(ds18b20_temp.toFloat()))return;
Serial.println("data to thingsboard:"+ds18b20_temp);
    tb.sendTelemetryFloat("ds18b20_temp", ds18b20_temp.toFloat());
  tb.sendTelemetryFloat("dht22_temp", dht22_temp.toFloat());
  tb.sendTelemetryFloat("heat_index", heat_index.toFloat());
  tb.sendTelemetryFloat("dht22_humidity", dht22_humidity.toFloat());
  tb.sendTelemetryFloat("load1", load1.toFloat());
  tb.sendTelemetryFloat("load2", load2.toFloat());
  tb.sendTelemetryFloat("load3", load2.toFloat());
  if(power_blackout=="0"){
    float power =ac_voltage.toFloat()*currentvalue.toFloat();
    tb.sendTelemetryFloat("power", power);
    }
  else if(power_blackout=="1"){
       float power =220*inv_current.toFloat();
    tb.sendTelemetryFloat("power", power);
    }
  tb.sendTelemetryFloat("ac_voltage", ac_voltage.toFloat());
      tb.sendTelemetryFloat("currentvalue", currentvalue.toFloat());
  tb.sendTelemetryFloat("frequency", frequency.toFloat());
  tb.sendTelemetryFloat("inv_current", inv_current.toFloat());
  tb.sendTelemetryFloat("pv_volt", pv_volt.toFloat());
  tb.sendTelemetryFloat("pv_current", pv_current.toFloat());
  tb.sendTelemetryFloat("charged_percent", charged_percent.toFloat());
  tb.sendTelemetryFloat("bat_volt", bat_volt.toFloat());
  tb.sendTelemetryFloat("person", person.toFloat());
    tb.sendTelemetryFloat("inv_temp", inv_temp.toFloat());
  tb.sendTelemetryFloat("inv_status", inv_status.toFloat());
  tb.sendTelemetryFloat("power_blackout", power_blackout.toFloat());
```

```
  tb.sendTelemetryFloat("door_status", reed_switch_indicator.toFloat());
  tb.sendTelemetryFloat("tanesco_charger_status", tanesco_charger_status.toFloat());

}

  void heart_beat()  {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
     heartbeat = ! heartbeat;
    previousMillis = currentMillis;
    digitalWrite(LEDpin1, heartbeat);
  }
    }

  int dBmtoPercentage(int dBm){
  int quality;
    if(dBm <= RSSI_MIN)
    {
       quality = 0;
    }
    else if(dBm >= RSSI_MAX)
    {
       quality = 100;
    }
    else
    {
       quality = 2 * (dBm + 100);
   }

     return quality;
}//dBmtoPercentage
```

Appendix 5: Telegram Chatbot Python Source Code

```
import datetime # to get current time for error logging
import time # to time routines/tasks durations
import pytz # to convert from server to Tanzania timezone

#import ems functions
import ems_functions as ems

# telegram bot libraries
import urllib3
import json
from telegram import *
from telegram.ext import *
bot_chat_id="160085"
group_chat_id="-58036"
bot_token="170232:AAGxLqurfNkiEYAaPPlzrWsKY"
bot = Bot(bot_token)
intruder_error="&#9940; you are not authorized &#8252; contact admin..."

# main menu keyboard
kbd_mainmenu = [['grid status', 'Forecasts','Charts'], ['consumption', 'blackout info','sub menu']]
#kbd_submenu = [['consumption', 'credit balance','blackout info'], ['main menu']]
kbd_submenu = [['tanesco on', 'tanesco off','credit balance'], ['inverter on','inverter off','grid reset'],['main
menu']]
#kbd = ReplyKeyboardMarkup(kbd_layout,resize_keyboard=True,one_time_keyboard=True)
kbd1 = ReplyKeyboardMarkup(kbd_mainmenu,resize_keyboard=True)
kbd2 = ReplyKeyboardMarkup(kbd_submenu,resize_keyboard=True)

#list for tracking message IDs
msg_id=[]

#dayload is the average daily kwh obtained from luku control module
dayload=1.14
dateformat = "%Y-%m-%d"

# approximation of MB used on each datalog record row
rownet = 0.0119601329
# approx 27MB are spent each day using halotel internet
daynet = 27
luku_tariff=356.13

def menu_msg():
    # Send the message with menu
    bot.send_message(group_chat_id,"choose an <b>option</b>
below:",parse_mode='html',reply_markup=kbd1)


def start_command(update, context):
  welcome_msg="\t\t\tWelcome! \n Type any of the following commands to get a reply:\n\t/start\n\t/help \n\tgrid
status \n\t tanesco balance \n\t total consumption \n\t internet balance\n\ttanesco on \n\t inverter on "
  chat_id=update.message.chat_id
  #make sure the sender belongs to our chat group
  if str(chat_id)==group_chat_id:
      #update.message.reply_text(welcome_msg)
      bot.send_message(group_chat_id,welcome_msg,parse_mode='html',reply_markup=kbd1)

  else:
      update.message.reply_text(intruder_error)
def standard_responses(user_message):
    con = pymysql.connect(host="127.0.0.1",user="tech ",password="TwO&",db="tech
",charset="utf8mb4",cursorclass=pymysql.cursors.DictCursor,autocommit=True)
```

```
    cursor = con.cursor()
    conn = create_engine('mysql+pymysql://techd').connect()
    if user_message in ("status", "grid status","voltage","volts","battery", "bat","soc","charge","current","inverter
current","ac current"):
        #select today's date and most recent time
        sql = 'SELECT * FROM logs ORDER BY `id` DESC limit 1';
        df = pd.read_sql(sql, conn)
        period=pd.to_datetime(df['Date'])+df['Time']
        df.insert(0,"datetime",period, True)
        sql = 'SELECT * FROM loads where `id` =1';
        df2 = pd.read_sql(sql, conn)
        muda = str(df.datetime[0].strftime("%Y-%m-%d %H:%M"))
        msg=""
        msg+="\n\tData last fetched at: "+muda
        if df['blackout'].astype('float64').values[0]!=1 and df2['tanesco_charger'].astype('float64').values[0]==1:
            power = df['ac_voltage'].astype('float64').values[0]*df['ac_current'].astype('float64').values[0]
            msg+="\n\tac Power: <b>"+str(power.round(1))+"</b> W"
            msg+="\n\tac voltage: <b>"+str(df['ac_voltage'].astype('float64').round(1).values[0])+"</b> V"
            msg+="\n\tac current: <b>"+str(df['ac_current'].astype('float64').round(2).values[0])+"</b> A"
        else:
            power = 220*df['inv_current'].astype('float64').values[0]
            msg+="\n\tac Power: <b>"+str(power.round(1))+"</b> W"
            msg+="\n\tac voltage: <b>"+str(220)+"</b> V"
            msg+="\n\tac current: <b>"+str(df['inv_current'].astype('float64').round(2).values[0])+"</b> A"

        msg+="\n\tac frequency: <b>"+str(df['frequency'].astype('float64').round(1).values[0])+"</b> Hz"
        msg+="\n\tPV volts: <b>"+str(df['pv_voltage'].astype('float64').round(1).values[0])+"</b> V"
        msg+="\n\tbat volts: <b>"+str(df['battery_voltage'].astype('float64').round(1).values[0])+"</b> V"
        msg+="\n\tbat charge: <b>"+str(df['soc'].astype('string').values[0])+"%</b> "
        #update.message.reply_text(welcome_msg)
        bot.send_message(group_chat_id, msg,parse_mode='html',reply_markup=kbd1)
        return
    if user_message in ("plots", "charts","graphs", "plot", "chart","graph"):

        bot.send_photo(group_chat_id, photo=open('img/svr_forecast_nextday.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/month_blackoutday_heatmap.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/dataset_hist_kde.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/megaplot_boxplot.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/hourly_megaplot.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/weekly_megaplot.png', 'rb'))

        bot.send_photo(group_chat_id, photo=open('img/correlation_heatmap.png', 'rb'))

        return
    if user_message in ("blackout", "power off", "power loss","blackout info"):
        bot.send_photo(group_chat_id, photo=open('img/weekday_blackoutday_heatmap.png', 'rb'))
        bot.send_photo(group_chat_id, photo=open('img/wday_blackoutday_heatmap.png', 'rb'))
        bot.send_photo(group_chat_id, photo=open('img/month_blackoutday_heatmap.png', 'rb'))
        bot.send_photo(group_chat_id, photo=open('img/dataset_blackoutday_heatmap.png', 'rb'))
        sqlSelect='SELECT `blackoutForecast` FROM loads where `id` =1'
        # Execute the SQL SELECT query
        cursor.execute(sqlSelect)
        # Fetch pydate
        blackoutForecast = cursor.fetchall()
        blackoutForecast = float(blackoutForecast[0]['blackoutForecast'])
        if blackoutForecast<60:
            msg="blackout occurance is <b>LOW</b> with <b>"+str(blackoutForecast)+"%</b> probability!  "
            bot.send_message(group_chat_id,text = msg,parse_mode='html',reply_markup=kbd1)
```

```
        elif blackoutForecast>60:
            msg="blackout occurance is <b>HIGH</b> with <b>"+str(blackoutForecast)+"%</b> probability!  "
            bot.send_message(group_chat_id,text = msg,parse_mode='html',reply_markup=kbd1)
        return
    if user_message in ("tanesco balance", "balance","internet balance", "internet",
"data","expire","expiry","credit balance","netbal"):
        luku_status('lukubal')
        return
    if user_message in ("total consumption", "consumption", "load","luku"):
        luku_status('lukubal')
        return
    if user_message in ("reset", "restart","grid reset"):
        #update the loadControl_flag
        updateStatement = "UPDATE loads set loadControl_flag = 'reset' where id=1"
        cursor.execute(updateStatement)
        bot.send_message(group_chat_id,text = "Grid will be restarted
shortly!",parse_mode='html',reply_markup=kbd2)
        return
    if user_message in ("tanesco on", "charger on"):
        #update the loadControl_flag
        updateStatement = "UPDATE loads SET tanesco_charger = 1 WHERE id=1"
        cursor.execute(updateStatement)
        #update the loadControl_flag
        updateStatement = "UPDATE loads set loadControl_flag = 'tanesco_on' where id=1"
        cursor.execute(updateStatement)
        bot.send_message(group_chat_id,text = "Tanesco power has been set
ON!",parse_mode='html',reply_markup=kbd2)
        return
    if user_message in ("tanesco off", "charger off"):
        #update the loadControl_flag
        updateStatement = "UPDATE loads SET tanesco_charger = 0 WHERE id=1"
        cursor.execute(updateStatement)
        #update the loadControl_flag
        updateStatement = "UPDATE loads set loadControl_flag = 'tanesco_off' where id=1"
        cursor.execute(updateStatement)
        bot.send_message(group_chat_id,text = "Tanesco power has been set
OFF!",parse_mode='html',reply_markup=kbd2)
        return
    if user_message in ("forecast","forecasts", "predict","prediction","predictions"):
        url = 'img/svr_forecast_nextday.png'
        bot.send_photo(group_chat_id, photo=open(url, 'rb'))
        #sqlSelect   = "select `blackoutForecast` from loads"
        sqlSelect='SELECT `blackoutForecast` FROM loads where `id` =1'
        # Execute the SQL SELECT query
        cursor.execute(sqlSelect)
        # Fetch pydate
        blackoutForecast = cursor.fetchall()
        blackoutForecast = float(blackoutForecast[0]['blackoutForecast'])
        if blackoutForecast<60:
            msg="blackout occurance is <b>LOW</b> with <b>"+str(blackoutForecast)+"%</b> probability!  "
            bot.send_message(group_chat_id,text = msg,parse_mode='html',reply_markup=kbd1)
        elif blackoutForecast>60:
            msg="blackout occurance is <b>HIGH</b> with <b>"+str(blackoutForecast)+"%</b> probability!  "
            bot.send_message(group_chat_id,text = msg,parse_mode='html',reply_markup=kbd1)
        return
    if user_message in ("inverter on", "inv on"):
        #update the loadControl_flag
        updateStatement = "UPDATE loads SET inverter = 1 WHERE id=1"
        cursor.execute(updateStatement)
        #update the loadControl_flag
        updateStatement = "UPDATE loads set loadControl_flag = 'inverter_on' where id=1"
        cursor.execute(updateStatement)
```

```
        bot.send_message(group_chat_id,text = "inverter has been set
ON!",parse_mode='html',reply_markup=kbd2)
        return
    if user_message in ("inverter off", "inv off","inverter_off"):
        sqlSelect   = "select `blackout` from logs ORDER BY `id` DESC limit 1'"
        # Execute the SQL SELECT query
        cursor.execute(sqlSelect)
        # Fetch pydate
        blackout = cursor.fetchall()
        if blackout==0:
            #update the loadControl_flag
            updateStatement = "UPDATE loads SET inverter = 0 WHERE id=1"
            cursor.execute(updateStatement)
            #update the loadControl_flag
            updateStatement = "UPDATE loads set loadControl_flag = 'inverter_off' where id=1"
            cursor.execute(updateStatement)
            bot.send_message(group_chat_id,text = "inverter has been set
OFF!",parse_mode='html',reply_markup=kbd2)
        else:
            bot.send_message(group_chat_id,text = "inverter can't be turned OFF!, contact
admin",parse_mode='html',reply_markup=kbd2)
        return
    msg=user_message.split()
    if len(user_message.split()) >= 2  and msg[0] in ("lukuinit", "luku_init","internetinit",
"internet_init","netinit","net_init"):
        #routine to load luku or check luku bal

        luku_status(user_message)
        return
    elif msg[0] in ("lukubal", "luku_bal"):
        luku_status(user_message)
        return
    else:
        #print(user_message)
        #print("invalid data format entered")
        bot.send_message(group_chat_id,text = "\t I don't understand you! \n \tType any of the following valid
commands to get a reply:\n\t/start\n\t/help  ",parse_mode='html',reply_markup=kbd1)



def handle_message(update, context):
    # make user input case insensitive
    user_message = str(update.message.text).lower()
    #get incoming message chat_id and compare with our group chat id
    chat_id=update.message.chat_id
    if str(chat_id)==group_chat_id:
            if user_message=="sub menu":
                kbd = ReplyKeyboardMarkup(kbd_submenu,resize_keyboard=True)
                bot.send_message(group_chat_id,"choose an option below:",parse_mode='html',reply_markup=kbd2)
            elif user_message=="main menu":
                kbd = ReplyKeyboardMarkup(kbd_mainmenu,resize_keyboard=True)
                bot.send_message(group_chat_id,"choose an option below:",reply_markup=kbd1)
            else:
                #bot.send_message(group_chat_id,"congratulations you've selected "+text,)
                standard_responses(user_message)

    else:
        update.message.reply_text(intruder_error)




def button(update: Update, context: CallbackContext) -> None:
```

```
    query = update.callback_query
    query.answer()

    # This will define which button the user tapped on (from what you assigned to "callback_data". As I assigned
them "1" and "2"):
    user_message = query.data
    user_message = str(user_message).lower()
    #check to see if button user pressed corresponds to our standard replies
    standard_responses(user_message)

def main():

  check_alarms()

  updater = Updater(bot_token, use_context=True)
  dp = updater.dispatcher

  dp.add_handler(CommandHandler("start",start_command))
  dp.add_handler(CommandHandler("help",help_command))
  #dp.add_handler(MessageHandler(Filters.text,handle_message))
  updater.dispatcher.add_handler(MessageHandler(Filters.text & ~Filters.command, handle_message))
  updater.dispatcher.add_handler(CallbackQueryHandler(button))
  dp.add_error_handler(error)
  updater.start_polling()
  updater.idle()


#---------------------- beginning of program -------------------------
#start by greeting user and displaying menu button options
print("starting...")
check_alarms()

'''
keyboard = [
    [
        InlineKeyboardButton("grid status", callback_data='status'),
        InlineKeyboardButton("Forecast", callback_data='forecast'),
        InlineKeyboardButton("Charts", callback_data='plots'),
    ],
    [
        InlineKeyboardButton("tanesco on", callback_data='tanesco on'),
        InlineKeyboardButton("tanesco off", callback_data='tanesco off'),
        InlineKeyboardButton("help ", callback_data='/help'),
        ]
    ]

reply_markup = InlineKeyboardMarkup(keyboard)
#update.message.reply_text("Replying to text", reply_markup=reply_markup)
bot.send_message(group_chat_id,"Greetings! choose an option below:",reply_markup=reply_markup)
'''

# access database and and get parameter status
```

RESEARCH OUTPUTS

(i) Publications

Mbuya, B. H., Dimovski, A., Merlo, M., & Kivevele, T. (2022). Very Short-Term Blackout Prediction for Grid-Tied PV Systems Operating in Low Reliability Weak Electric Grids of Developing Countries. *Complexity*, 2022, 1-13.

Mbuya, B., Moncecchi, M., Merlo, M., & Kivevele, T. (2019). Short-term load forecasting in a hybrid microgrid: A case study in Tanzania. *Journal of Electric Systems*, 15(4), 593-606

(ii) Poster Presentation