Application of Machine Learning Algorithm to Forecast Load and Development of a Battery Control Algorithm to Optimize PV System Performance in Phoenix, Arizona

by

Aashiek Hariharan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved August 2018
Graduate Supervisory Committee:

George G. Karady, Chair
David R. Allee
Jiangchao Qin
G. Thomas Heydt

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

The students of Arizona State University, under the mentorship of Dr George Karady, have been collaborating with Salt River Project (SRP), a major power utility in the state of Arizona, trying to study and optimize a battery-supported grid-tied rooftop Photovoltaic (PV) system, sold by a commercial vendor. SRP believes this system has the potential to satisfy the needs of its customers, who opt for utilizing solar power to partially satisfy their power needs.

An important part of this elaborate project is the development of a new load forecasting algorithm and a better control strategy for the optimized utilization of the storage system. The built-in algorithm of this commercial unit uses simple forecasting and battery control strategies. With the recent improvement in Machine Learning (ML) techniques, development of a more sophisticated model of the problem in hand was possible. This research is aimed at achieving the goal by utilizing the appropriate ML techniques to better model the problem, which will essentially result in a better solution. In this research, a set of six unique features are used to model the load forecasting problem and different ML algorithms are simulated on the developed model. A similar approach is taken to solve the PV prediction problem. Finally, a very effective battery control strategy is built (utilizing the results of the load and PV forecasting), with the aim of ensuring a reduction in the amount of energy consumed from the grid during the “on-peak” hours. Apart from the reduction in the energy consumption, this battery control algorithm decelerates the “cycling aging” or the aging of the battery owing to the charge/dis-charges cycles endured by selectively charging/dis-charging the battery based on need.

The results of this proposed strategy are verified using a hardware implementation (the PV system was coupled with a custom-built load bank and this setup was used to simulate a house). The results pertaining to the performances of the built-in algorithm and the ML algorithm are compared and the economic analysis is performed. The findings of this research are in the process of being published in a reputed journal.

To my parents,

Mrs. Shanthi Hariharan & Mr. S Hariharan

and in memory of,

Dr. George G. Karady

ACKNOWLEDGMENTS

My heartfelt and sincere thanks to my advisor, Dr. Karady, who made my dream of working with him in 2014, a reality in 2017. I cannot thank him enough for granting me this opportunity to work with him. It helped me rebuild myself and my confidence after a very tough period personally. I would not have got this closure at the end of graduate studies if I had not gotten the opportunity to pursue this dream of mine.

Sadly, as the date of writing this thesis, Dr. Karady passed away after a truly illustrious career of over 60 years. Dr. G. T. Heydt of the School of Electrical, Computer and Energy Engineering at the Arizona State University stood in place of Dr. Karady during the defense of this work. I shall forever be grateful to Dr. Heydt for this kind and generous gesture of his.

Dr. Allee's classes were the inspiration for me to choose this specific topic for my master's thesis. He was very helpful and was always there when I had a query right through the semester. Apart from Machine Learning, I learnt a lot on how to be a better and dignified person from Dr. Allee. A special mention to Dr. Allee's PhD student Mr. Zachary Drummond for his insightful thoughts and opinions on this project.

Dr. Qin was my first teacher at the Arizona State University. A very kind and calm person, I feel privileged to have him as my teacher. I would like to thank him for the time he has invested in me and for his invaluable suggestions and insights.

From SRP's side, Mr. Joel Dickinson has been a pillar of support, without whom, the completion of this thesis would have been impossible. He has been supporting me through thick and thin for the last 6 months. I am immensely grateful to him for all the phone calls attended, emails replied to and meetings he granted me. I am also thankful to SRP, for the financial/intellectual support they had extended to me throughout this research period.

I must thank my predecessor, Pavan Etha, for being with me right from the start through the end of this project. He has a very big heart and this entire project was possible only because of his continuous input of ideas and advices.

My colleagues, Mohammed Mubashir Shaikh and Chinmay Vad were with me in my lab and the terrace, helping me out with everything they could offer in terms of physical and intellectual help. A thank you is the least I could offer them.

Lekha M. is another such soul who, through her expertise, helped me put together the automated control circuit used in this project, apart from standing long hours on the terrace, helping me with the tedious hardware testing process.

Being so far away from home for so long would not have been easy had it not been for my brother, Harish Govindarajan, and sister-in-law, Aarthi Raveendran. They provided me with a lot of emotional support and were there every time I felt low or fatigued. They were the main reason behind me wanting to pursue a thesis as a part of my master's education.

Finally, I would like to thank god and my parents for the continued support and blessings showered upon me.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

LIST OF SYMBOLS

AC	Alternating Current
ASU	Arizona State University
DC	Direct Current
DNI	Direct Normal Irradiance
DNN	Deep Neural Networks
EA	Economic Analysis
ERC	Engineering Research Center
KNN	K-Nearest Neighbors
ML	Machine Learning
MPPT	Maximum Power Point Tracker
NREL	National Renewable Energy Laboratory
PV	Photovoltaic
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
SD	Site Demand
SoC	State of Charge
SRP	Salt River Project
SVM	Support Vector Machines

Chapter 1

INTRODUCTION

1.1 Statement of Problem and Motivation

The state of Arizona has abundant solar potential. In terms of potential, it is next to Nevada and in terms of installed capacity, it is third in the list behind California and North Carolina [1]. The state also has a target of reaching 15% renewable integration by the year 2025 and 30% of this required renewable energy should come from non-utility distributed generation. 50% of this requirement must be from, residential sites [1]. The Arizona State University has a very elaborate solar program. The program has a 50 MW generating capacity from on-campus and off-campus sites [2]. To support this research, a residential PV roof-top system from a well-established vendor, was installed on the roof of the Engineering Research Center (ERC) at the Tempe campus.

The built-in algorithm of the residential rooftop PV system has a simple load forecasting technique, in which a specific daily load is predicted, based on the load pattern of the corresponding day of the previous month. Based on this prediction, the charging/discharging pattern of the battery is determined. Though this is a simple approach, the method has a lot of disadvantages, especially, when there are seasonal changes. In such a case, the prediction becomes inaccurate and as a result, the battery maybe charged/discharged unnecessarily, shortening its life (due to 'cycling aging') [8] and reducing the ability of the storage system to reduce demand during on-peak hours. This in turn, will result in a higher Demand Charge and higher electricity bill for the customer. The cascaded effect leads to a longer investment return period for the customer too.

As mentioned earlier, the state of Arizona has a target of achieving 15% renewable penetration by the year 2025. Hence, the utilities in the state are trying to embrace customer choice and are supporting residential solar production, by coming forward with net metering and residential demand charges for customers with on-site generation, who do not purchase all of their energy requirements from the utility. There is also, the Customer Generation Price Plan, i.e. the E-27 plan for such customers from one such utility (see appendix for the E-27 plan details). The rooftop PV system is one of their more elaborate ways, of ensuring that, the targeted renewable penetration is achieved by 2025. This PV system will yield better performance, when upgraded with a more robust and accurate load forecasting algorithm. With the advancement in Machine Learning (ML) over the past few years, we are now able to develop high performance algorithms, for various problems with ease. This is one such scenario, where the room for improvement is vast.

As mentioned earlier, the built-in algorithm controlling this residential PV system, which predicts a specific daily load based on the load pattern of the corresponding day in the previous month, has a serious flaw. Generally, load pattern of a particular house is closely related to the temperature. More elaborately, the temperature of a day determines the nature of devices used in a house. A hot day will result in the use of the air conditioner for prolonged periods. The compressor of the refrigerator will run for longer periods, compared to the winter months, before going into power saver mode as there is more strain on the device to maintain the inside temperature. Similarly, in the winter months, the room heater and the water heaters maybe used more. Another hidden pattern here is the time at which these “seasonal devices” will be used.

The air conditioners will be required mostly during the summer afternoons and the air heaters might most likely be switched on during the winter nights. All these factors influence the power requirement of the house during a day. On top of this, in a state like Arizona, for example, the temperature (i.e. the load pattern) of June is very different from May. Similarly, August and September are very different from each other.

A combination of these factors leads to huge deviation in the predicted and the actual values for predicted load. Since the battery charging and dispatch depends on this prediction, it results in the inefficient operation of the PV system, thus, defeating its purpose of reducing the overall electricity bill of the customer. Therefore, there is sufficient evidence necessitating a more sophisticated way of predicting the load pattern — An approach that takes into account, the various temperature and seasonal fluctuations unique to the state and an algorithm that must be able to learn and adapt rather than a simple "one-size-fits-all" approach.

1.2 Scope of this Research

Students at the Arizona State University have been collaborating with SRP and trying to solve this problem since 2016 [3]. The initial paper on this topic, tried to solve the problem with a better prediction algorithm compared to the built-in method. Weighted K-means clustering technique was utilized to predict the load, by essentially performing a weighted average of the load values, pertaining to the previous year of the same residence. Regarding the Demand Charge reduction, the battery power was dispatched equally between the 8 summer on-peak hours. The results were better than that of the built-in algorithm but, there was a huge scope for improvement. This research work thus, takes a more sophisticated approach to still better the results.

The load forecasting is performed using Supervised Machine Learning algorithms, which result in more accurate predictions over the weighted K-means clustering and a unique battery control algorithm involving the availability of solar power has been developed, with the sole aim of reducing the number of charging cycles the battery undergoes.

The ML load forecasting algorithm utilizes six features to model the problem appropriately. The features are —

- Temperature
- Month of the year
- Day of the year
- Time of the day
- Day of the week
- Holidays/Working days.

The load values have been rounded to the nearest 200 W values. This process effectively converts the forecasting problem into a Supervised Learning — Classification problem. Seven different algorithms have been tried and the results from the Random Forest algorithm have been chosen, since it gives the best prediction. Similar approach has been adapted to predict the available solar (PV) energy, on the day of interest.

The forecast load is subtracted from the available PV power, to determine the power needed from the battery at that specific point in time. The procedure is repeated for all the on-peak hours and the maximum Site Demand (SD) for every half an hour interval during the on-peak hours is calculated, to be used in the Economic Analysis(EA). The Economic Analysis is based on the assumption that the simulated day reflects the load pattern of the house, for a specific billing period. The Demand Charge that would be paid by the customer at the end of the billing period in such a case, is calculated along with similar calculations based on the results of the built-in algorithm. A comparison of both is presented. The result — The ML based algorithm outperforms the built-in algorithm (see the Results section of each chapter)

Summary of Contributions

- The algorithm developed earlier [3] has been improved further by re-classifying the load forecasting problem, as a Supervised Learning — Classification problem and applying the appropriate Machine Learning algorithms.
- A more realistic model of the data set, by including features like Temperature, Month of the year, Day of the year, Time of the day, Day of the week and Holidays/Working days, has been constructed.
- An algorithm to predict the PV power generation of the system in use on a specific day in Arizona, has been developed.
- An algorithm to decide the specific power contribution of the battery, towards the load at any point of time on a given day, has been developed.
- A significant reduction in the Demand Charge and optimization of battery usage, thus resulting in a reduced investment return period for the customer was proven through Economic Analysis.

1.3 Description of Procedure

The main components of the system in discussion are the solar panels, a battery to store power, a power inverter and two custom built load banks. The system is described in detail in Chapter 2 of this thesis.

The power stored in the battery is solely reserved to supply the load during the on-peak hours, so that the Demand Charge and the net energy used from the grid during the on-peak hours are minimum resulting in the reduction in the cost customer must pay the utility at the end of the billing cycle. Thus, the development of an algorithm, to appropriately distribute the stored battery energy during the on-peak hours is the agenda. The algorithm developed supplies the load initially from the solar power, with the battery making up any deficit supply and the grid taking up any further deficit. In case of excess PV availability, the excess power must be exported to the grid. This would enable the customer to earn money (from the utility) in the form of credits on their electricity bill. For such an algorithm to work efficiently, the prediction of the amount of PV power available and the load forecast are prerequisites. Additionally, the prolongation of the battery life, by charging and dispatching the battery only when necessary, is also ensured by the implementation of such an algorithm.

A day is chosen from the one year load data set, given by a large electric utility company, based on the temperature on the day of simulation. The chosen load is simulated on two similar days — on the first day, the data pertaining to the built-in algorithm is extracted while the second day is for the new Machine Learning (ML) algorithm implementation. The 11 kW load bank simulates the chosen load while the developed ML algorithm will be used to predict this exact load by providing the rest of the load data as training data. The predicted values are fed into the battery control algorithm and the battery power required for every half-hour during the on-peak hours is determined. This is fed into the Sunverge system (through the online portal) in the form of pre-defined rules governing the battery charge/discharge. The results from both the algorithms are compared and an Economic Analysis is performed. The end results are tabulated.

1.4 Assumptions

The following assumptions have been made for this research. Specific explanations or justifications for the assumptions are provided at the respective chapters.

- For the test purpose, four load sets were chosen from the yearlong load data set provided by the sponsoring utility, to be simulated over 8 days. The ML and the built-in algorithms were tested on all the load sets separately (2 days/simulations for each load set).
- Since the utility considers load averages over half-hour period during the on-peak hours for its Demand Charge calculation, we cluster the given loads, predictions and results into half-hour clusters.
- Both the algorithms are tested on each load data set according to the procedure described in detail earlier.
- The average site demand (power drawn from the grid) of the on-peak half-hour intervals are compared and the interval with the maximum positive site demand value becomes the standard for the house. Assuming, the same load pattern continues for the entire billing period, the specific maximum value of site demand will be the value for which the Demand Charge will be calculated at the end of the billing period. The economic analysis is performed based on this assumption.

1.5 Thesis Outline

Chapter 1 gives an introduction about the problem and answers questions on why the specific problem is important, an update regarding the previous works done to solve this problem, a brief description of the procedure and the assumptions made.

Chapter 2 provides a complete/detailed description of the hardware components of the commercial system installed on the roof of ERC along with important information about the auxiliary equipment used.

Chapter 3 focuses on the literature surveyed and provides detailed pros and cons of every method considered before and a justification for why the currently adopted method is the best for this research.

Chapter 4 is on the first objective – load forecasting. The entire process is described in detail along with the background on the algorithm adopted and the comprehensive explanation on the feature extraction process. The chapter also provides the test results of two data sets in forms of figures and tabulations pertaining to this research.

Chapter 5 is on the secondary objective – The PV prediction and battery control algorithm development. The chapter explains in detail the procedure adopted and the specific reasons for each decision made is provided. The chapter also has the in-depth analysis of the results (of two data sets) obtained from the experiments. There are numerous figures and tabulations to present the data in easy to understand format.

Chapter 6 explains the economic analysis and the results reaffirms the fact that the method adopted is indeed an improvement over the built-in algorithm of the system.

Chapter 7 is the final chapter that provides the conclusion and the scope for future work on this topic.

Finally, all the python programs used for the extraction of data, the control logic for the Raspberry Pi circuit along with the implementation of the algorithms described in Chapters 5 and 6 in Python are attached in the appendix for further reference.

Chapter 2

SYSTEM DESCRIPTION

2.1 The Battery-Supported Grid-Tied PV System

The Engineering Research Center (ERC) at the Arizona State University (ASU) has a 6.36 kW PV system installed on the roof. The hardware was funded by SRP for research purposes. The installed system is battery-supported and also grid-tied which enables the power transfer with the grid (export and import). The Figure 2.1 is a schematic of a similar system.

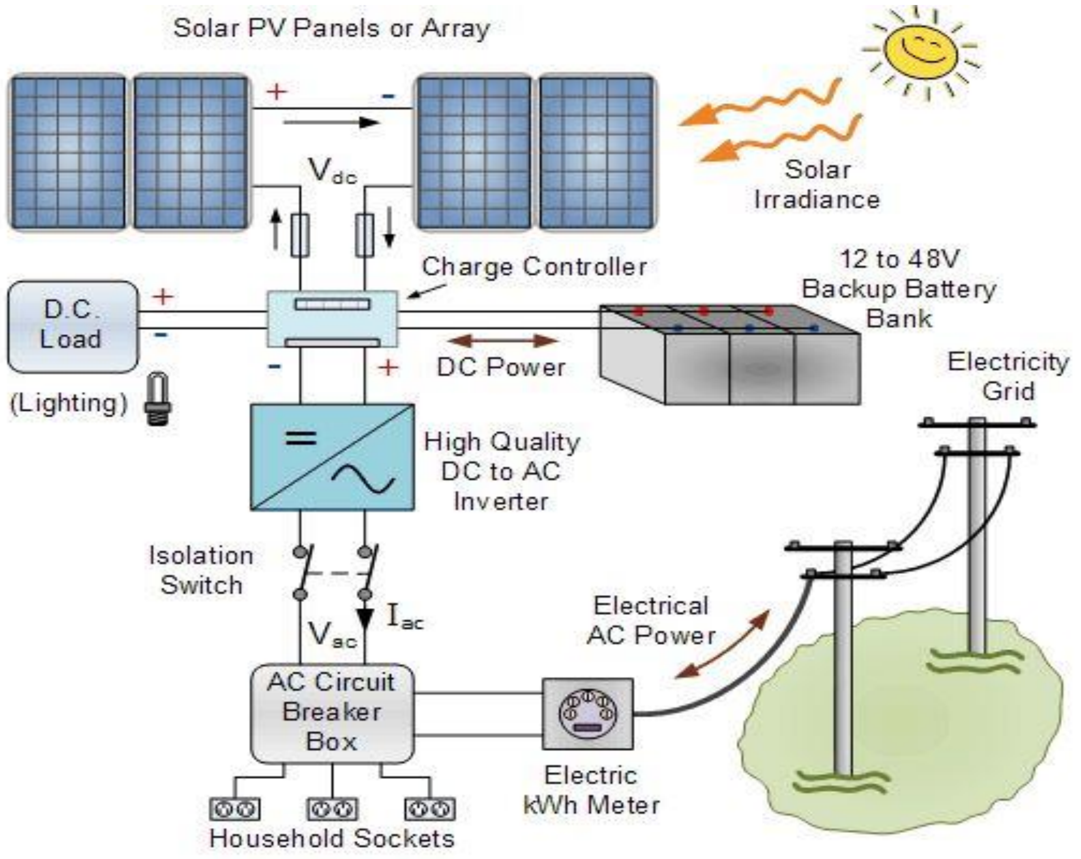


Figure 2.1 Schematic of a Battery-supported Grid-tied PV System. [4]

The main components of this system are:

- The solar panels
- The Sunverge system
    - The charge controllers
    - The storage system, i.e. the battery
    - The inverter
- The custom-built 11 kW load banks
- The Raspberry Pi microcontroller and Relay circuits

The components of the Sunverge system are commercially available together. Each of these components is described in detail in separate sections of this chapter.

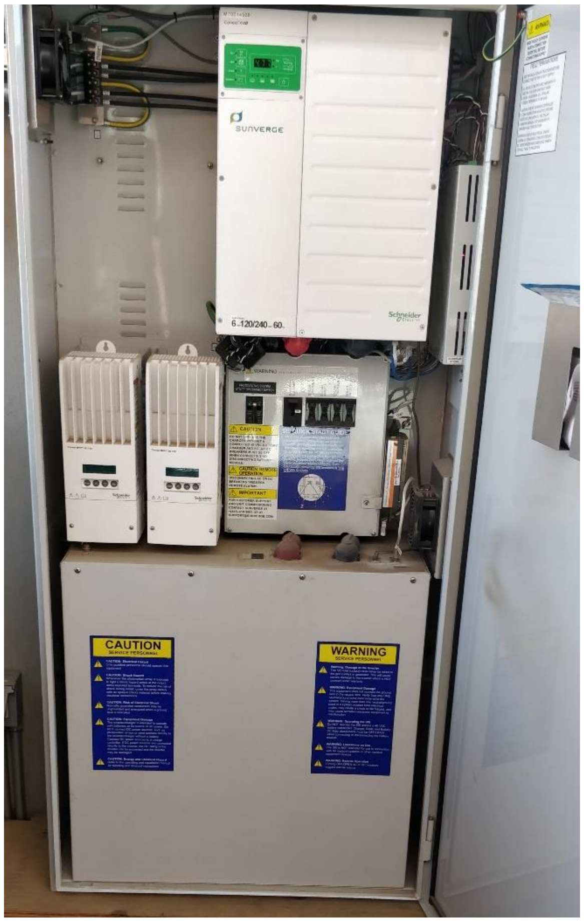


Figure 2.2 The Sunverge system.

2.2 The Solar Panels

The PV system is made up of 24 polycrystalline panels, each rated at 265 W. The panels are divided into two sub-arrays with 12 panels in each sub-array. In each sub-array, three panels are connected in series to form a sub-module. Four such sub-modules are connected in parallel to make up the sub-array.

Table 2. 1 Specifications of a solar panel.

Parameter	Value
Voltage	30.96 V
Current	8.56 A
Power	265 W
Efficiency	19%

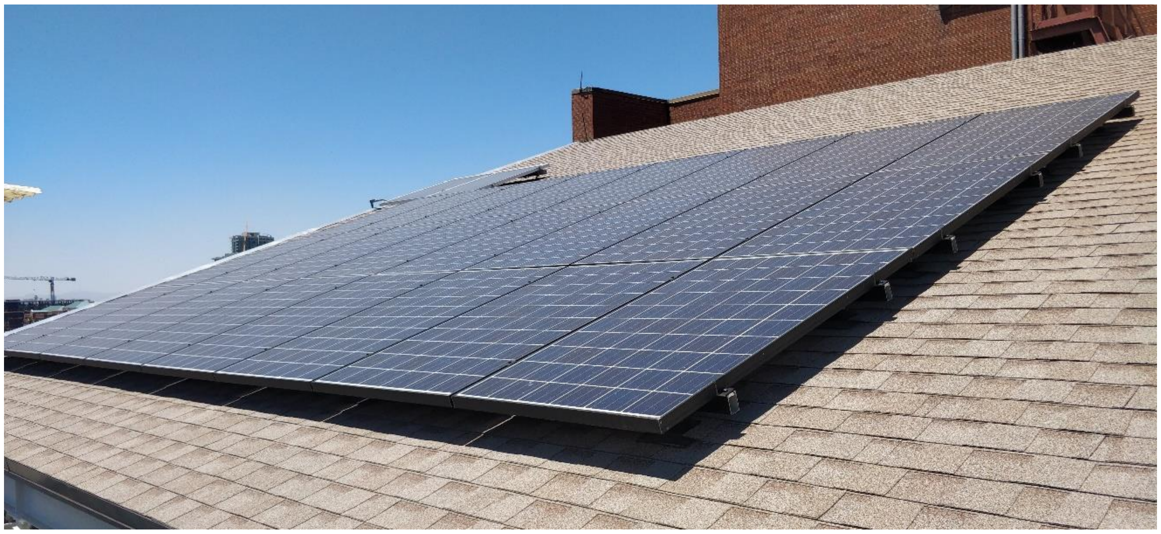


Figure 2.3 Solar panels at the roof of ERC.

2.3 Sunverge system

2.3.1 The Charge Controllers

The first component of the Sunverge system is the Maximum Power Point Tracking (MPPT) charge controllers. There are two such devices (One for each subarray of solar panels) in the commercially available Sunverge system. The main function of these devices is to use an inbuilt algorithm to extract maximum available power from PV module under certain conditions. The voltage at which PV module can produce maximum power is called 'maximum power point' (or peak power voltage). Maximum power varies with solar radiation, ambient temperature and solar cell temperature. [9] The specifications of these devices are given in the Table 2.2.

Table 2. 2 Specifications of a MPPT charge controller.

Parameter	Value
Maximum output power	3500 W
PV array operating voltage	140 V
PV array open-circuit voltage	150 V
PV array short circuit current	60 A

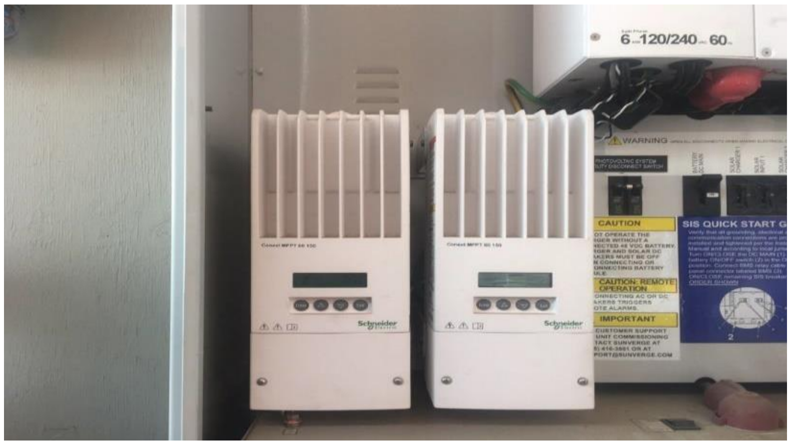


Figure 2.4 The Sunverge MPPT charge controllers.

2.3.2 The Battery

The PV system comes with a power storage device in the form of a Lithium-ion battery ($LiFePO_4$). Owing to better performance and compactness, the Lithium-ion battery is preferred over the lead-acid battery.

Table 2. 3 Specification of the battery.

Parameter	Value
Maximum storage capacity	19.4 kWh
Output DC voltage	48 V

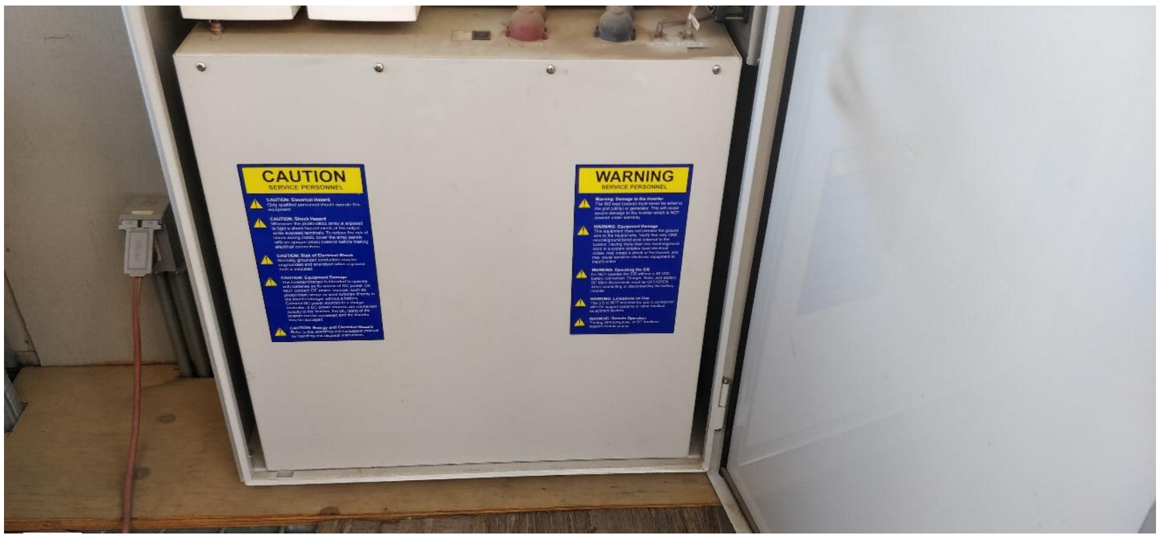


Figure 2.5 The Sunverge battery.

2.3.3 The Inverter

The inverter is used in between the circuit breaker box and the DC sources. Since there is power flow in both the directions, a bi-directional inverter is used. It is an adaptable single-phase and three-phase hybrid inverter with grid-tie functionality. The rating of the inverter determines the maximum output possible from the solar panels and the battery at any point in time. The inverter specifications are tabulated in Table 2.4.

Table 2. 4 Specifications of the inverter.

Parameter	Value
AC nominal power	6000 W
Battery charging voltage	48 V
Battery charging current	100 A
Peak efficiency	95.6%
Operating voltages	120/240 V

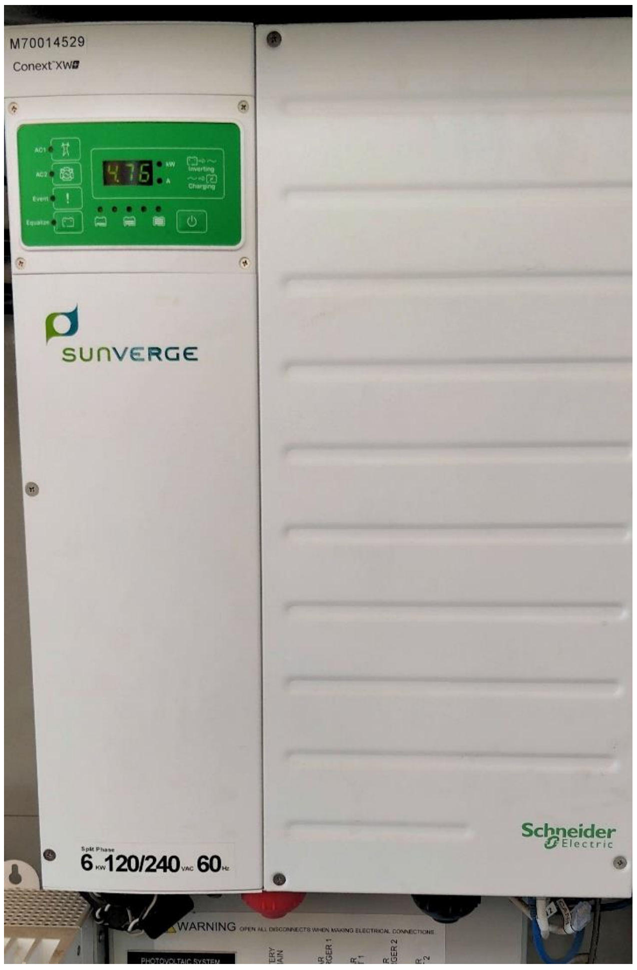


Figure 2.6 The Sunverge inverter

2.4 Load Banks

There are two custom built load banks with a total capacity of 11 kW. The load banks are used to simulate a house and could be connected to the PV system to perform tests. There are two load banks, each used to vary the load in different steps. The smaller load bank is used to vary loads in steps of 72 W while the bigger load bank is used to vary the loads in steps of 880 W. Both the load banks are completely resistive in nature. The smaller load bank is built using commercially available resistors, while the bigger bank is composed of electric burner coils. The materials were chosen primarily based on the cost factor.

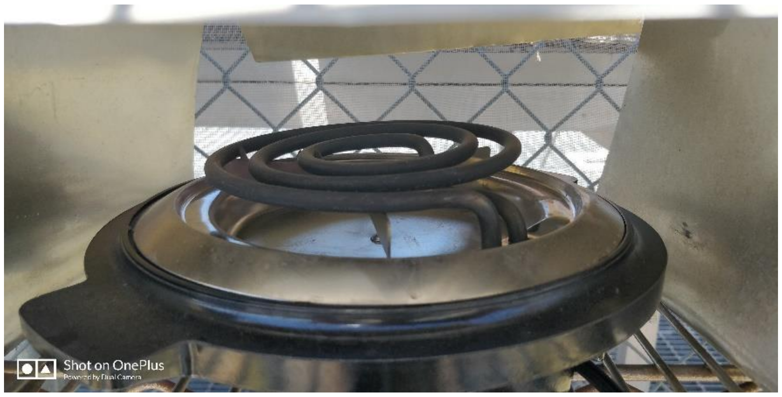


Figure 2.7 The burner coil used as an 880 W step.

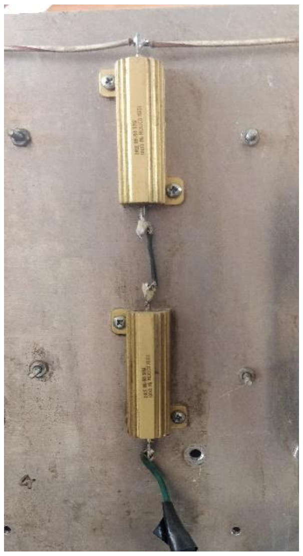


Figure 2.8 A set of resistors used as a 72 W step.

2.4.1 Load Bank 1

This is the smaller load bank and it is made up of 100 ohm resistors. Two 100 ohm resistors are connected in series to form a set (of 200 Ω) and 29 such sets comprise this bank. Each set can dissipate a maximum of 72 W when 120 V is applied across them.

Table 2. 5 Specifications of the smaller load bank.

Parameter	Value
Maximum capacity	2088 W
Minimum step size	72 W
Maximum voltage applied across the set	120 V
Resistance value of each set	200 Ω
Maximum current in each set	0.6 A
Total number of sets	29

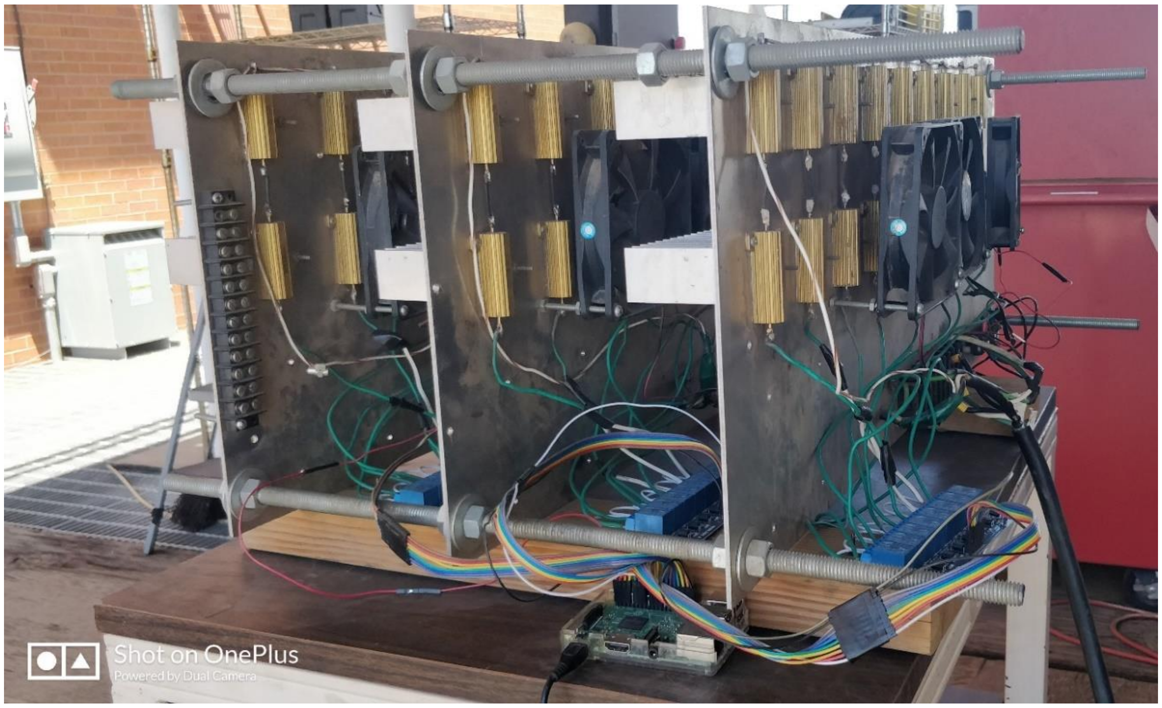


Figure 2.9 The smaller load bank with 29 sets of 72 W steps.

2.4.2 Load Bank 2

This is the bigger load bank and it is made up of electric heater coils. Each device can dissipate around 880 W when a voltage of 120 V is applied. The Table 2.6 tabulates the specification of this load bank.

Table 2. 6 Specifications of the bigger load bank.

Parameter	Value
Maximum capacity	8800 W
Minimum step size	880 W
Maximum voltage applied across a coil	120 V
Resistance value of each coil	16.3 Ω
Maximum current in each coil	7.33 A
Total number of coils	10

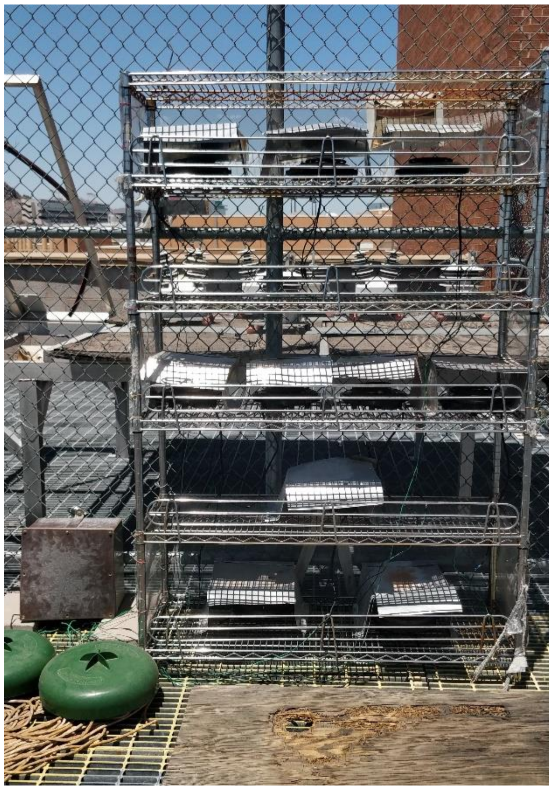


Figure 2.10 The bigger load bank with 10 steps of 880 W each.

2.5 Microcontroller and Relay Circuit

The load banks are switched on and off in different combinations based on the load profile being simulated. The load profile varies every 15 minutes and thus, there is a necessity to vary the loads simulated every 15 minutes. This switching on/off the appropriate coils/resistors is handled by the microcontrollers and the relays. A Python program with specific instruction on the load pattern to follow is used to govern the microcontrollers. The program energizes specific pins of the microcontrollers at specific times. These pins are connected to appropriate relay control switches. By energizing/de-energizing these pins, the relays can be switched on/off. The 880 W coils and the resistor sets are connected to the source through these relays. Thus, the switching on/off the relays connects/dis-connects the loads from the source effectively varying the load profile. Figure 2.11 through Figure 2.15 show the connection diagram and the pictures of the load banks, relays and the microcontrollers.

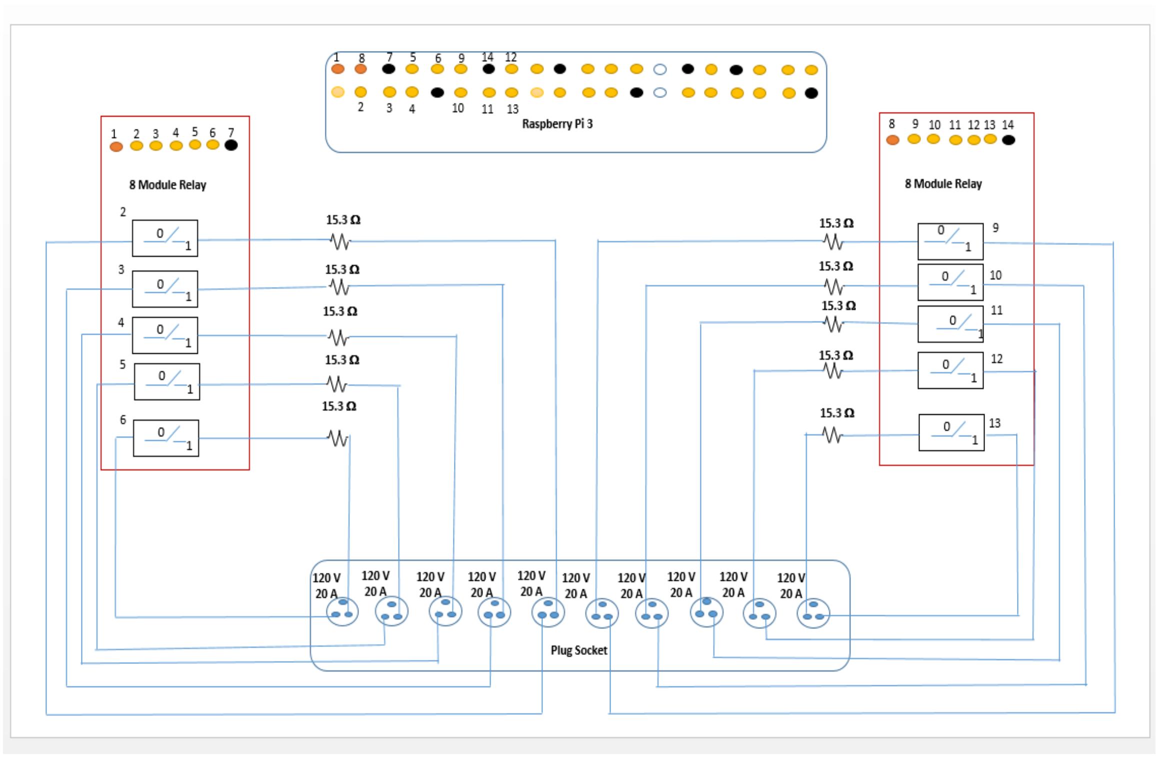


Figure 2.11 880 W step load bank connection diagram. [4]

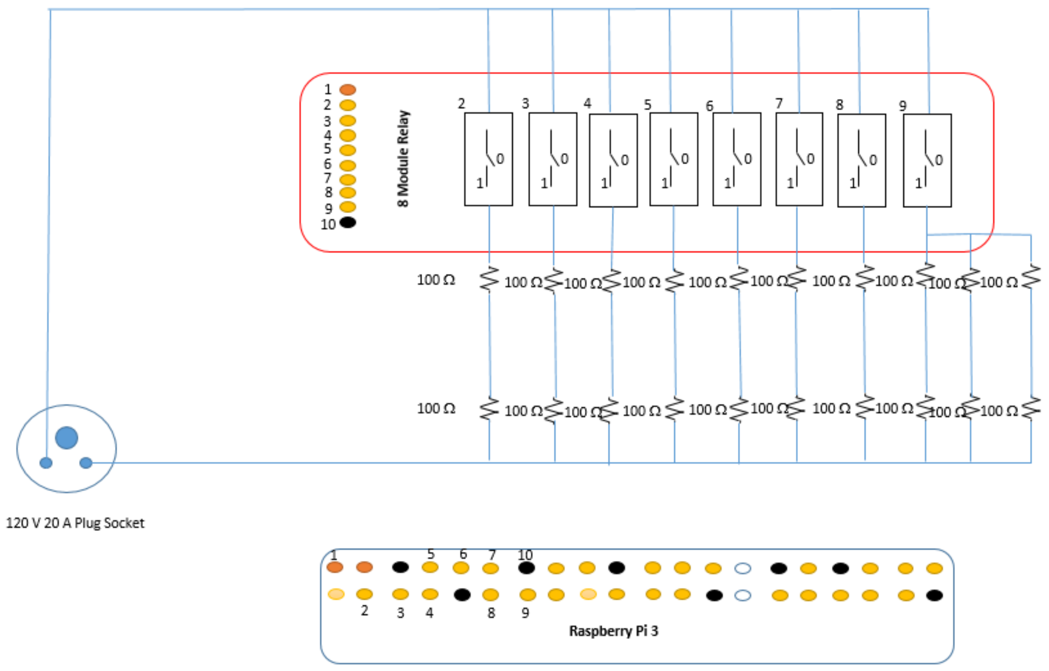


Figure 2.12 72 W step load bank connection diagram. [4]

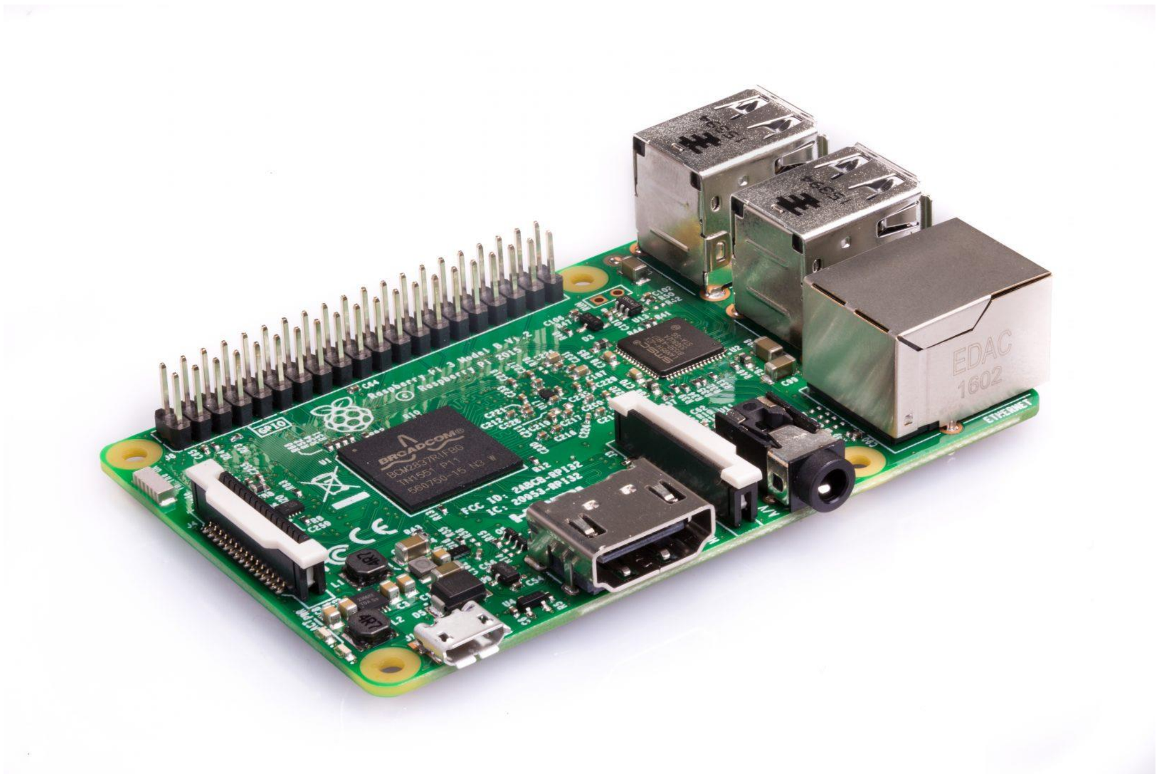


Figure 2.13 A Raspberry Pi 3 microcontroller. [10]

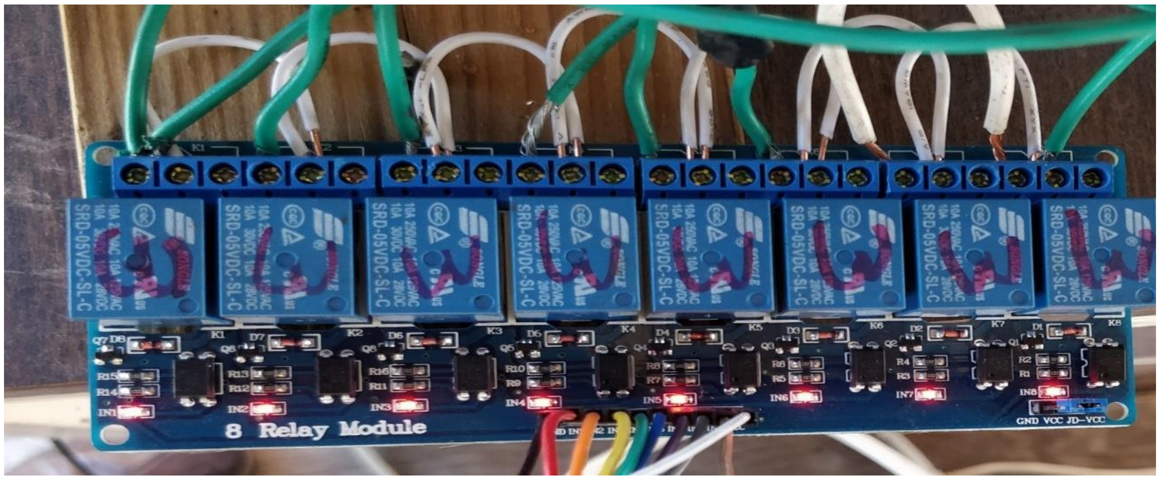


Figure 2.14 8-module digital relay – smaller load bank.

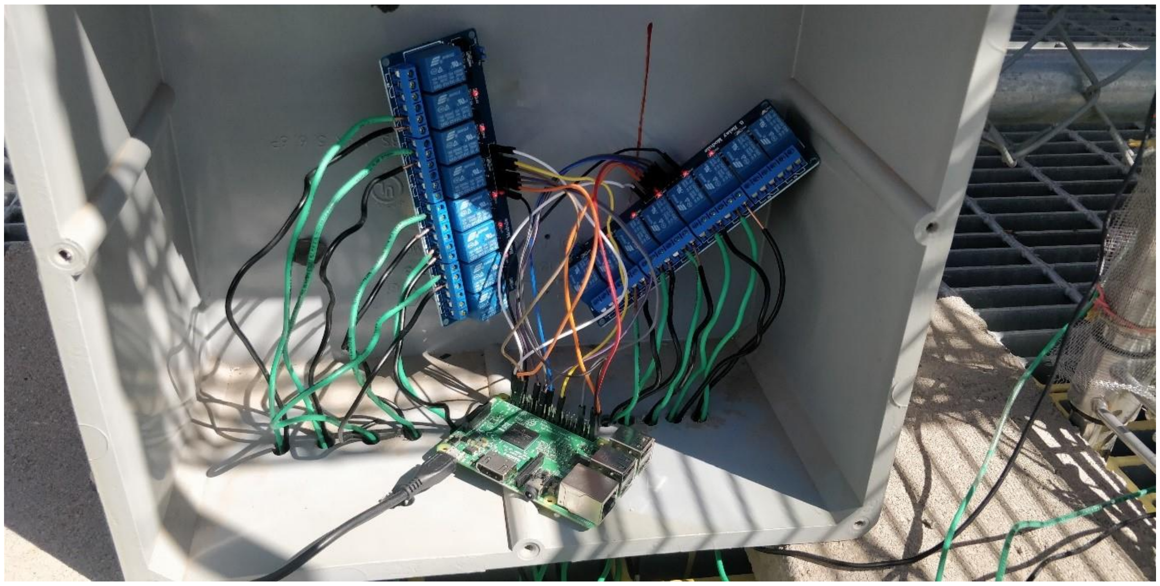


Figure 2.15 8-module relay controlled by the Raspberry Pi – bigger load bank.

Chapter 3

LITERATURE SURVEY

3.1 Introduction

The fossil fuel reserves of the world are being exploited at a rate never seen before, owing to the increased global demand for energy. Apart from the environmental issues like pollution and global warming [11], these fuels are bound to get exhausted in the near future and there is a chance for a global energy crisis, if alternate sources of energy cannot supply the global demand. The main problem that limits the extent of reach of the renewable sources is the reliability issue.

Directly or indirectly, most of the renewable energies are derived from the sun. The hydro power generation depends on monsoons largely, which are a part of the water cycle involving the sun. Similarly, winds are caused due to the uneven heating of the earth's surface by the sun. One thing in common with these indirect sources of solar energy is the huge uncertainty accompanying them. In a state like Arizona, the solar energy is directly available in abundance through most parts of the year. The uncertainty factor is greatly reduced if not eliminated here. With the renewable penetration target, mentioned earlier, in mind, the utilities have started offering customers a chance to tap into this seemingly infinite potential source.

The major factors that hinders the maximum utilization of this energy source are the cost of the equipment and the efficiency. The utilities are providing its customers with incentives to make the solar program more attractive. These incentives, in the form of credits, enable customers to earn money by operating the system and at some point, in time, returns their initial investment in procuring the system. The time taken for a customer to earn back the investment is called the investment return period. According to [4], the investment return period for the Sunverge system installed at the ASU campus is 25 years including the cost of the battery. This is one of the major challenges this research work is trying to conquer. 25 years to earn back the investment made seems a long time. Customers may thus, hesitate to make the switch owing to this factor.

One way to approach this problem is to find the major influencing factor in determining the return period. Given the fact that the research is being performed on a commercial system, the changing of the hardware is beyond the scope of this collaborated research. Therefore, the students at the Arizona State University developed a technique to better monitor the system [3]. This resulted in a better investment return period of 17 years for the customer. The ASU algorithm still had scope for improvement. The thesis presented here is a compilation of results of such work performed in the spring of 2018.

3.2 Related Work

Short term load forecasting has been tried before by various authors for different purposes. Based on the papers reviewed and conversations with experts through forums and articles, it is evident that Recurrent Neural Networks (RNN) seem to be the best technique to adopt when dealing with timeseries data. The author of [13] provides ample evidence for the same. The deep learning techniques provide reliable forecast of the data. But owing to the complexity involved in this process and also the fact that, this system in study does not have a working ML algorithm in place, a more simpler approach was taken for the purpose of this study. The study by the authors of [14] in the Indian city of Bangalore was more closely related to the approach taken in this thesis. The authors had tried to predict the load pattern of a specific university building by utilizing past two years data as the test data. They had published [14] by comparing the performance of various ML techniques in this scenario. The important inference related to this thesis from [14] was that the Random Forest technique adopted in this thesis was a relatively simple and reliable technique. When pitched against the Deep Neural Network (DNN) techniques, the Random Forest algorithm performed fairly. Another interesting study in [15] probed the various factors that might influence the load pattern. The author provides the temperature and humidity pattern of the area as a feature in the effort to develop a better load forecasting model. The [15] along with other articles and educated guesses lead to the extraction of the features utilized in this thesis.

All the articles mentioned here were forecasting load, but, none of them had the same application as this project. The uniqueness of this requirement of trying to control a commercial hardware system without making any external changes to the hardware was a big challenge faced. Also, the conditions unique to Arizona made factors like humidity less significant to be passed as a unique feature. A better understanding of the climate and behavior of the people of the state of Arizona was required to analyze and extract the features for this problem.

The author in [16] has pitched the logical solution to beating the on-peak hour charges. A similar approach has been adopted in this thesis as well. The conditions again differed from [16]. The idea for the battery control algorithm was primarily from interactions with the author of [3]. The work done in [3] and [4] were attempts at identifying the actual cause of prolonged return period. Since the equipment is a commercially available system, the built-in control algorithm is not revealed to the public. The author had thoroughly studied the system to be reasonably able to predict its behavior under various circumstances. This along with the conversations with the system provider helped decide a possible solution to this situation.

The primary agenda of this thesis is to establish a working control model that outperforms both the built-in algorithm and the algorithm developed in [4]. With the initial idea in place, further developments of this model would be easier. The Chapters 4 and 5 would cover this developed model in great detail.

Chapter 4

LOAD FORECASTING

4.1 Introduction

The power requirement from the battery at any given point during the on-peak hours is based on the predictions of the load at that point in time. Thus, the efficiency of the battery control algorithm will largely depend on the load forecasting results. As explained earlier, another objective of this research work is to decelerate the cycling aging of the battery by controlling the charging/discharging cycles the battery undergoes. This algorithm is based on the load prediction results as well. The main factor inhibiting the performance of the built-in algorithm and the ASU algorithm developed previously was the accuracy of the load prediction methodology adopted. This chapter explains the feature extraction process, the algorithm selection process and the assumptions made for the research involving the Machine Learning algorithm.

4.2 Dataset Description

The dataset describing the one-year load pattern of a “Stratum Three” customer/house in the state of Arizona was provided by the sponsoring utility specifically for this research. It was cleaned of any personal information of the customer. The dataset contains sets of 15-minute load profile (average kW value) of the house along with the temperature data and the timestamp providing the necessary information about any specific 15-minute interval. Thus, there are 4 datapoints pertaining to every hour, 96 data points per day and 35,040 points for the entire year. This specific dataset is chosen for simulation because, it has already been used by the students of ASU in 2016-2017 to simulate the load pattern of the house on Sunverge's system thus, the built-in algorithm is familiar with the entire dataset. Since the built-in algorithm predicts loads based on its past experiences, the prediction it comes up for the day of simulation will be based on its deduction of this specific dataset. Such a prediction by the built-in algorithm would be the best it can make for the given day. This pitches the built-in algorithm and the newly developed Machine Learning algorithm, which will be fed the entire one year load data as training data, on equal grounds for a direct comparison of the performances.

4.3 Algorithm Description

The load forecast has been obtained from the Random Forest Classification algorithm. The "scikit-learn" implementation of the algorithm is used for this study. The following section briefs the general working of the algorithm.

The Random Forest algorithm is an extension of the Decision Tree algorithm. Decision Tree is a Supervised Machine Learning algorithm, which solves the problem by adopting the representation of a tree to model the problem. Each internal node of the tree is an attribute and the leaves of the tree are the classification labels. The best attribute of the data set is placed at the roots of the tree. This is where, the prediction of the label for a record starts. The root attribute and the record's attribute values are compared. Based on the true or false result obtained, the appropriate branch is followed. This leads to the next node. The record's attribute values are compared with the next internal node. This again results in a true/false answer and we follow the appropriate branch to the next internal node. This process is continued till we reach a leaf (label). Once we reach the leaf, the classification of the record is complete.

The disadvantage of the method is, each tree is constructed based on a data source and since no model is perfect, the constructed tree model has an error. Since we base our classification of the record, on the output of a single tree, the classification is prone to errors as well. The error can be significantly reduced, by considering the decisions from not one but multiple trees and taking an average. This solution is nothing but the Random Forest algorithm.

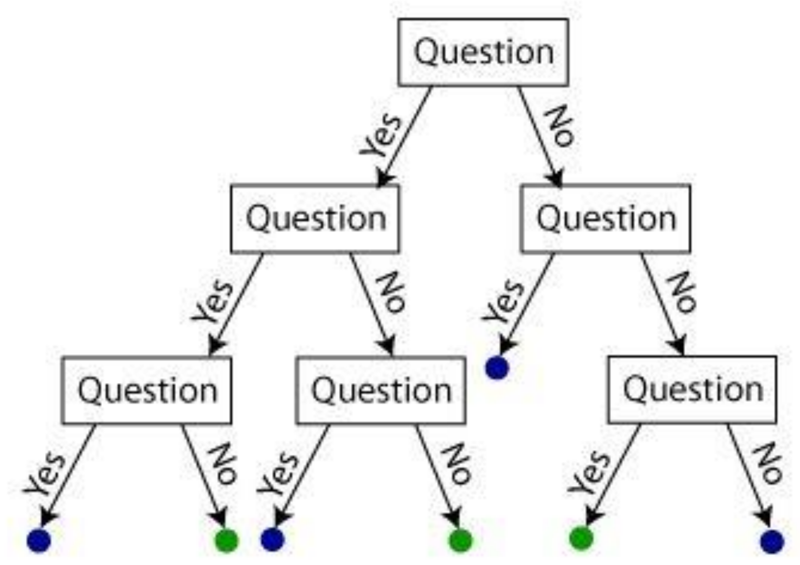


Figure 4.1 The structure of a Random Forest tree. [12]

A Random Forest algorithm is a collection of such Decision Trees. When a new object from the input vector needs to be classified, the object is fed into each tree of the forest and the classification decision of each tree is considered. The classification with the majority of the 'votes' from the trees is chosen as the most appropriate one [5]. This reduces the possibility of an erroneous classification by a huge margin. The decision of the number of trees in the forest depends on the memory constraints. From the analysis, it is clear that more the number of trees, the better the algorithm works (again, this is true until an optimal point, beyond which, the accuracy decreases). But, the number of trees in the forest is limited by the amount of memory required, to process the output from so many trees. We therefore, choose the number of trees in the forest based on obtaining a reasonable accuracy with reasonable memory requirement. Ultimately, it is a trade-off which depends on the problem/data set on hand.

4.4 Feature Extraction

The accuracy with which the algorithm predicts the load for a specific day, actually decides the extent of the economic benefit. The previously employed weighted K-means clustering [3] had an issue — The load forecasting was treated as an Unsupervised Machine Learning problem whereas a better classification would be under the Supervised Machine Learning category. The reasons being:

- Availability of labeled data.
- Possibility of a feedback.
- Objective is to predict the outcome/future.

More specifically, the load forecasting problem is a Supervised Learning — "Regression" problem. But, for the sake of simplicity, the problem has been treated as Supervised Learning — "Classification" problem in our case. This is achieved by rounding-off the output power to the nearest 200 W value, which results in the creation of "labels" or groups into which each prediction could be classified. The grouping though, may introduce a maximum deviation of +/- 100 W from the actual value of the load. For example, a load of 299 W will be classified as 200 W and a load of 301 W will be classified as 400 W. This is acceptable as the system we are simulating has a maximum capacity of 11 kW and 100 W is thus, a mere ~1% error.

To better model the data set, six features have been extracted and utilized. The timestamp data has been stripped and five specific features have been extracted. The year value is not utilized as a feature since, same days of a month that are years apart have very similar load pattern. The six features utilized are:

- Temperature — The feature is part of the data set. The temperature pertaining to every 15-minute interval is recorded and provided.
- Month of the year — The months in a year have been clustered together appropriately, based on the average temperature observed from weather data, pertaining to the past 100 years in the state. (See appendix for the clustering details).
- Day of the year — The days of the year are numbered from 1 through 365. Day number 1 and 365 (January 1st and December 31st — Peak winter in Arizona) have more similarity to each other than to day number 180 (end of June — Peak summer in Arizona). To convey this, instead of passing the number of the day in the year, the absolute value of the standard deviation of the number of the day in the year from the mean of 1 through 365 is passed. This ensured the similarity of day 1 and day 365 (both are 180).
- Day of the week — The load pattern from weekday to weekend varies greatly. Therefore, the information pertaining to whether the given day was a weekday or a weekend, is passed as a feature.
- Holidays — The load pattern varies, if a given day was part of a long-weekend or the holiday season. This information too is passed as a feature.

- Time of the day — The load pattern of a house normally varies every hour. The time of the day plays a huge part in determining the load pattern of that particular house. This is extracted and passed as a feature.

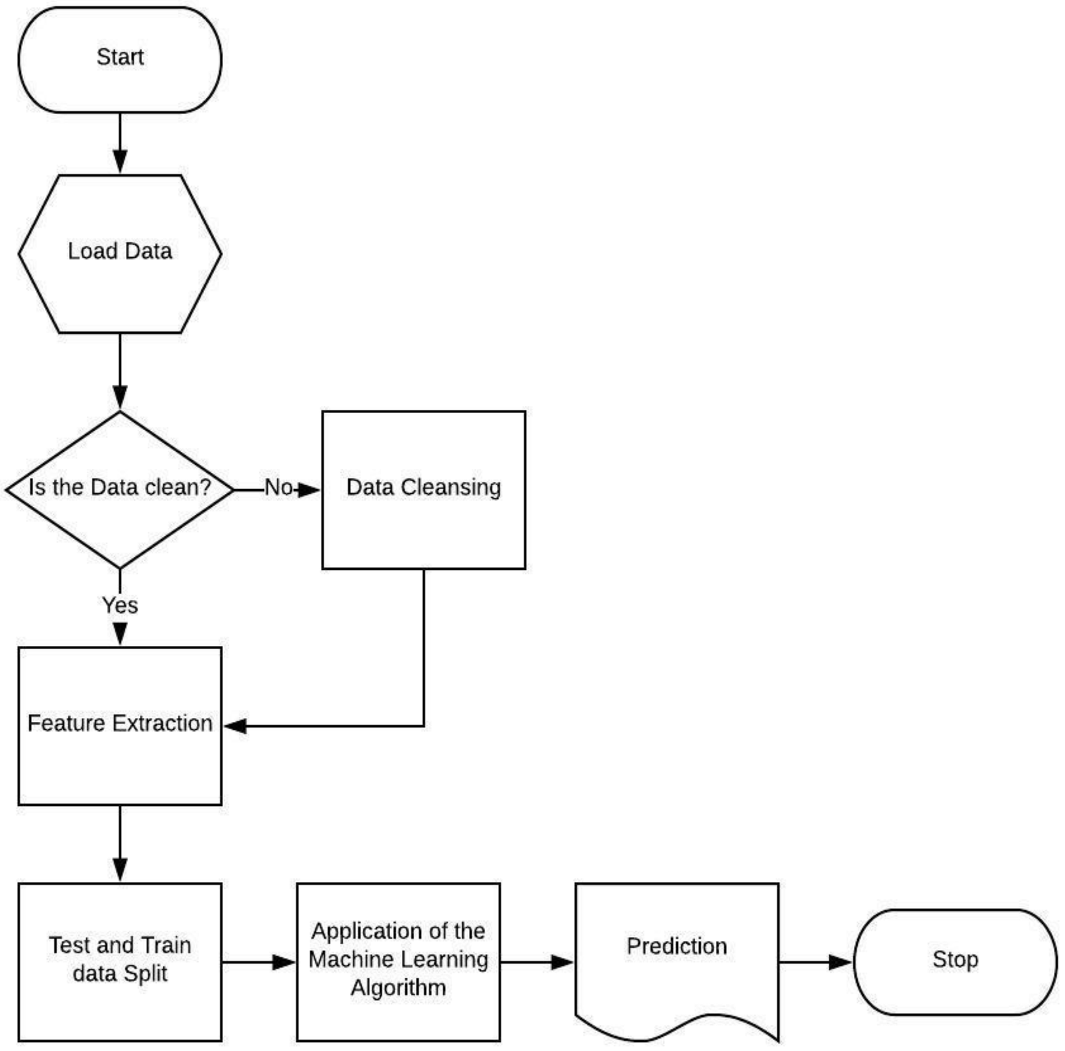


Figure 4.2 Flowchart of the load forecasting procedure.

4.5 Results

4.5.1 Machine Learning Algorithm

Seven different Supervised Learning Classification algorithms are simulated after passing the said features. The K-nearest neighbors, Support Vector Machines ("rbf" kernel) and Random Forest are the algorithms that gave the most promising results. The "scikit-learn" implementation of these algorithms has been utilized, owing to performance and reliability. Results from the Random Forest are used as a base for further analysis, owing to its higher accuracy in predicting the grouped samples.

Table 4. 1 Accuracy of the three Supervised Learning algorithms for the 1st set of simulations.

Algorithm	Accuracy (200 W grouping)
SVM – 'rbf' kernel	78%
K-Nearest neighbor	73%
Random Forest	82%

Table 4. 2 Accuracy of the three Supervised Learning algorithms for the 2nd set of simulations.

Algorithm	Accuracy (200 W grouping)
SVM – "rbf" kernel	79%
K-Nearest neighbor	79%
Random Forest	80%

4.5.2 The Load

The Tables 4.3 – 4.8 compare the actual load value existing on the days of simulation during the on-peak hours with the predicted values of the ML algorithm and the built-in algorithm.

Note: All the time intervals dealt with here are half-hour clusters and the parameter values are half-hour averages.

Table 4. 3 Tabulation comparing the actual load with the predicted values pertaining to the 1st set of simulations.

Interval	Actual load (kW)	ML algorithm prediction (kW)	Built-in algorithm prediction (kW)
13:00	1.621	1.6	1.79
13:30	1.73	1.8	1.886
14:00	1.844	1.8	1.974
14:30	1.963	2	2.037
15:00	2.107	2.3	2.158
15:30	2.251	2.6	2.257
16:00	2.374	2.6	2.339
16:30	2.48	2.6	2.384
17:00	2.553	2.6	2.427
17:30	2.571	2.6	2.416
18:00	2.539	2.6	2.37
18:30	2.451	2.4	2.299
19:00	2.373	2.6	2.282
19:30	2.35	2.4	2.254

Table 4. 4 Tabulation comparing the actual load with the predicted values pertaining to the 2nd set of simulations.

Interval	Actual load (kW)	ML algorithm prediction (kW)	Built-in algorithm prediction (kW)
13:00	1.643	1.6	1.053
13:30	1.721	1.8	1.071
14:00	1.782	1.8	1.108
14:30	1.857	1.9	1.164
15:00	1.955	1.9	1.243
15:30	2.064	2.2	1.340
16:00	2.160	2.2	1.439
16:30	2.25	2.2	1.532
17:00	2.335	2.2	1.61
17:30	2.341	2.2	1.651
18:00	2.3	2.2	1.667
18:30	2.206	2.2	1.661
19:00	2.119	2.2	1.691
19:30	2.083	2.1	1.736

The data presented in the Tables 4.3 and 4.4 are plotted and presented as Figure 4.3 and Figure 4.4. It is evident from the tables and the graphs that the prediction obtained using the ML algorithm is more loser to the actual value of the load than the built-in algorithm's predictions.

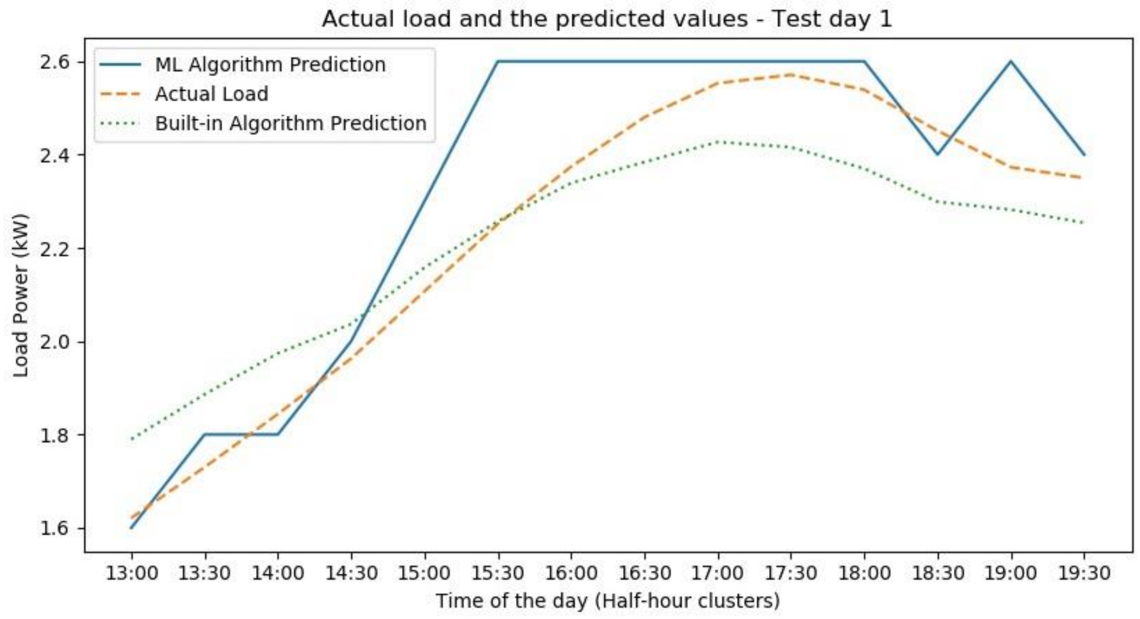


Figure 4.3 Plot of the actual load and the predictions made by the ML and the built-in algorithms for the load profile simulated on test day 1.

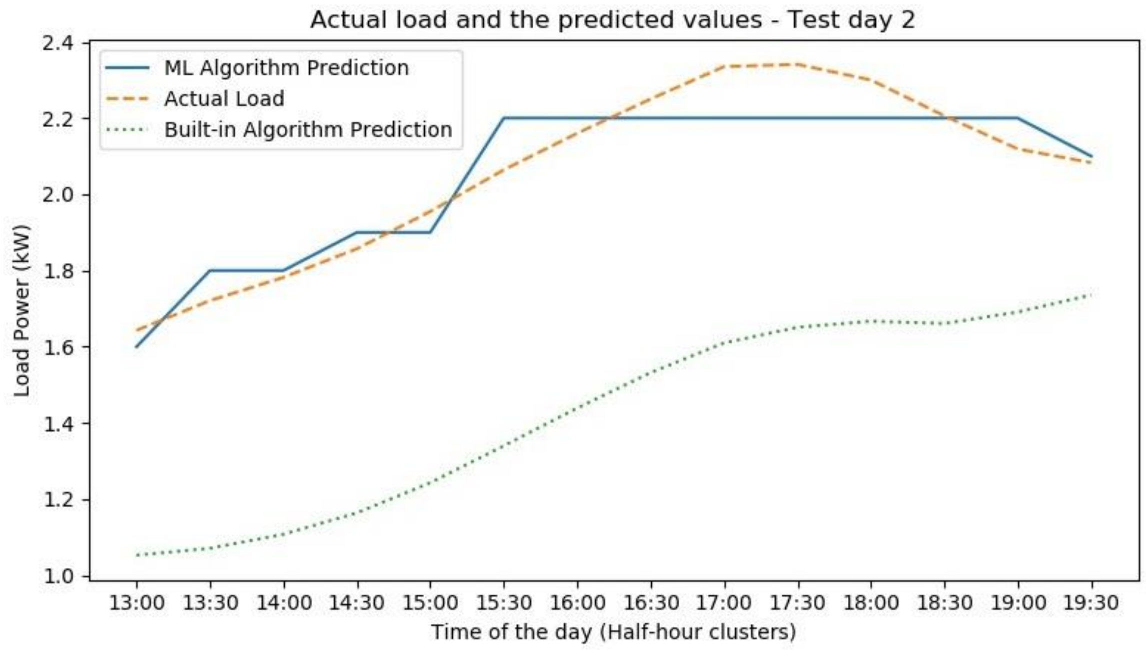


Figure 4.4 Plot of the actual load and the predictions made by the ML and the built-in algorithms for the load profile simulated on test day 2.

The Tables 4.5 and 4.6 show the magnitude of the deviation between the actual load values and the predicted load values using the ML algorithm. Since it is the magnitude if the deviation, the sign of the value is omitted. The 1st test day was based on the load data pertaining to the 1st of May and the 2nd test day was based on the data pertaining to 9th of May.

Table 4. 5 Tabulation of the deviation of ML algorithm predictions from the actual load values for test day 1.

Interval	Actual load (kW)	ML algorithm prediction (kW)	Deviation (kW)
13:00	1.621	1.6	0.021
13:30	1.73	1.8	0.07
14:00	1.844	1.8	0.044
14:30	1.963	2	0.037
15:00	2.107	2.3	0.193
15:30	2.251	2.6	0.349
16:00	2.374	2.6	0.226
16:30	2.48	2.6	0.12
17:00	2.553	2.6	0.047
17:30	2.571	2.6	0.029
18:00	2.539	2.6	0.061
18:30	2.451	2.4	0.051
19:00	2.373	2.6	0.227
19:30	2.35	2.4	0.05

Table 4. 6 Tabulation of the deviation of ML algorithm predictions from the actual load values for test day 2.

Interval	Actual load (kW)	ML algorithm prediction (kW)	Deviation (kW)
13:00	1.643	1.6	0.043
13:30	1.721	1.8	0.079
14:00	1.782	1.8	0.018
14:30	1.857	1.9	0.043
15:00	1.955	1.9	0.055
15:30	2.064	2.2	0.136
16:00	2.160	2.2	0.04
16:30	2.25	2.2	0.05
17:00	2.335	2.2	0.135
17:30	2.341	2.2	0.141
18:00	2.3	2.2	0.1
18:30	2.206	2.2	0.006
19:00	2.119	2.2	0.081
19:30	2.083	2.1	0.017

From the Tables 4.5 and 4.6, it can be observed that, the maximum deviation generally occurs during the mid-afternoon period. This can be attributed to the fact that the load pattern seems to be on a continuous increase and the prediction has been rounded to the nearest 200 W value. This error could be further reduced if the 200 W margin is reduced. But, in this scenario, one can observe that this deviation is very small and does not contribute to large error in the results.

The Tables 4.7 and 4.8 provide the comparison between the actual value of load and the value of the load simulated by hardware. Again, as before, all the load values are average values over half-hour intervals. The two tables provide the simulated load values for each test day. Finally, Figure 4.5 and 4.6 are plotted based on the data from the tables and they pictorially describe the load tracing capabilities of the hardware system.

Table 4. 7 Tabulation comparing the intended load value and the simulated load value for test day 1.

Interval	Actual load (kW)	Output from the hardware (kW)	Deviation (kW)
13:00	1.621	1.553	0.068
13:30	1.73	1.637	0.093
14:00	1.844	1.726	0.118
14:30	1.963	1.912	0.051
15:00	2.107	2.002	0.105
15:30	2.251	2.159	0.092
16:00	2.374	2.259	0.115
16:30	2.48	2.367	0.113
17:00	2.553	2.416	0.137
17:30	2.571	2.475	0.096
18:00	2.539	2.422	0.117
18:30	2.451	2.346	0.105
19:00	2.373	2.294	0.079
19:30	2.35	2.144	0.206

Table 4. 8 Tabulation comparing the intended load value and the simulated load value for test day 2.

Interval	Actual load (kW)	Output from the hardware (kW)	Deviation (kW)
13:00	1.643	1.583	0.06
13:30	1.721	1.635	0.086
14:00	1.782	1.7	0.082
14:30	1.857	1.757	0.1
15:00	1.955	1.853	0.102
15:30	2.064	1.838	0.226
16:00	2.160	1.909	0.251
16:30	2.25	2.009	0.241
17:00	2.335	2.207	0.128
17:30	2.341	2.179	0.162
18:00	2.3	2.182	0.118
18:30	2.206	2.067	0.139
19:00	2.119	2.065	0.054
19:30	2.083	2.071	0.012

From the Tables 4.7 and 4.8, we can observe the actual hardware output value to have deviated from the actual value. There can be a few reasons why this happened. All the calculations for the number of units to switch on/off were based on the rated output values of the resistors/coils. Thus, practically, there could be deviations in those values. The system tries to optimize between three energy sources. Based on the availability, there is switching happening between these three sources every four seconds. This contributes to transients that distorts the output power momentarily.

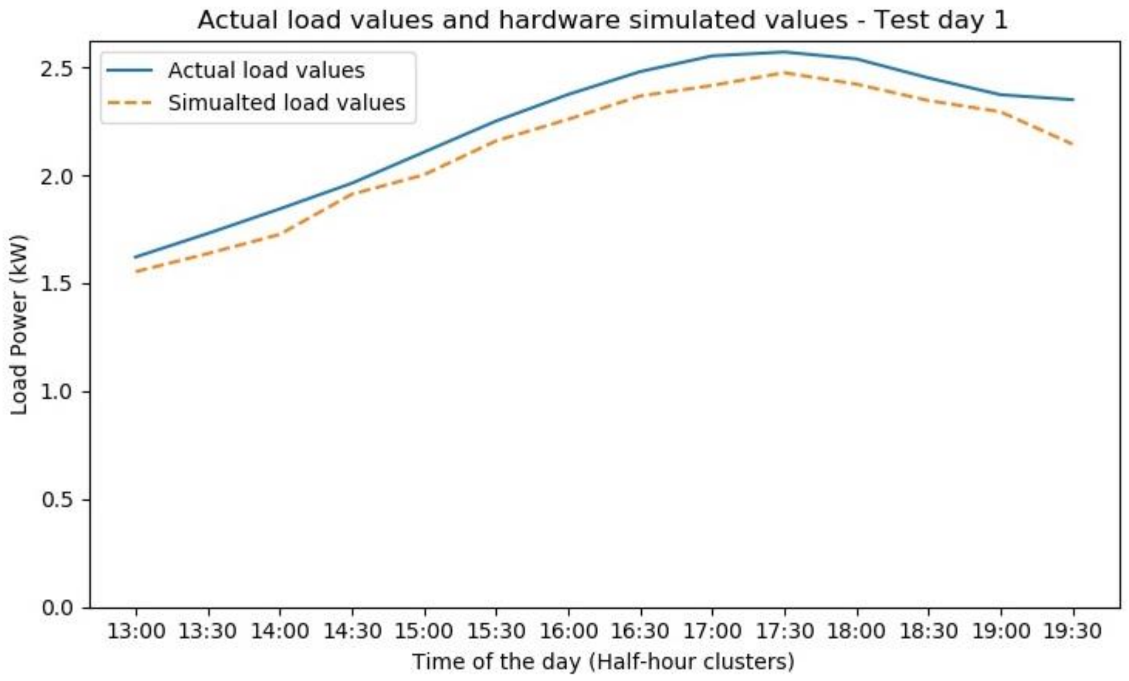


Figure 4.5 Plot of the actual load and load simulated on the hardware on test day 1.

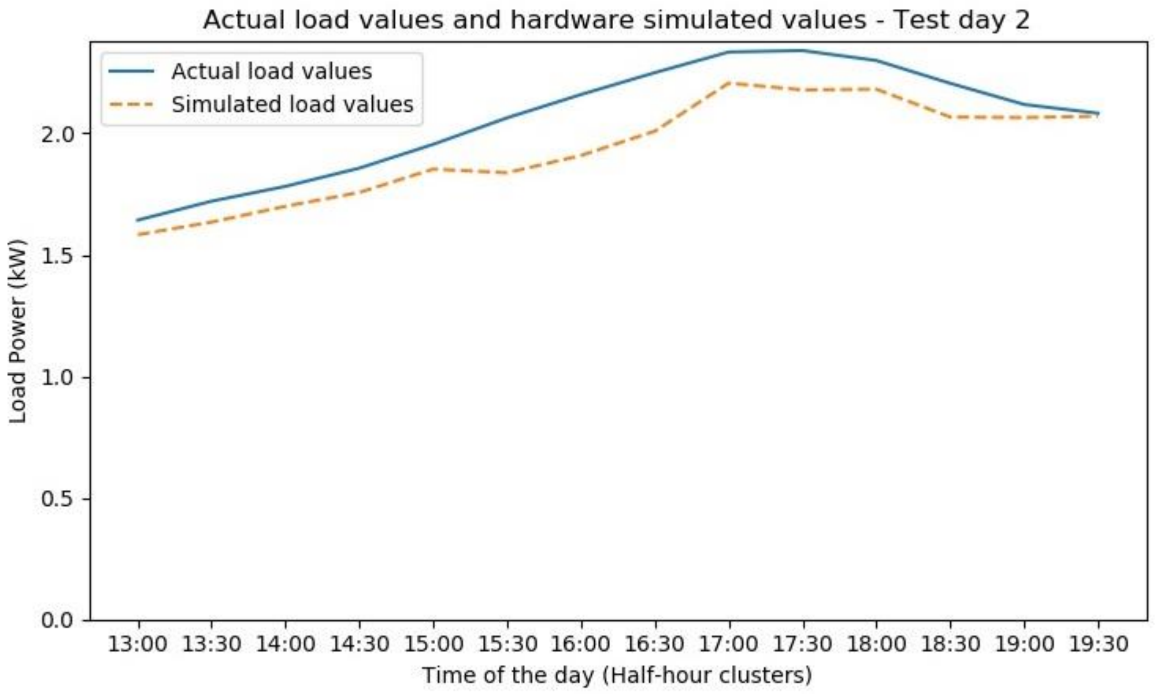


Figure 4.6 Plot of the actual load and load simulated on the hardware on test day 2.

4.6 Inference

The load forecasting is of prime importance and this chapter leaves us with a lot of clues on what to expect when the economic analysis is performed on the results of the battery control algorithm. It is evident that the load forecasting is the basic structure that influences the efficiency of the other algorithms working on top of it. Any error here simply propagates through the entire process and ultimately magnifies the error component in the result thus, resulting in a higher electricity bill for the customer. The features extracted from the timestamp data are the building blocks of this algorithm. The chapter presents compelling evidence in support of these arguments.

Chapter 5

BATTERY CONTROL ALGORITHM

5.1 Introduction

The battery control algorithm that has been developed, is used to decide the amount of power, the battery needs to provide the load, at every point in time during the on-peak hours while also deciding on whether or not to charge the battery for the next day. This decision requires a prediction of the PV availability along with the load forecast, during the specified on-peak hours. The following sections are explanations of the procedures adopted and decisions made.

5.2 PV Prediction

The PV availability during the on-peak hours of the specific day, is a required parameter. This data is obtained from NREL “PVWATTS Calculator”. It is a web-application that predicts the PV output of a system based on the parameters listed in Table 5.1 [6]. The parameters are passed as input and the hourly prediction is obtained.

Another method employed for the prediction of PV is, using the Direct Normal Irradiance (DNI) values obtained from the NREL website for Arizona [7]. The obtained values are processed using the “PVLib" library in Python. The DNI values from NREL are based on research data collected over a 25-year period from 1977 through 2002. It has 8762 data points with each data point pertaining to a specific hour of a specific year in the period mentioned above. The DNI values are multiplied by the size of the system to obtain the solar generation of the system in watts. Further, factors like the ones listed in Table 5.1 including the DC to AC conversion ratio, the angle of the solar panel inclination etc. have been utilized for obtaining the output value of the solar power produced by the system installed at the ERC.

Table 5. 1 List of parameters passed to the NREL "PVWATTS Calculator"

Parameter	Value
Latitude (deg N)	33.45
Longitude (deg W)	111.98
Elevation (m)	337
DC System Size (kW)	6.36
Module Type	Standard
Array Type	Fixed (roof mount)
Array Tilt (deg)	20
Array Azimuth (deg)	180
Invert Efficiency (%)	96
DC to AC Size Ratio	1.2
Latitude (deg N)	33.45

The results of these two methods are very similar to each other and therefore, the results of the "PVWATTS", is taken as the predicted PV value of a specific hour. A similar year-long study comparing the predictions of "PVWATTS" and the actual output generated by the commercial system installed at the Arizona State University was performed at the beginning of 2018 (for 2016-17). The results proved that the

PVWATTS" predictions are very close to the actual values obtained. The result of this study is presented in the Figure 5.1.

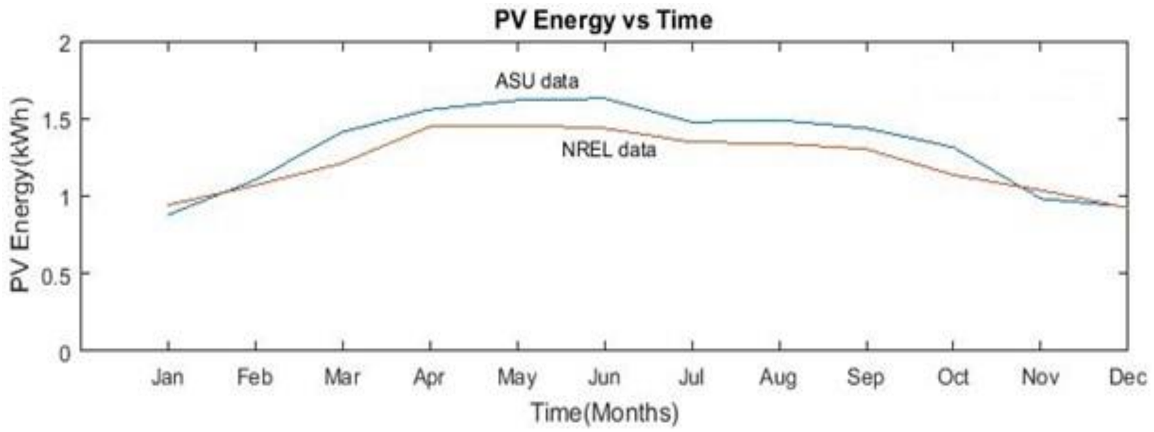


Figure 5.1 Comparison of PVWATTS predicted value and the observed value at the site for 2017.

5.3 Battery Control Algorithm

Previously, the battery was completely charged every night and the power was equally divided between the on-peak hours of the following day [3]. This method poses a problem. The utility's on-peak hours during the summer are from 1 pm through 8 pm. But, the solar energy is unevenly distributed during these hours. Thus, the power from the battery might be needed more during the latter part of the day i.e. the evenings. Therefore, the ideal algorithm would try to supply the load through the PV first, then, a part of the deficit must be made up by the battery and any further deficit should be made up with power from the grid.

The capacity of the battery is split, based on the ratio of effective load (load in watts — available solar power in watts) at each half-hour interval compared to the load of the entire on-peak hours of the day. The reason such a method has been chosen is, to ensure that the Peak Demand of any particular half-hour does not stand out from the rest and is as uniform as possible. This is the working logic behind this algorithm. The load in watts is predicted by the load forecasting algorithm and similarly, the availability of PV is predicted using the PV prediction.

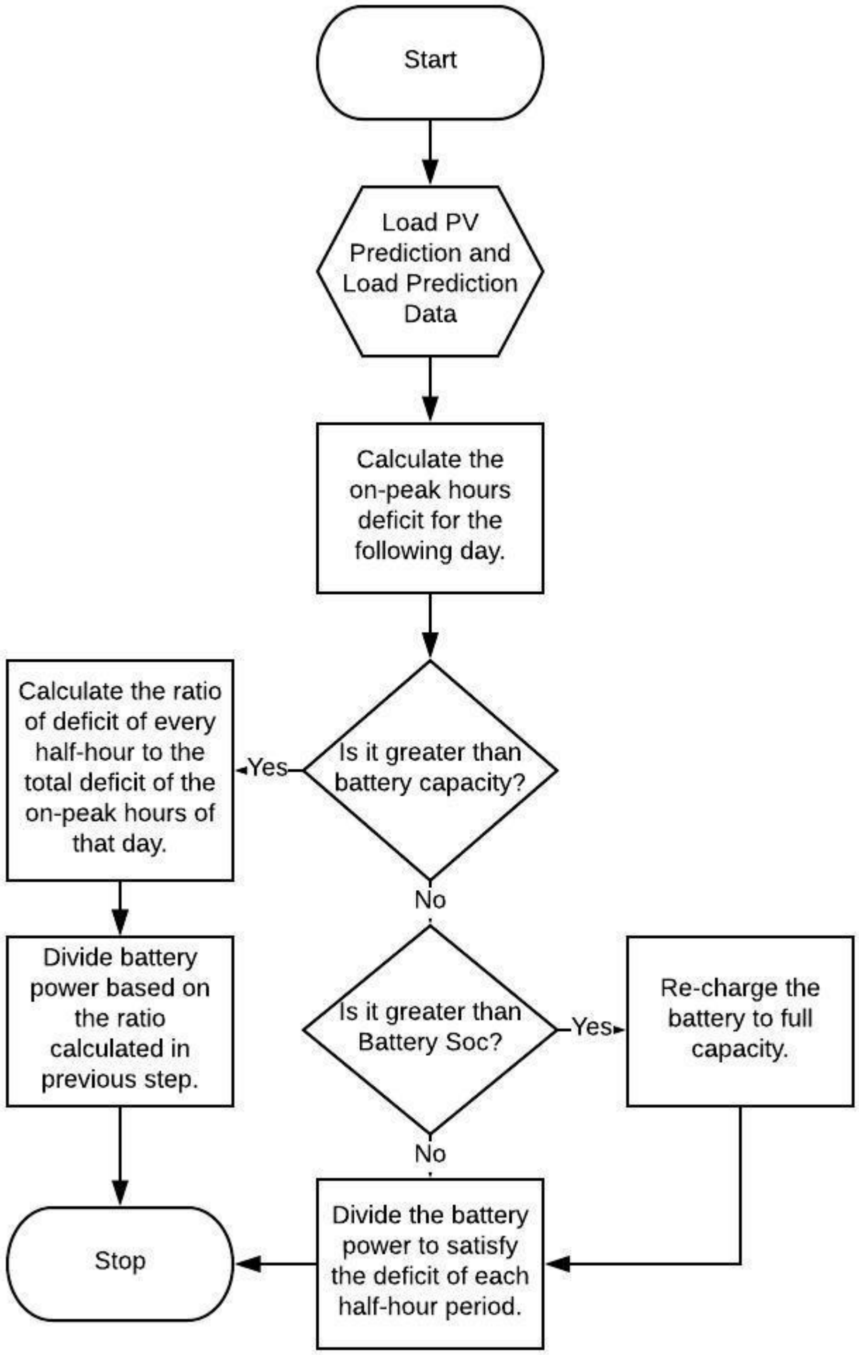

Start
Load PV Prediction and Load Prediction Data
Calculate the on-peak hours deficit for the following day.
Is it greater than battery capacity?
Yes
No
Calculate the ratio of deficit of every half-hour to the total deficit of the on-peak hours of that day.
Divide battery power based on the ratio calculated in previous step.
Is it greater than Battery Soc?
Yes
No
Re-charge the battery to full capacity.
Divide the battery power to satisfy the deficit of each half-hour period.
Stop

Figure 5.2 Flowchart of the battery control procedure.

The charging of the battery every night is also controlled by this algorithm. For example, if the entire next day battery power requirement could be satisfied with 40% state of charge (SoC) of the battery, and the amount of charge left in the battery at the end of the current day is 60%, the battery will not be charged at the end of the day. Alternatively, if the amount of charge left is 60% and the following daily requirement is 50% (battery at 20% SoC is considered fully dis-charged), the battery is then charged to its full capacity (or 90% SoC).

This is especially useful when the occupants go on a vacation and during weekday afternoons of a peak summer month. The house in such cases, will most likely require minimal power which could be supplied by the PV, the remaining charge on the battery and grid (without inducing a huge Demand Charge) respectively. The built-in algorithm on the other hand, charges the battery using the excess PV during the day time as well. The charging of the battery does not happen unless necessary and as a result, prolonging the battery life, by avoiding the unnecessary charging/discharging cycles.

The following section of this chapter presents the experimental results obtained.

5.4 Results

This section is further divided into two halves. The first half is the set of graphs, namely Figure 5.3 through Figure 5.23, pertaining to the actual parameters experienced by the hardware on the test days. The second half of this section presents the results/deviations of the theoretical calculations from the actual observed values at the test site. Finally, a tabulation of all the important numbers that the economic analysis would be based.

5.4.1 PV Data Extraction

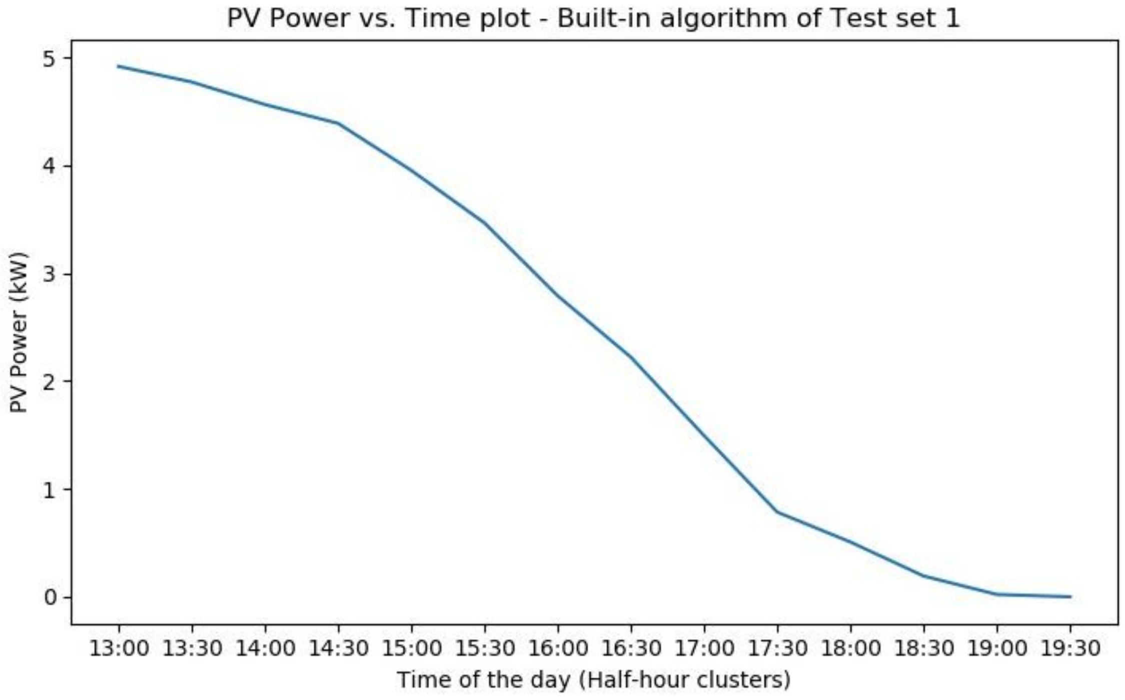


Figure 5.3 Plot of the PV power variation over time on the built-in algorithm simulation day – Test set 1.

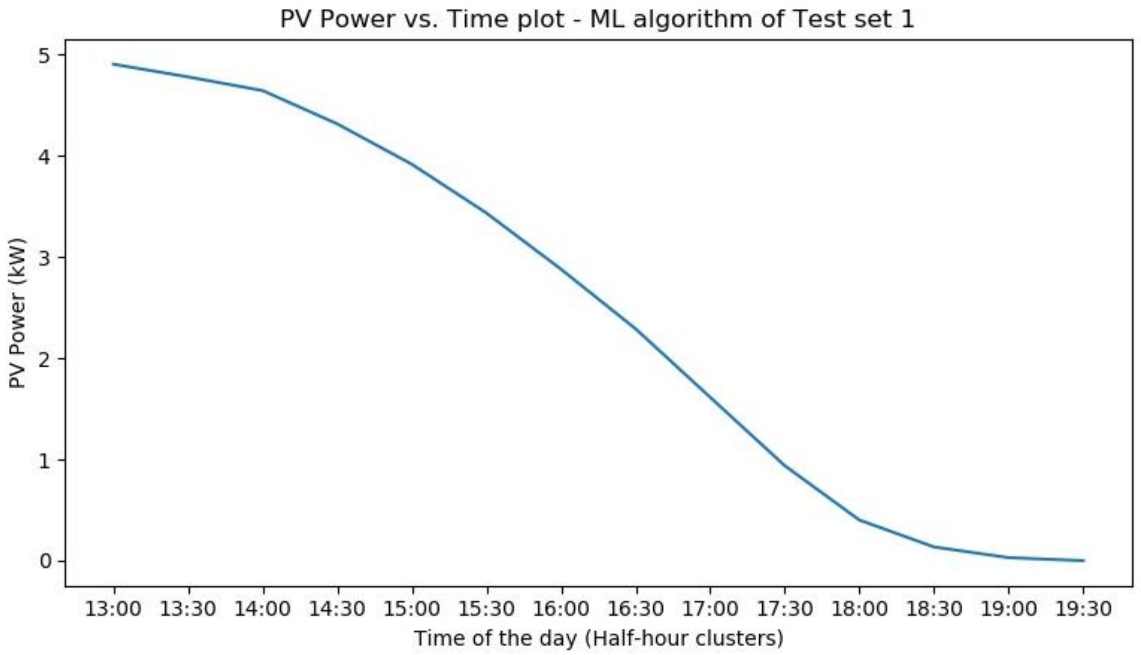


Figure 5.4 Plot of the PV power variation over time on the ML algorithm simulation day – Test set 1.

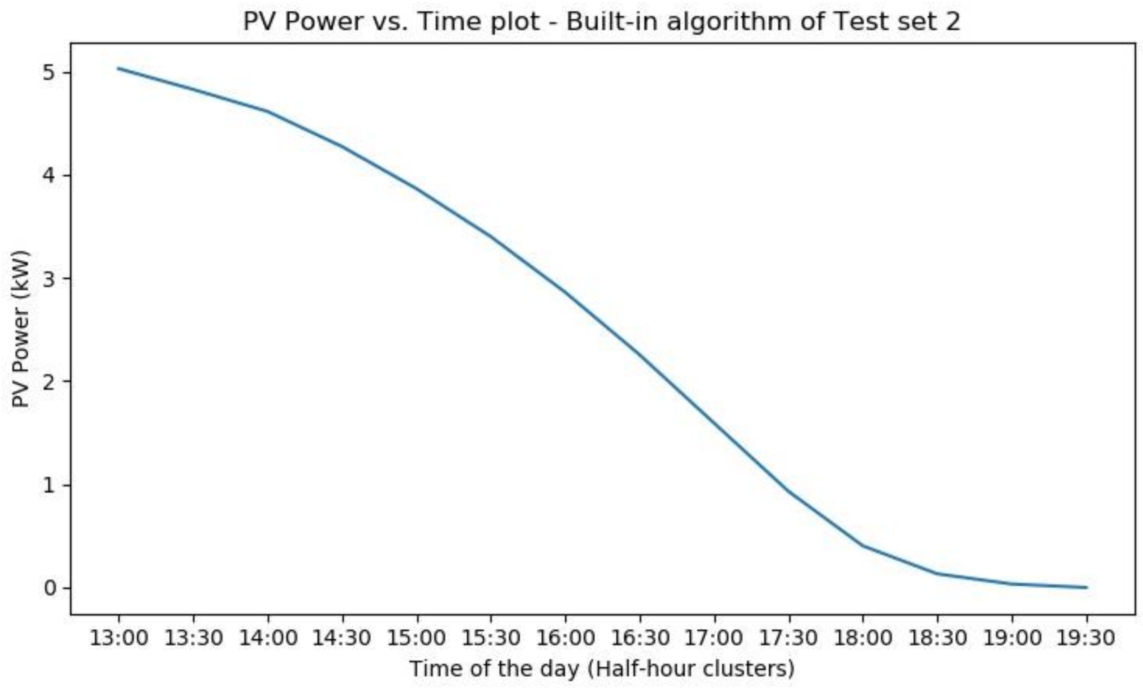


Figure 5.5 Plot of the PV power variation over time on the built-in algorithm simulation day – Test set 2.

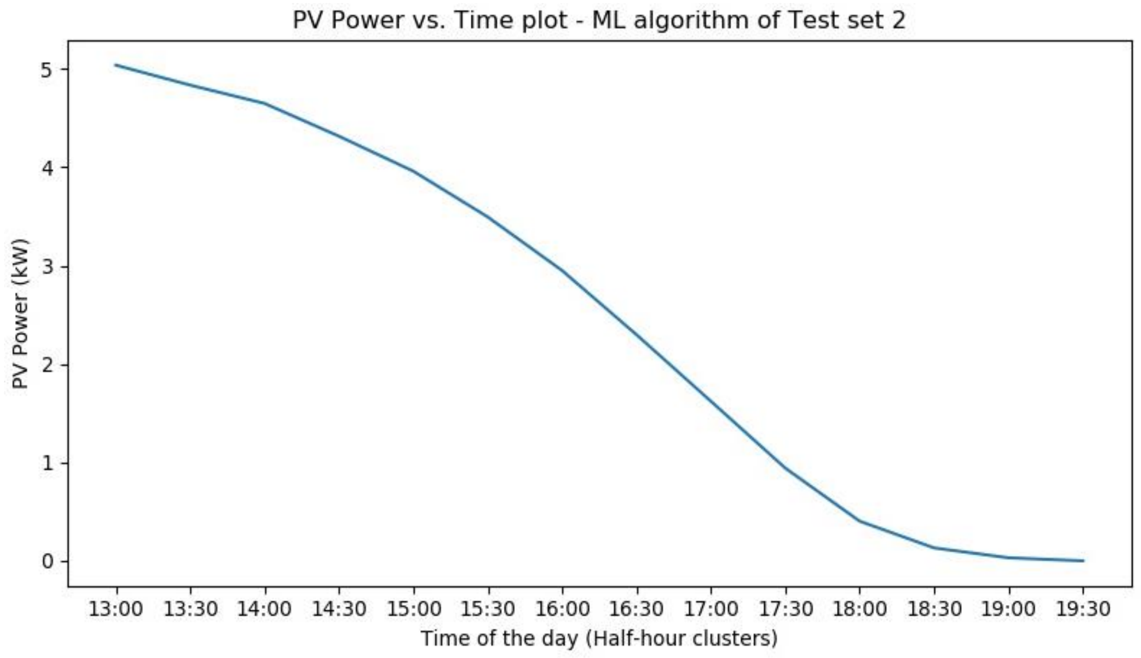


Figure 5.6 Plot of the PV power variation over time on the ML algorithm simulation day – Test set 2.

5.4.2 Site Demand Extraction

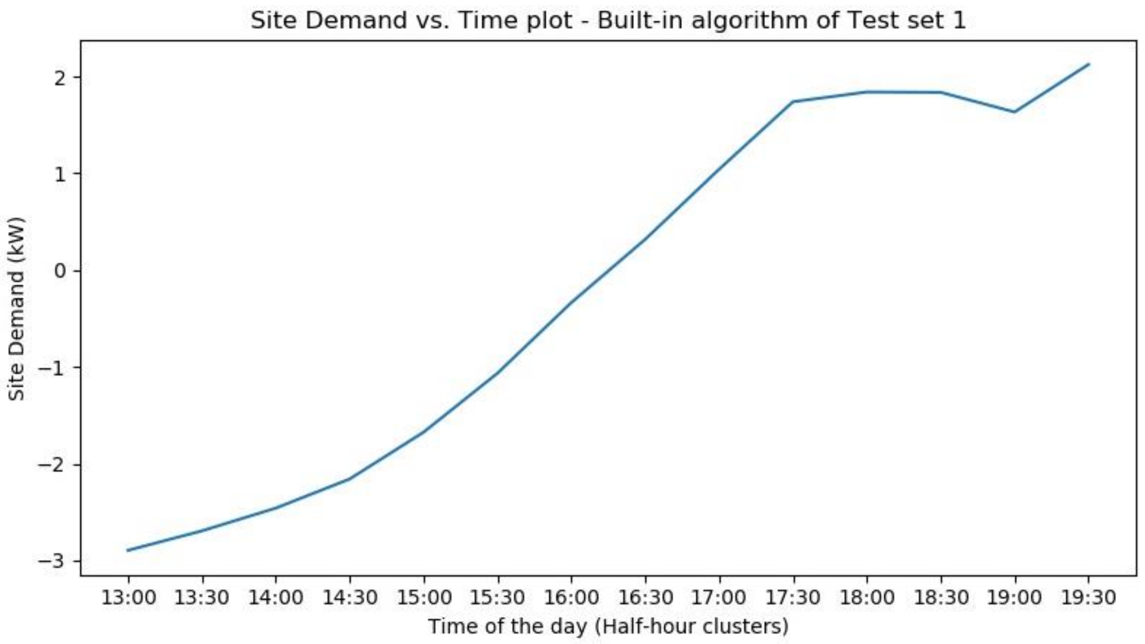


Figure 5.7 Plot of the site demand variation over time for the built-in algorithm – Test set 1.

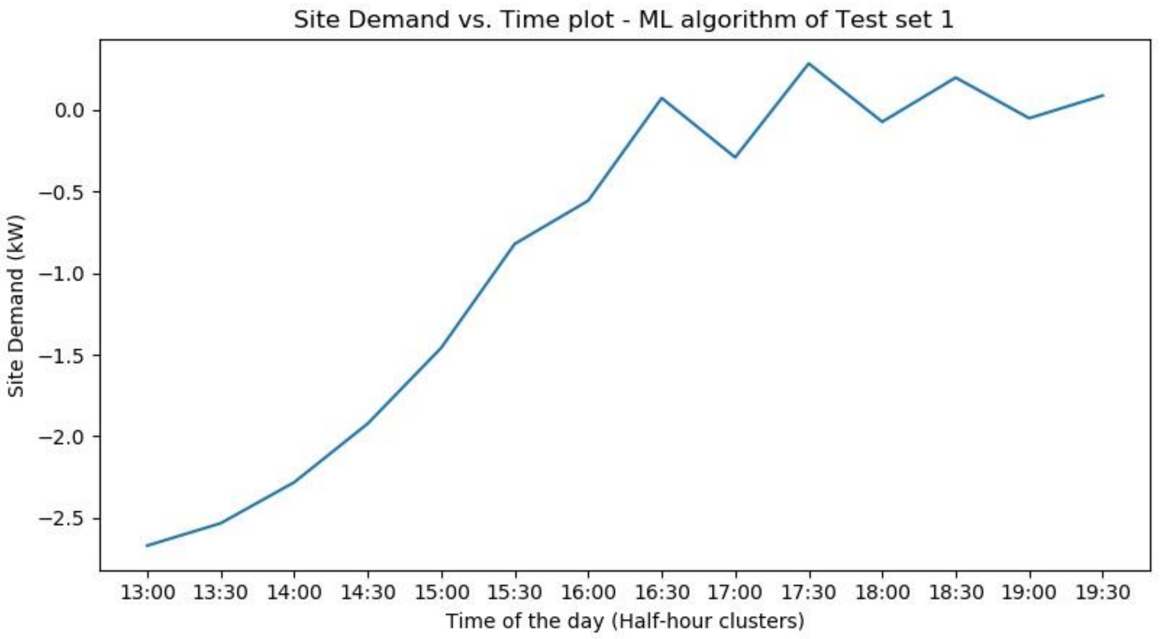


Figure 5.8 Plot of the site demand variation over time for the ML algorithm – Test set 1.

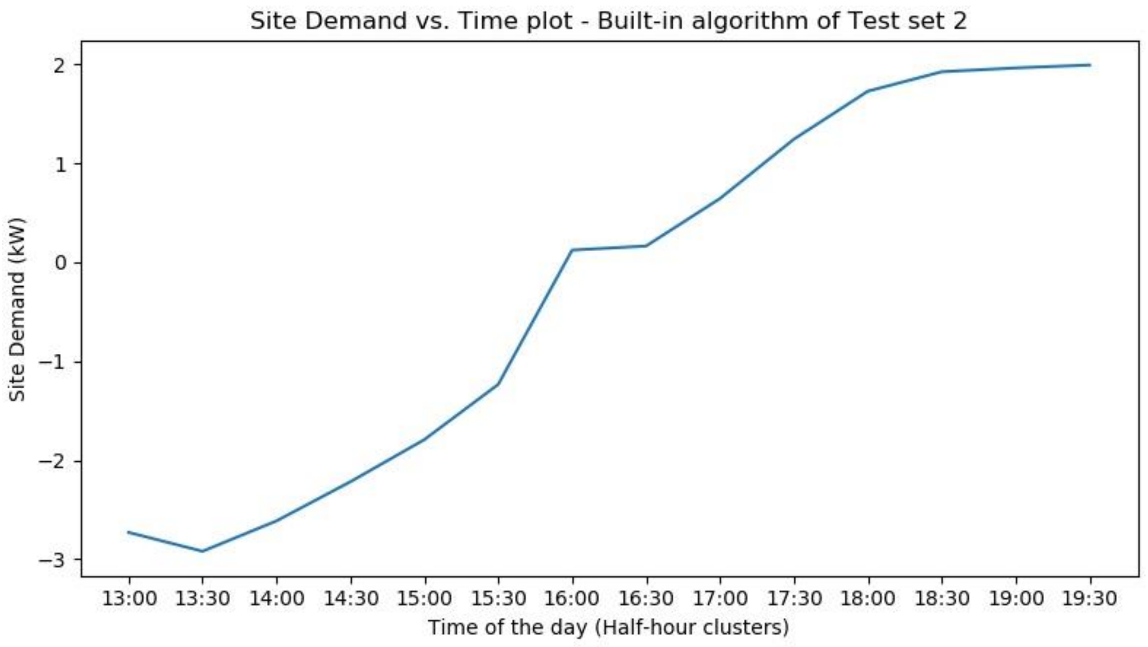


Figure 5.9 Plot of the site demand variation over time for the built-in algorithm – Test set 2.

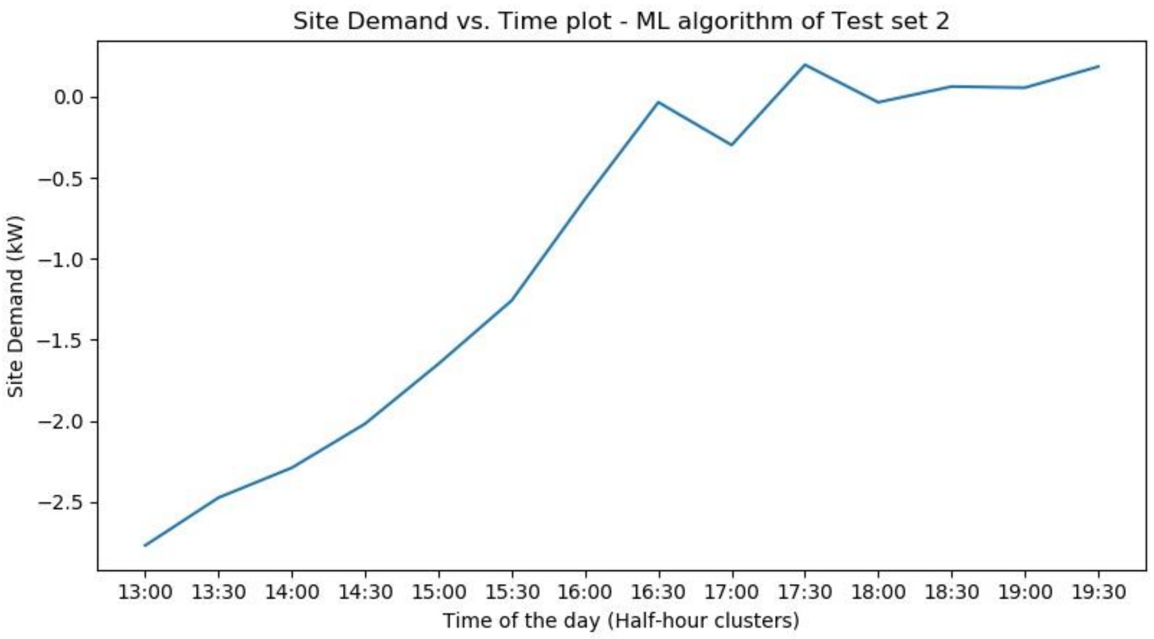


Figure 5.10 Plot of the site demand variation over time for the ML algorithm – Test set 2.

5.4.3 Load Power Extraction

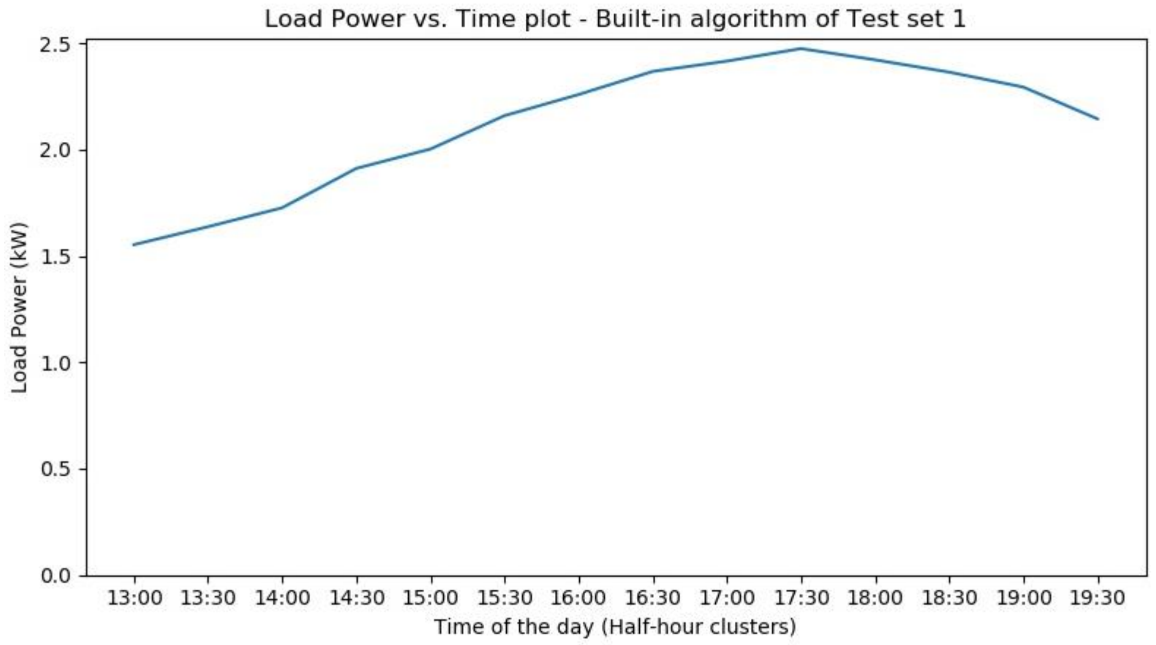


Figure 5.11 Plot of the load power variation over time for the built-in algorithm – Test set 1.

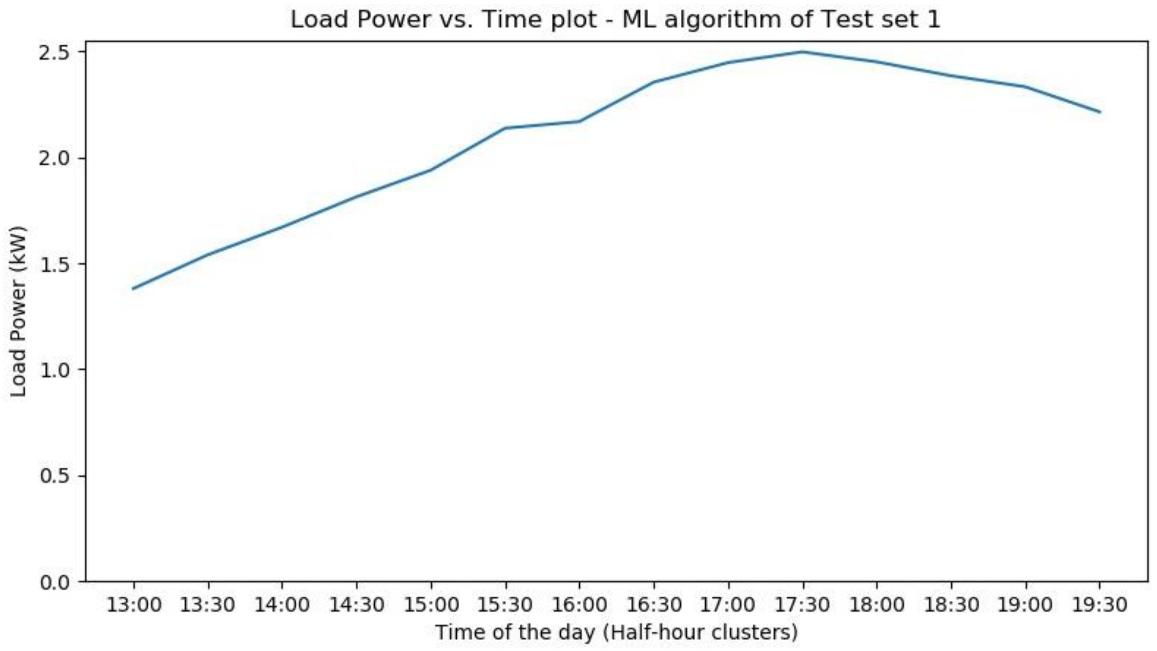


Figure 5.12 Plot of the load power variation over time for the ML algorithm – Test set 1.

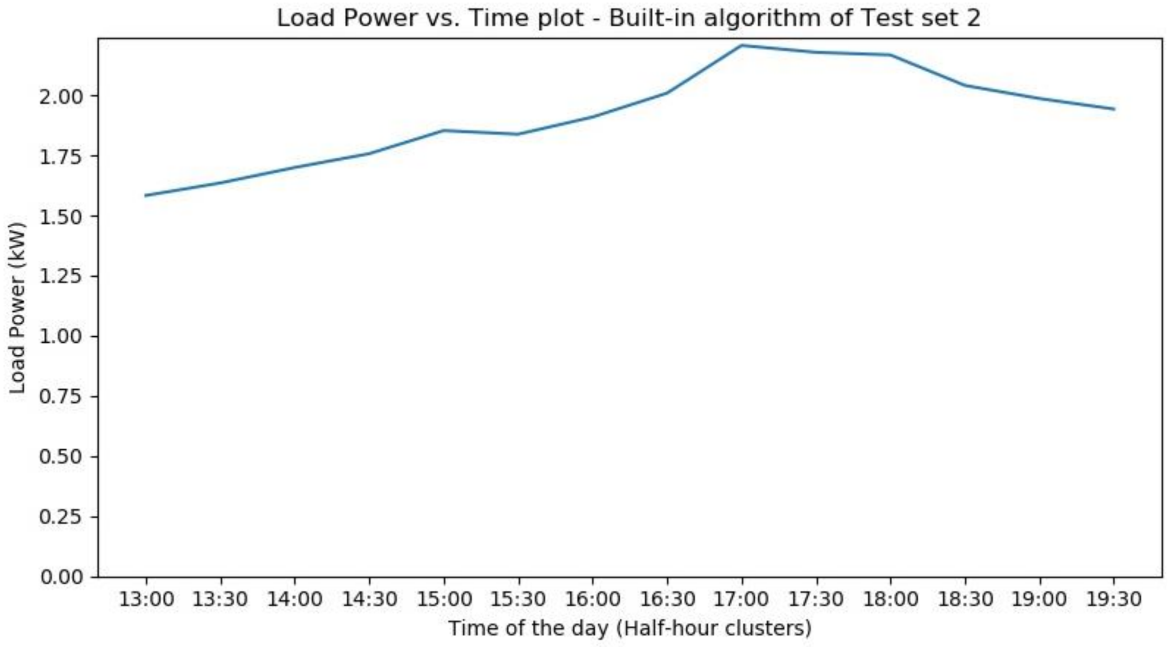


Figure 5.13 Plot of the load power variation over time for the built-in algorithm – Test set 2.

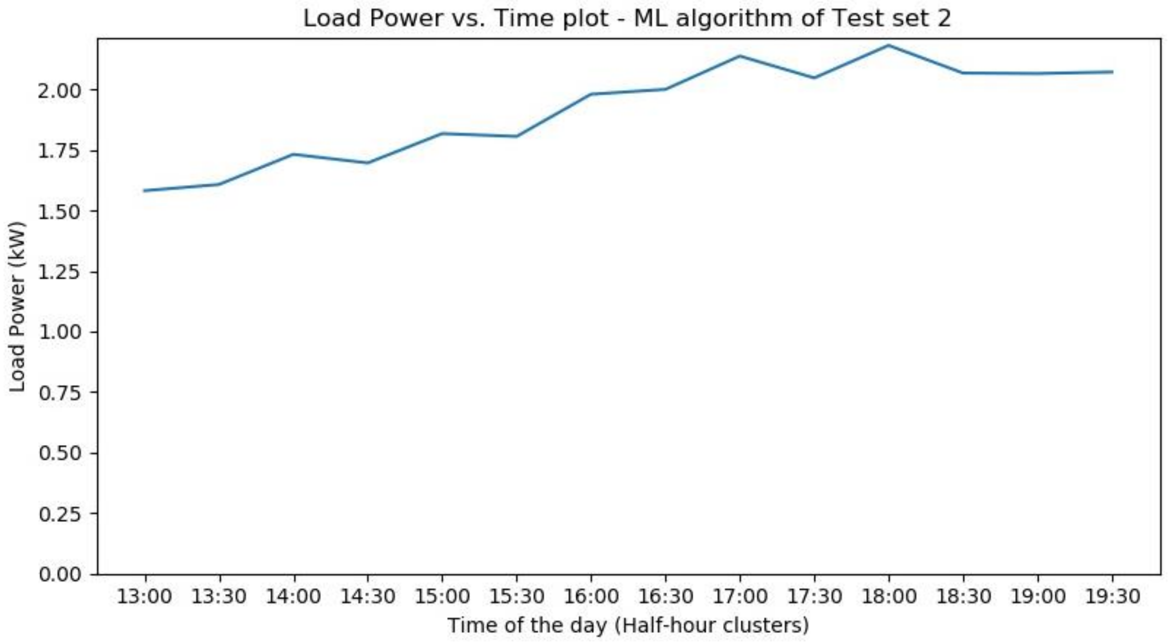


Figure 5.14 Plot of the load power variation over time for the ML algorithm – Test set 2.

5.4.4 Battery State of Charge Extraction

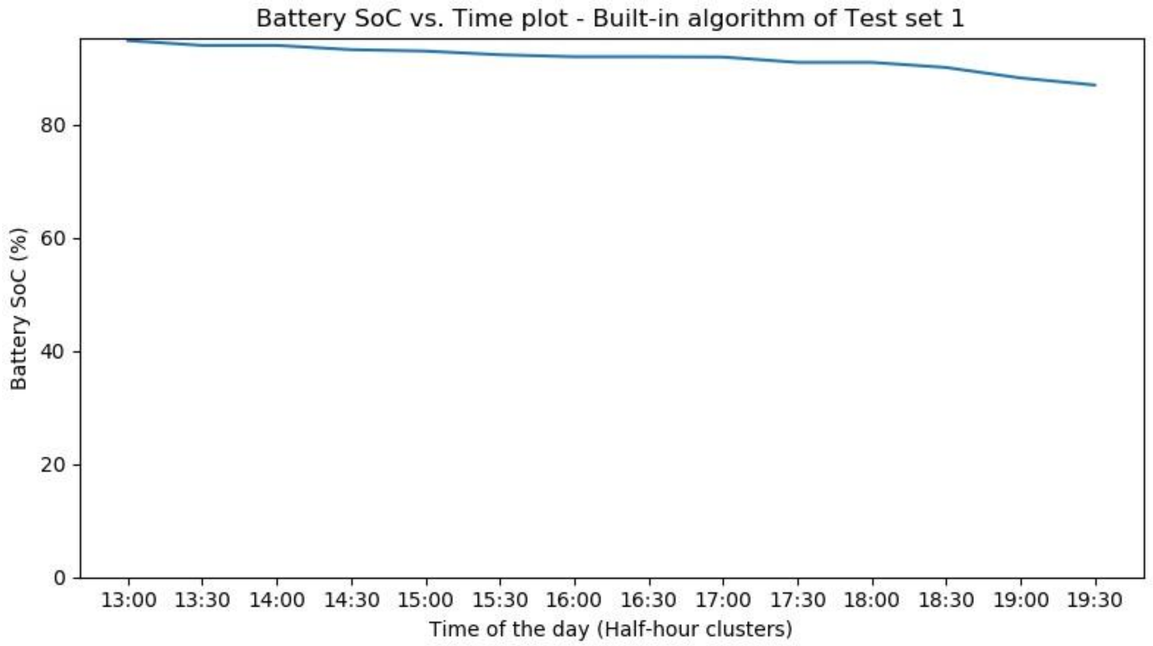


Figure 5.15 Plot of the battery SoC variation over time for the built-in algorithm – Test set 1.

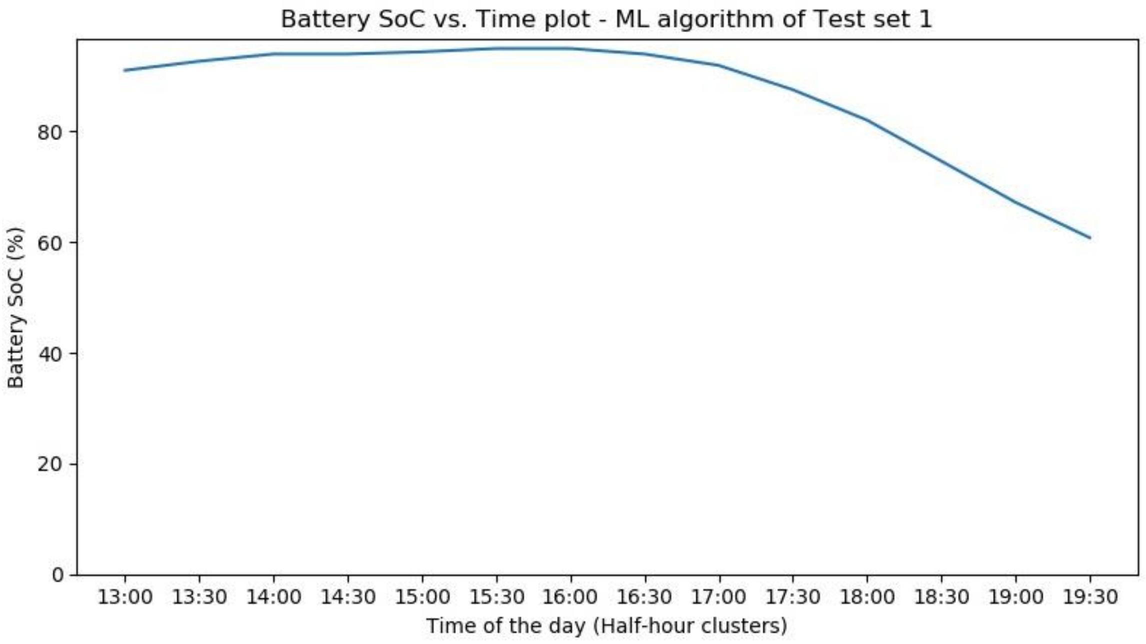


Figure 5.16 Plot of the battery SoC variation over time for the ML algorithm – Test set 1.

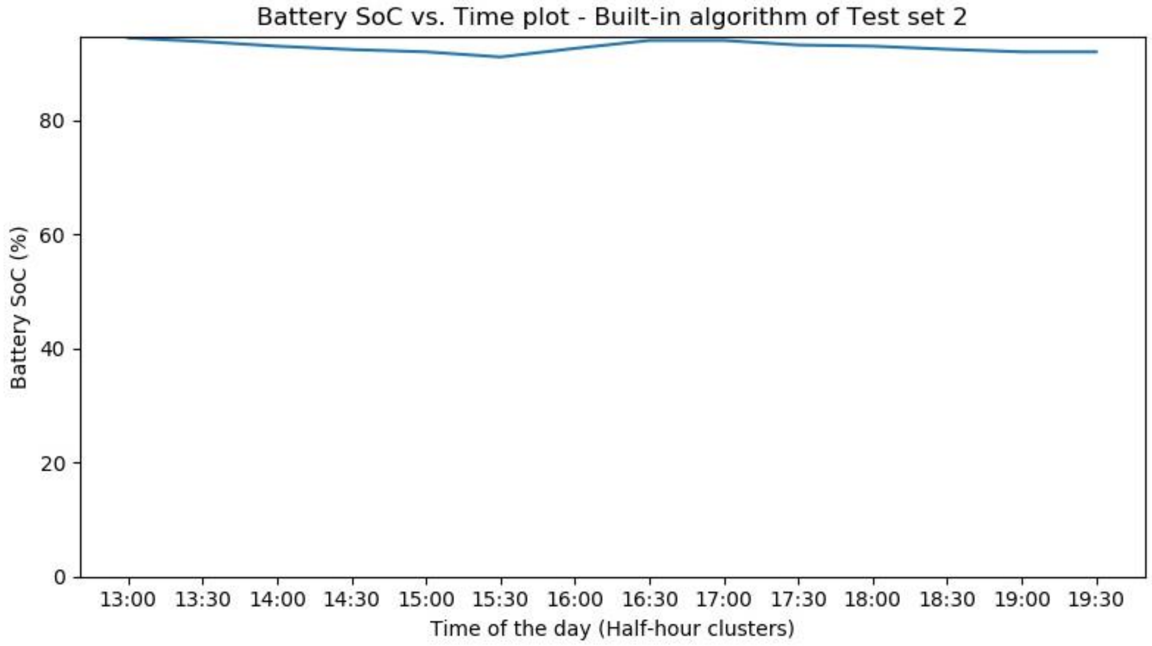


Figure 5.17 Plot of the battery SoC variation over time for the built-in algorithm – Test set 2.

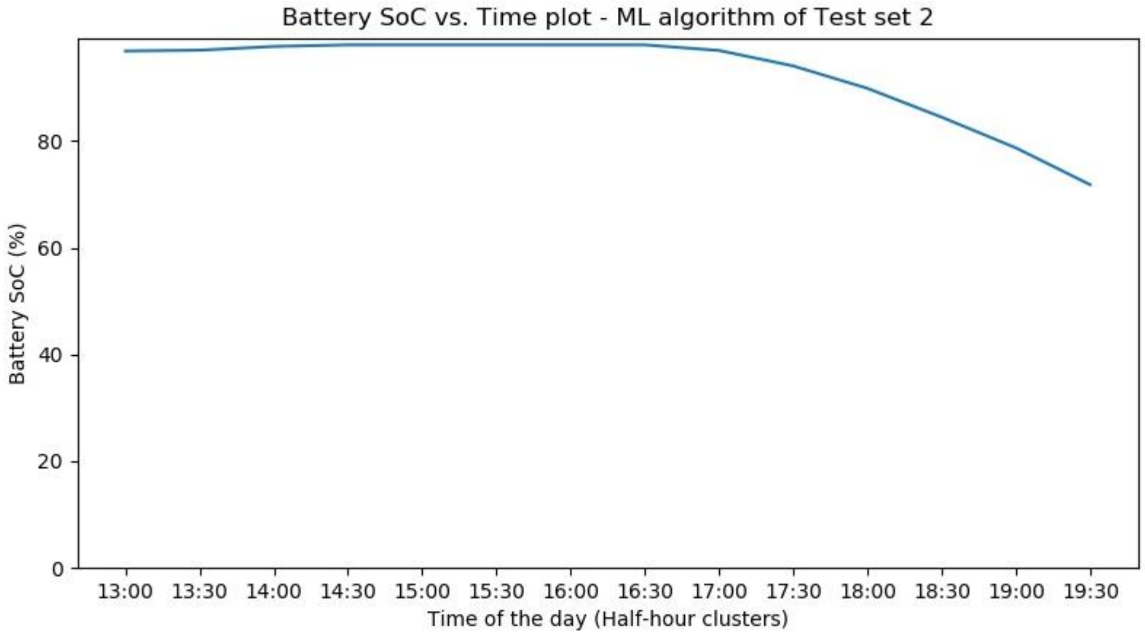


Figure 5.18 Plot of the battery SoC variation over time for the ML algorithm – Test set 2.

5.4.5 Battery Power Extraction

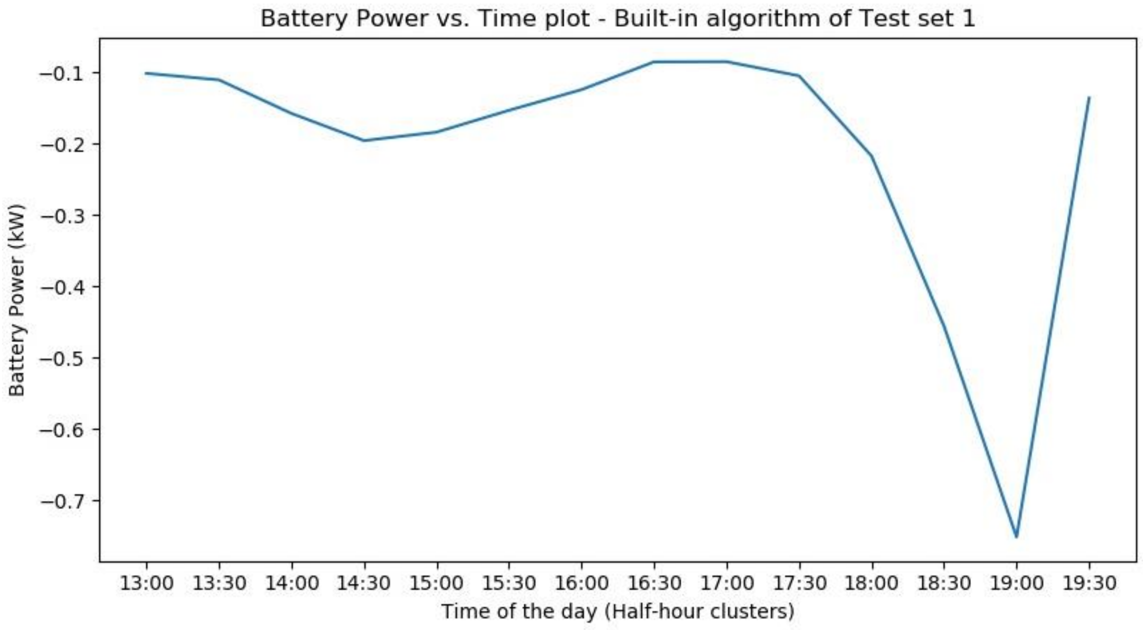


Figure 5.19 Plot of the battery power variation over time for the built-in algorithm – Test set 1.

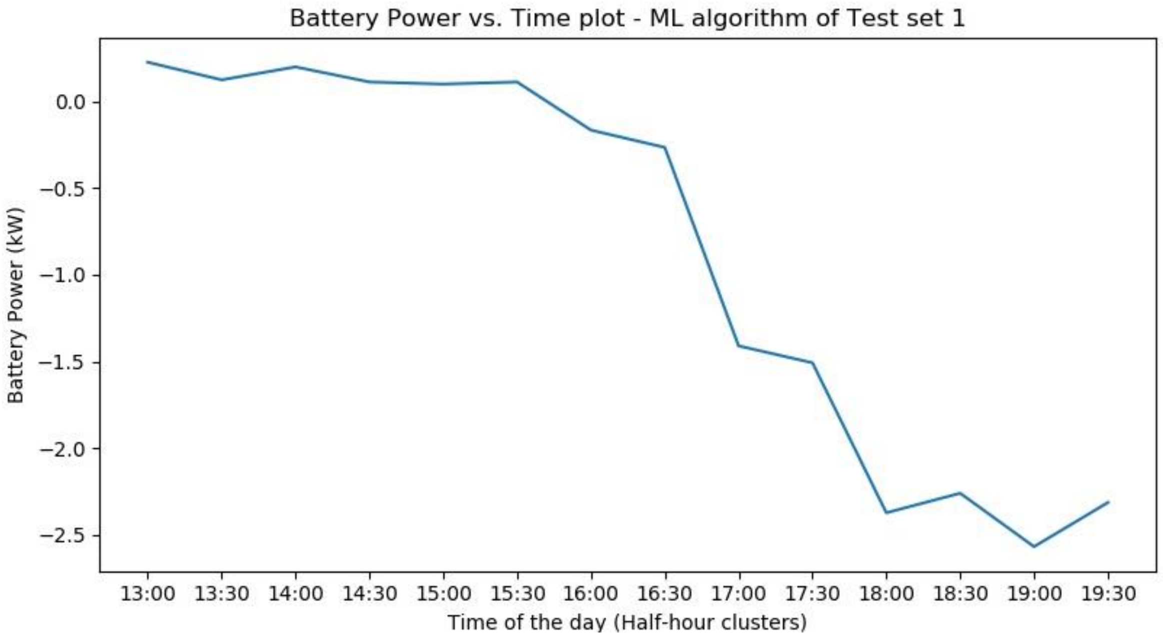


Figure 5.20 Plot of the battery power variation over time for the ML algorithm – Test set 1.

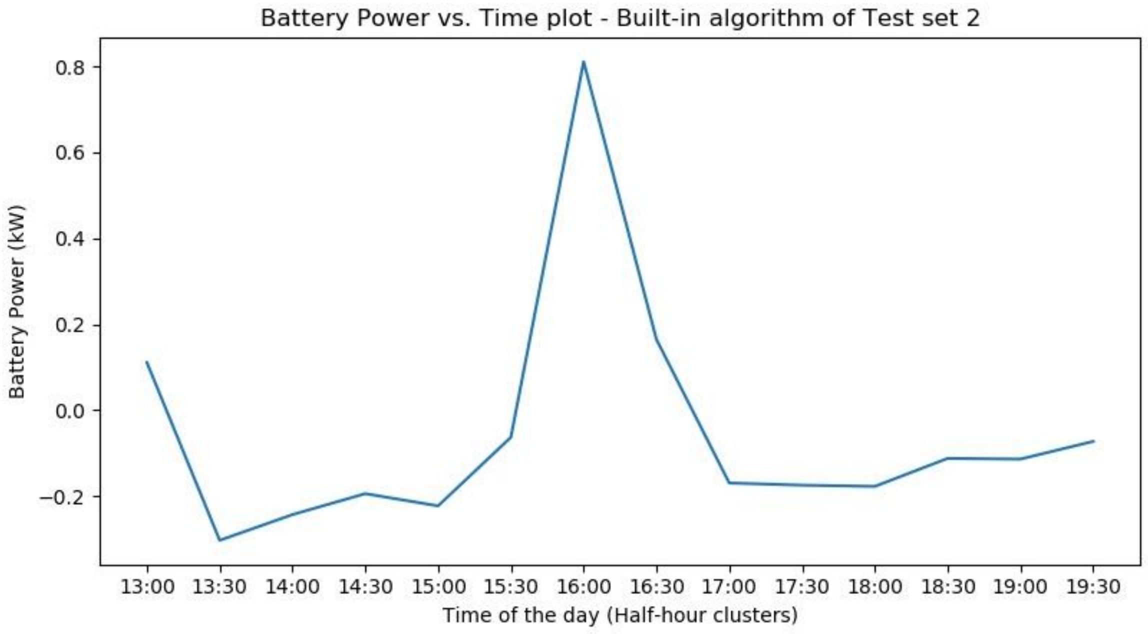


Figure 5.21 Plot of the battery power variation over time for the built-in algorithm – Test set 2.

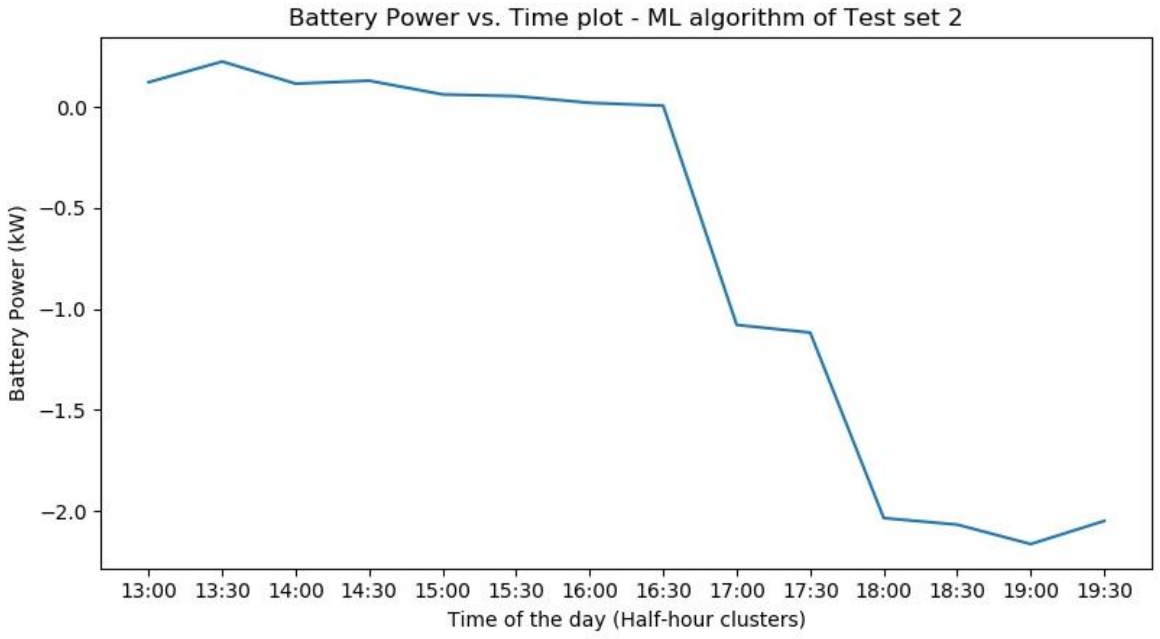
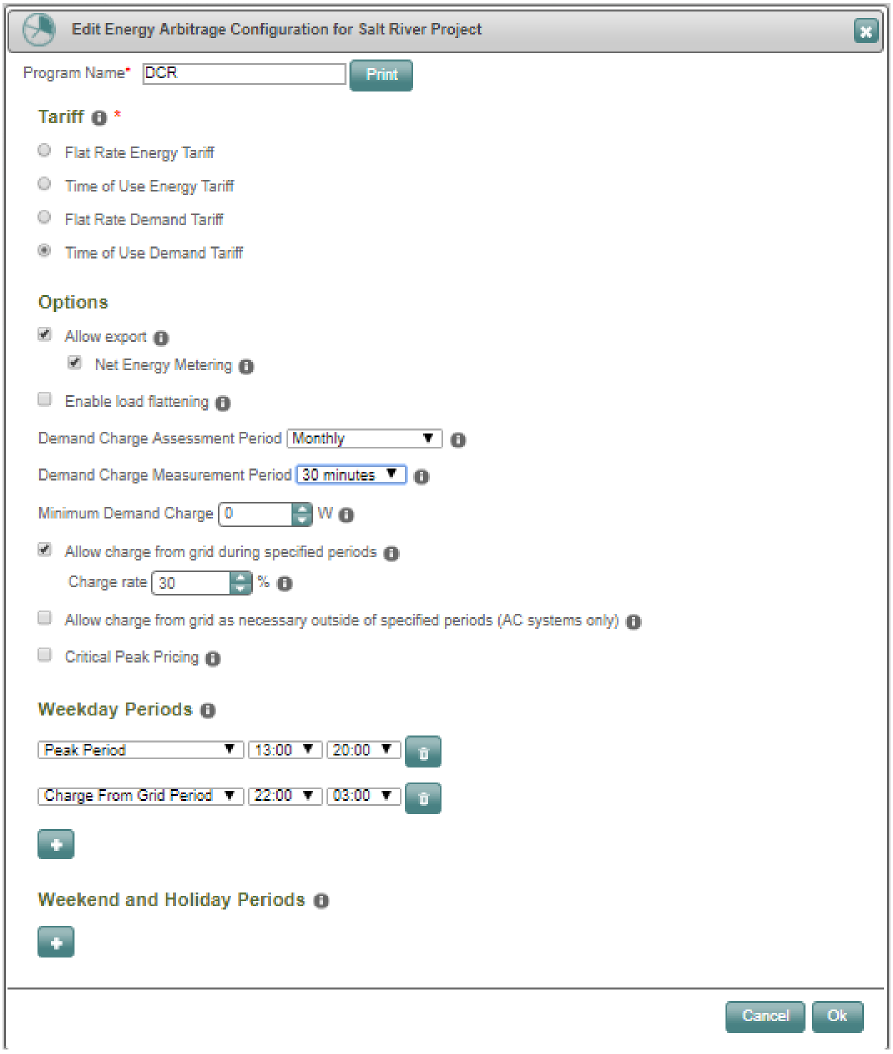


Figure 5.22 Plot of the battery power variation over time for the ML algorithm – Test set 2.

Since the built-in algorithm is an algorithm built by Sunverge for commercial purpose, the exact working logic governing this algorithm is not very well known. Most of the inferences gathered are based on experimentation results and conversations with the Sunverge representatives in form of online meetings through the electric utility. The settings used to extract the above results pertaining to the built-in algorithm are shown in Figure 5.23. The reason for adopting this setting is, the works of the ASU students prior to this research on the Sunverge system considered this setting to be the default setting, i.e. Sunverge's setting. All the results extracted were based on this setting. Thus, the same has been followed for this research as well.


Edit Energy Arbitrage Configuration for Salt River Project

Program Name* DCR

Tariff *

- ○ Flat Rate Energy Tariff
- ○ Time of Use Energy Tariff
- ○ Flat Rate Demand Tariff
- ● Time of Use Demand Tariff

Options

- ✓ Allow export
  - ✓ Net Energy Metering
- ☐ Enable load flattening

Demand Charge Assessment Period: Monthly

Demand Charge Measurement Period: 30 minutes

Minimum Demand Charge: 0 W

- ✓ Allow charge from grid during specified periods
  - Charge rate: 30 %
- ☐ Allow charge from grid as necessary outside of specified periods (AC systems only)
- ☐ Critical Peak Pricing

Weekday Periods

Peak Period	13:00	20:00
Charge From Grid Period	22:00	03:00

Weekend and Holiday Periods

Figure 5.23 The Sunverge's default setting.

5.4.6 PV Prediction Results

For convenience, the hourly average of the PV power has been used in Table 5.2 through Table 5.5. This is so as to remain fair to the PVWATTS Calculator which provides the predictions as hourly averages.

Table 5. 2 Tabulation comparing the actual PV values with the predicted values pertaining to the day 1 of simulations (Built-in algorithm).

Interval	Actual PV (kW)	NREL prediction (kW)	Deviation (kW)
13:00	4.84	4.74	0.1
14:00	4.47	4.18	0.29
15:00	3.7	3.43	0.27
16:00	2.5	2.31	0.19
17:00	1.14	1.05	0.09
18:00	0.35	0.15	0.2
19:00	0.01	0	0.01

Table 5. 3 Tabulation comparing the actual PV values with the predicted values pertaining to the day 2 of simulations (ML algorithm).

Interval	Actual PV (kW)	NREL prediction (kW)	Deviation (kW)
13:00	4.84	4.74	0.1
14:00	4.48	4.18	0.3
15:00	3.68	3.43	0.25
16:00	2.58	2.31	0.27
17:00	1.27	1.05	0.22
18:00	0.26	0.15	0.11
19:00	0.01	0	0.01

Table 5. 4 Tabulation comparing the actual PV values with the predicted values pertaining to the day 3 of simulations (Built-in algorithm).

Interval	Actual PV (kW)	NREL prediction (kW)	Deviation (kW)
13:00	4.92	4.49	0.43
14:00	4.44	4.23	0.21
15:00	3.63	3.25	0.38
16:00	2.55	2.32	0.23
17:00	1.26	1.03	0.23
18:00	0.26	0.16	0.1
19:00	0.01	0	0.01

Table 5. 5 Tabulation comparing the actual PV values with the predicted values pertaining to the day 4 of simulations (ML algorithm).

Interval	Actual PV (kW)	NREL prediction (kW)	Deviation (kW)
13:00	4.93	4.49	0.44
14:00	4.48	4.23	0.25
15:00	3.72	3.25	0.47
16:00	2.62	2.32	0.3
17:00	1.28	1.03	0.25
18:00	0.26	0.16	0.1
19:00	0.01	0	0.01

From Tables 5.2-5.5, it is evident that the PVWATTS Calculator is efficient in predicting the PV power availability with minimal error. It must also be noted that certain parameters provided as an input to this web application were a bit on the conservative side. The deviations observed were positive deviations, i.e. the available

PV was always greater than the predicted value. This is acceptable because, if excess PV is available, the excess energy is merely exported into the grid and the customer ends up earning money for this. A problem would arise if the prediction ends up being greater than the actual available PV power. In such a case, the system would draw power from the grid possible inducing a higher Demand Charge and energy bill. This prediction (being conservative) suits the requirement, of not drawing power from the grid, perfectly. Figures 5.24 – 5.27 shows the actual and predicted PV power values.

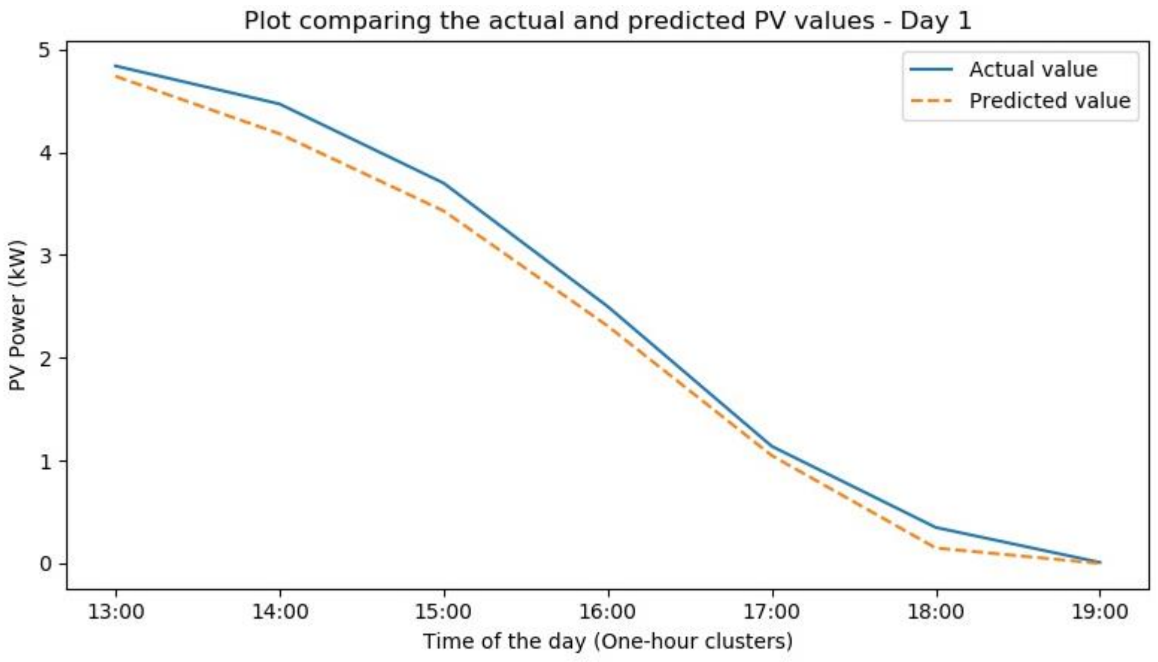


Figure 5.24 Plot comparing the actual and predicted PV power values for day 1.

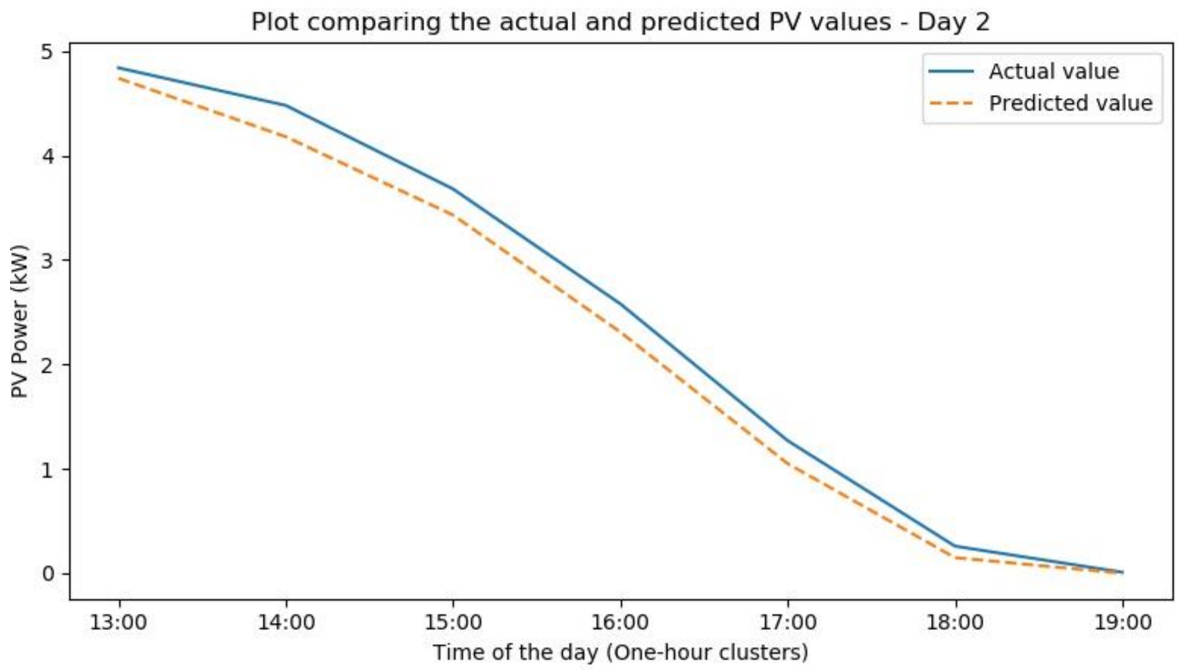


Figure 5.25 Plot comparing the actual and predicted PV power values for day 2.

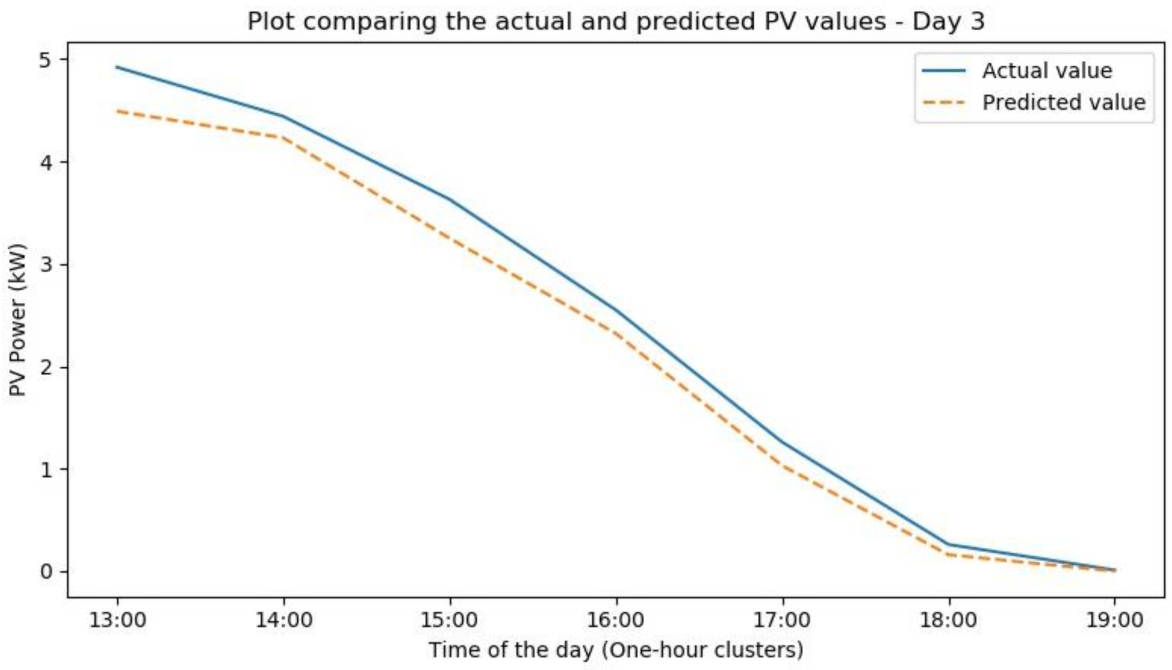


Figure 5.26 Plot comparing the actual and predicted PV power values for day 3.

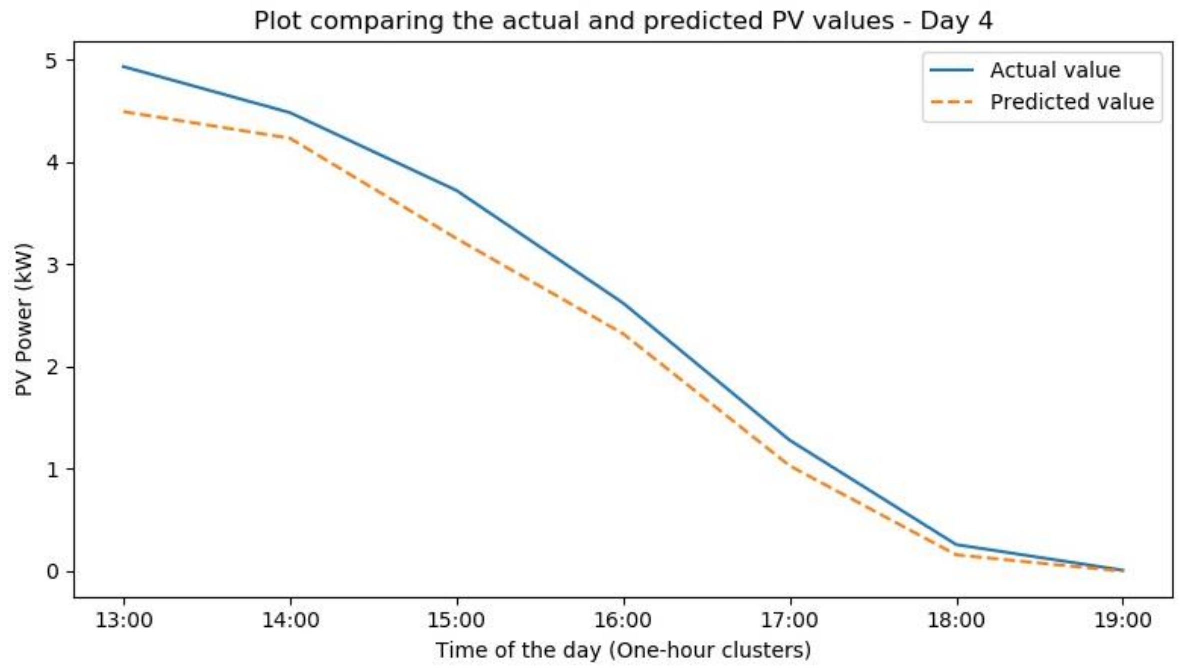


Figure 5.27 Plot comparing the actual and predicted PV power values for day 4.

The Tables 5.6 and 5.7 contain data pertaining to the prediction of the battery power required at every point in time during the on-peak hours of the days of simulation. Since the utility calculates the Demand Charge based on average load values over half-hour intervals, the tables contain the prediction for every half-hour interval of the on-peak hours. The solar prediction and the load prediction are the input to this algorithm. The difference between these two powers is the deficit that needs to be made up by the power supplied by the battery. In case these predictions are 100% accurate, then the power drawn from the grid would be zero. But, as seen earlier, both the predictions do have some error and thus, the power drawn from the grid at the end of this scenario would be a few watts.

Note: A negative sign in the deficit value indicates power of the indicated magnitude being exported into the grid.

Table 5. 6 Tabulation of the battery power requirement over every half-hour interval during the on peak hours pertaining to the 1st set of simulations.

Interval	Predicted load – Predicted PV (kW)	Battery power requirement (kW)	Notes
13:00	-3.14	0	Excess PV exported to grid. Battery on standby.
13:30	-2.94	0	Excess PV exported to grid. Battery on standby.
14:00	-2.38	0	Excess PV exported to grid. Battery on standby.
14:30	-2.18	0	Excess PV exported to grid. Battery on standby.
15:00	-1.13	0	Excess PV exported to grid. Battery on standby.
15:30	-0.84	0	Excess PV exported to grid. Battery on standby.
16:00	0.289	0.289	Battery discharges at a constant rate.
16:30	0.289	0.289	Battery discharges at a constant rate.
17:00	1.55	1.55	Battery discharges at a constant rate.
17:30	1.55	1.55	Battery discharges at a constant rate.
18:00	2.45	2.45	Battery discharges at a constant rate.
18:30	2.25	2.25	Battery discharges at a constant rate.
19:00	2.6	2.6	Battery discharges at a constant rate.
19:30	2.4	2.4	Battery discharges at a constant rate.

Table 5. 7 Tabulation of the battery power requirement over every half-hour interval during the on peak hours pertaining to the 2nd set of simulations.

Interval	Predicted load – Predicted PV (kW)	Battery power requirement (kW)	Notes
13:00	-2.89	0	Excess PV exported to grid. Battery on standby.
13:30	-2.69	0	Excess PV exported to grid. Battery on standby.
14:00	-2.43	0	Excess PV exported to grid. Battery on standby.
14:30	-2.33	0	Excess PV exported to grid. Battery on standby.
15:00	-1.35	0	Excess PV exported to grid. Battery on standby.
15:30	-1.05	0	Excess PV exported to grid. Battery on standby.
16:00	-0.125	0	Excess PV exported to grid. Battery on standby.
16:30	-0.125	0	Excess PV exported to grid. Battery on standby.
17:00	1.17	1.17	Battery discharges at a constant rate.
17:30	1.17	1.17	Battery discharges at a constant rate.
18:00	2.04	2.04	Battery discharges at a constant rate.
18:30	2.04	2.04	Battery discharges at a constant rate.
19:00	2.2	2.2	Battery discharges at a constant rate.
19:30	2.1	2.1	Battery discharges at a constant rate.

The Figures 5.28 and 5.29 compare the predicted and the actual difference between the load and the PV powers at any interval.

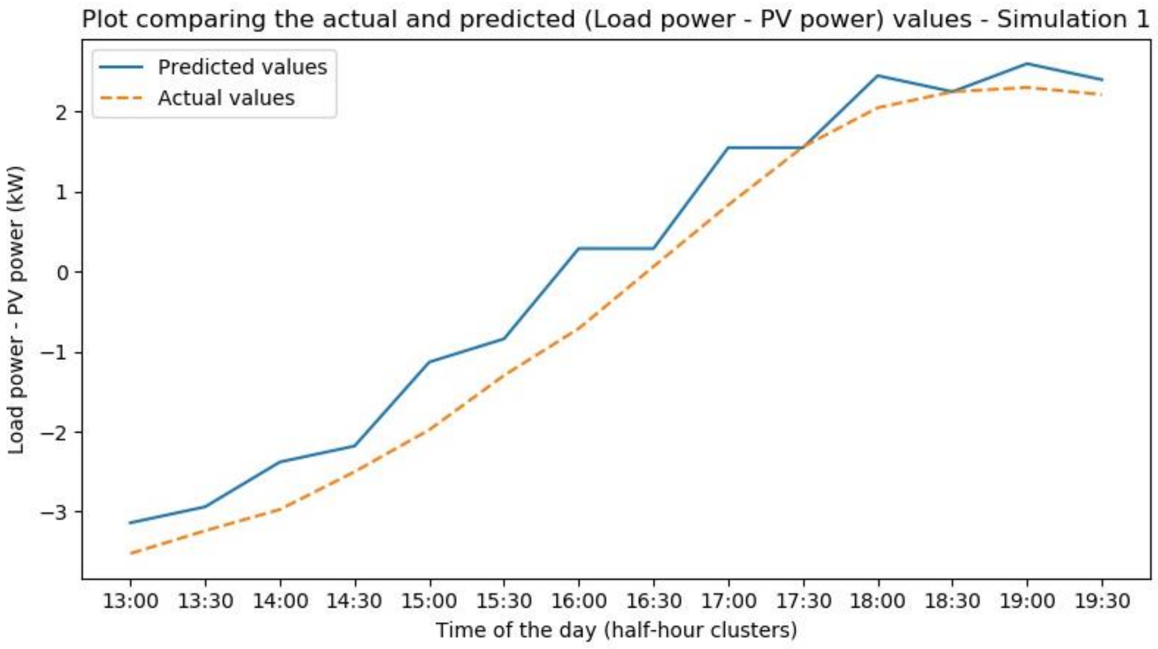


Figure 5.28 Plot comparing the actual and predicted (Load power- PV power) values for simulation 1.

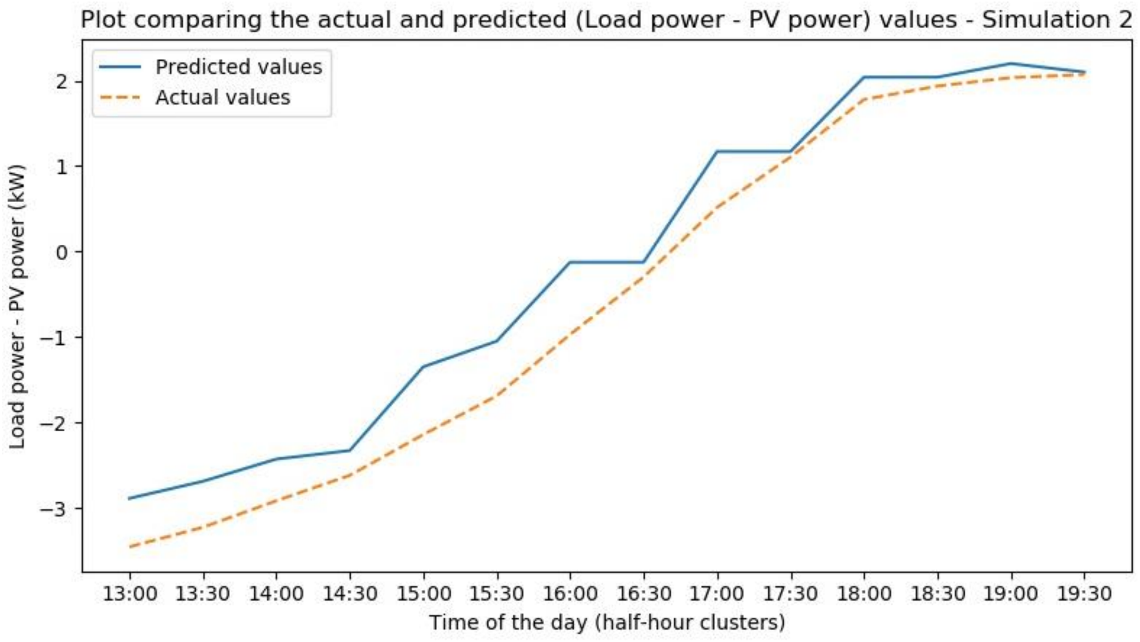


Figure 5.29 Plot comparing the actual and predicted (Load power- PV power) values for simulation 2.

It can be observed from the graphs that the predicted values are always a bit conservative compared to the actual values.

During the afternoon when there is excess solar power, the prediction is conservative and thus predicts the value of the available power to export to the grid to be lesser than the actual available value. This is beneficial because, if the algorithm had predicted more availability than available, then that difference between the actual and predicted values would be drawn from the battery to be supplied to the grid. This happens since the Sunverge system's internal control manages everything in terms of specific numbers, meaning, for every interval, the power that must be drawn/supplied from/to the grid needs to be specified. The system tries to stick to this number and thus in case of over optimism, ends up drawing the deficit power from the battery and exporting it to the grid.

Similarly, the ML algorithm is conservative in the evenings by predicting the battery demand to be slightly more than the actual demand. In this case, similar to earlier, the system tries to discharge the specified amount of power from the battery. In case the demand is lesser than the predicted value, the excess battery discharge is exported into the grid thus earning the customer credits. On the other hand, if the prediction had been lesser than the actual value, the battery is discharged at the predicted value and the deficit is made up by power drawn from the grid. This costs the customer money and there is a chance of a higher Demand Charge. This is what happens in the case of the built-in algorithm leading to its poor performance. But again, since there is also a constraint to not discharge the battery unless necessary to decelerate cycling aging, the predicted value cannot be too conservative i.e. way higher than the actual requirement. It must be as close to the actual value as possible.

Another thing to note is the smoothness of both the curves. The curve representing the actual value is smooth while the curve representing the predicted value rises and falls in steps. This is because of the initial treatment of this problem as a "Classification" problem and the grouping of the values into 200 W clusters. A grouping of smaller magnitude or treating the problem as a "Regression" problem would increase the smoothness.

The Tables 5.8 and 5.9 compile the important and conclusive results of this chapter that are a measure of working efficiency of both the algorithms. Since these parameters will be used in the Economic Analysis, they directly affect the investment return period of the customer.

Table 5. 8 Tabulation of the maximum site demand of the on-peak hours on the test days.

Test	Built-in Algorithm (kW)	ML Algorithm (kW)	Difference (kW)
1	2.125	0.283	1.842
2	1.991	0.197	1.793

Table 5. 9 Tabulation of the battery SoC at the end of on-peak hours for the days of ML algorithm simulations.

Test Day	Battery SoC at day end	Battery power requirement of the following day in terms of SoC	Charging required tonight?
1	60.8%	39.07%	NO
2	71.8%	38.91%	NO

5.5 Inference

This chapter shows with evidence the higher efficiency and accuracy of the ML algorithm when compared to the built-in algorithm and the algorithm developed earlier by the students of ASU presented in [4]. Also, the assumptions made at the beginning of this research have proven to be valid. Especially the conversion of the problem into a “Classification” problem rather than the more natural “Regression” problem has been justified here.

The Tables 5.8 and 5.9 show the ML algorithm to be 10 times better than the built-in algorithm while achieving the deceleration in the cycling aging process which, is evident from Table 5.8. The battery is just charged once to last through the simulation day as well as the day following the simulation. By this pattern, for the entire month of May, the battery will only be charged 15 times compared to the 30 times it is being charged as per [4].

The next chapter will utilize these results to perform the economic analysis which will further prove the ML algorithm to be much effective than the built-in algorithm in reducing the investment return period.

Chapter 6

ECONOMIC ANALYSIS

6.1 Introduction

An economic analysis (EA) is performed to showcase the capability of the ML algorithm to reduce the Demand Charge and the overall electricity bill by reducing the energy consumption during the on-peak hours from the grid. In this chapter, the economic analysis focuses on determining and comparing the savings for the customer in the case of ML algorithm and built-in algorithm. Four load profiles were simulated using these algorithms and thus economic analysis is performed on these four cases and appropriate comparisons/conclusions are drawn. Since this is a basic analysis to just show the impact of the ML algorithm, the benefits of decelerating cycling aging are not taken into account. Also, these calculations try to showcase the savings made in a single billing period and long-term savings due to the conservation of battery life, earnings from the off-peak hour energy export, off-peak hours energy charges are not considered. The calculations/tabulations are shown for two sets of data.

6.2 Assumptions

The EA performed is based upon the following assumptions:

- The simulations are run only during the summer on-peak hours and thus, the EA is based on the results obtained only during this 7-hour period.
- The off-peak hour happenings are not taken into account and are ignored completely.

- The EA shows only the on-peak hour savings and on-peak hour charges induced by the customer on to their electricity bill for this specific billing period.
- All the rates/charges are based on the E-27 plan of the utility.
- The entire EA is based on the assumption that for the entirety of the billing period of concern, the load profile of the house is identical to the simulated day load profile.
- Throughout this chapter, a negative sign in front of a cost indicates credits provided by the utility to the customer.
- A “charge” is paid by the customer to the utility and a “credit” is granted by the utility to the customer.
- The summer on-peak hours are from 13:00 through 20:00.
- All the load data, results are clustered into half-hour averages for simplicity as the utility handles Demand Charge calculation in half-hour intervals only.

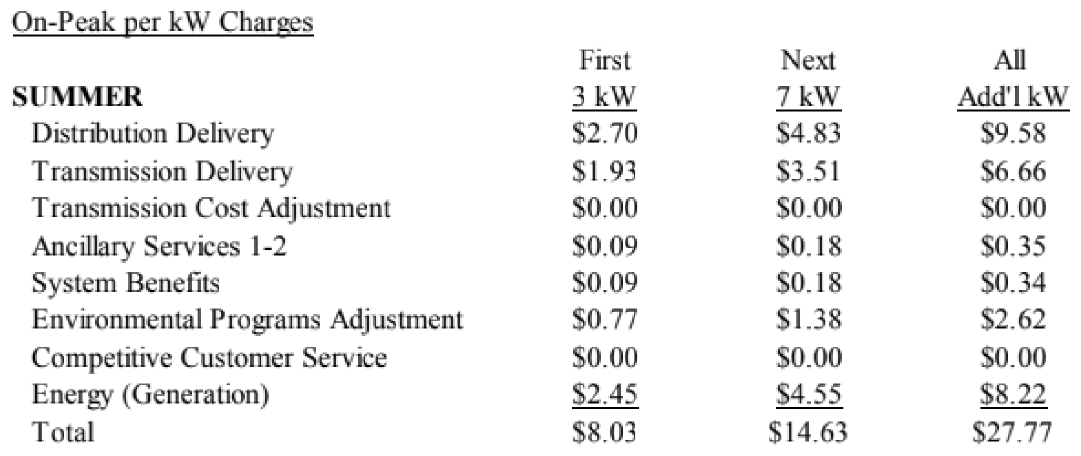
6.3 Procedure

- The site demand pertaining to both the algorithms for the simulation sets are tabulated.
- The E-27 plan is used as a reference for the costs.
- Based on the half-hour interval with the maximum positive power value, the Demand Charge calculation I performed.
- The net energy exported to the grid during the on-peak hours is calculated.
- The net energy imported from the grid during the on-peak hours is calculated.

- The appropriate costs from E-27 plan are multiplied to the numerical values obtained in the steps 3 through 5.
- The cost of energy exported is provided as a credit while the Demand Charge and energy imported are charged for. (Total = Demand Charge + Energy imported - Energy exported)
- The net electricity bill, assuming this specific pattern exists throughout the billing period, for this specific billing period is calculated.
- Steps 3 through 8 are repeated for both algorithms for both the load sets.
- The results are tabulated.

6.4 The Prerequisites

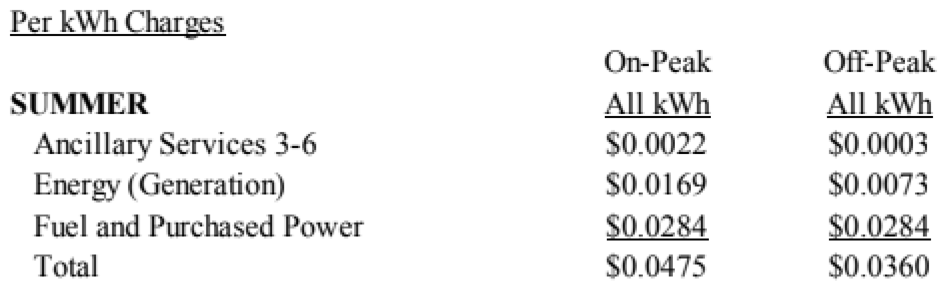
The Figures 6.1 and 6.2 provide the utility company E-27 charges.



On-Peak per kW Charges

SUMMER	First 3 kW	Next 7 kW	All Add'l kW
Distribution Delivery	$2.70	$4.83	$9.58
Transmission Delivery	$1.93	$3.51	$6.66
Transmission Cost Adjustment	$0.00	$0.00	$0.00
Ancillary Services 1-2	$0.09	$0.18	$0.35
System Benefits	$0.09	$0.18	$0.34
Environmental Programs Adjustment	$0.77	$1.38	$2.62
Competitive Customer Service	$0.00	$0.00	$0.00
Energy (Generation)	$2.45	$4.55	$8.22
Total	$8.03	$14.63	$27.77

Figure 6.1 Demand Charge for the summer months according to the E-27 plan.



Per kWh Charges

SUMMER	On-Peak All kWh	Off-Peak All kWh
Ancillary Services 3-6	$0.0022	$0.0003
Energy (Generation)	$0.0169	$0.0073
Fuel and Purchased Power	$0.0284	$0.0284
Total	$0.0475	$0.0360

Figure 6.2 Energy charges for the summer months according to the E-27 plan.

Tables 6.1 and 6.2 are the tabulation of the site demand for two sets of simulation.

Table 6. 1 Tabulation of the SD pertaining to the 1st set of simulations.

Interval	ML algorithm (kW)	Built-in algorithm (kW)
13:00	-2.67	-2.9
13:30	-2.53	-2.7
14:00	-2.28	-2.46
14:30	-1.92	-2.16
15:00	-1.46	-1.67
15:30	-0.82	-1.07
16:00	-0.56	-0.335
16:30	0.07	0.32
17:00	-0.29	1.04
17:30	0.28	1.74
18:00	-0.07	1.84
18:30	0.2	1.84
19:00	-0.05	1.63
19:30	0.09	2.12

- The maximum SD from the Table 6.1 for the ML algorithm is 0.283 kW.
- The maximum SD from the Table 6.1 for the built-in algorithm is 2.125 kW.
- The maximum SD has been reduced by 1.842 kW when the ML algorithm was used.

Table 6. 2 Tabulation of the SD pertaining to the 2nd set of simulations.

Interval	ML algorithm (kW)	Built-in algorithm (kW)
13:00	-2.8	-2.73
13:30	-2.47	-2.92
14:00	-2.29	-2.613
14:30	-2.02	-2.212
15:00	-1.65	-1.79
15:30	-1.26	-1.23
16:00	-0.63	0.122
16:30	-0.034	0.162
17:00	-0.298	0.642
17:30	0.197	1.24
18:00	-0.034	1.73
18:30	0.063	1.92
19:00	0.056	1.96
19:30	0.186	1.99

- The maximum SD from the Table 6.2 for the ML algorithm is 0.197 kW.
- The maximum SD from the Table 6.2 for the built-in algorithm is 1.991 kW.
- The maximum SD has been reduced by 1.793 kW when the ML algorithm was used.

The investment return period for a customer is mainly dependent on two factors — The reduction in Demand Charge paid and the income from the export of excess PV energy to the grid. The customer is charged for the energy consumed from the grid.

The economic analysis will hence be based on these parameters. There are two sets of data and the economic analysis is performed on both the sets separately, assuming each set to be representing the on-peak hours pattern of a summer billing period. All the rates chosen are according to the E-27 plan for the "Summer" billing cycles. According to the plan, 1 kWh of energy in the summer on-peak hour costs $0.0475 (both export and import). Similarly, a Demand Charge of $8.03 per kW is charged when the maximum demand is less than 3 kW.

6.5 The Calculation

The calculation procedure has been explained in detail in the section 6.3. The Tables 6.3 and 6.4 contain the calculations for the data tabulated in Tables 6.1 and 6.2 respectively.

Table 6. 3 Calculations pertaining to the 1st set of simulations.

Parameters	ML algorithm	Built-in algorithm
Maximum SD	0.283 kW	2.125 kW
Demand Charge	0.283 * 8.03 = $ 2.27	2.125 * 8.03 = $17.06
Total energy imported	0.32 kWh	5.267 kWh
Energy import charge (entire billing period)	0.32 * 0.0475 * 31 = $0.47	5.267 * 0.0475 * 31 = $7.755
Total energy exported	6.348 kWh	6.639 kWh
Energy export credit (entire billing period)	6.348 * 0.0475 * 31 = $9.347	6.639 * 0.0475 * 31 = $9.776
Bill	$2.27 + $0.47 - $9.347 = - $6.61	$17.06 + $7.755 - $9.776 = $15.04
Total amount	- $6.61	$15.04
Nature	Credit	Charge

Table 6. 4 Calculations pertaining to the 2nd set of simulations.

Parameters	ML algorithm	Built-in algorithm
Maximum SD	0.197 kW	1.991 kW
Demand Charge	0. 197 * 8.03 = $ 1.58	1.991 * 8.03 = $15.99
Total energy imported	0.251 kWh	4.906 kWh
Energy import charge (entire billing period)	0.251 * 0.0475 * 31 = $0.37	4.906 * 0.0475 * 31 = $7.224
Total energy exported	6.731 kWh	6.753 kWh
Energy export credit (entire billing period)	6.731 * 0.0475 * 31 = $9.911	6.753 * 0.0475 * 31 = $9.944
Bill	$1.58 + $0.37 - $9.911 = -$7.961	$15.99 + $7.224 - $9.944 = $13.27
Total amount	- $7.96	$13.27
Nature	Credit	Charge

Table 6.5 contains the consolidation of the results of EA based on the acquired data.

Table 6. 5 Consolidation of the results.

Day - Algorithm	Energy Cost	Demand Charge	Total Cost	Type
1 – Built-in	- $2.02	$17.06	$15.04	Charge
1 – ML	- $8.88	$2.27	- $6.60	Credit
2 – Built-in	- $2.72	$15.99	$13.27	Charge
2 – ML	- $9.54	$1.58	- $7.96	Credit
3 – Built-in	$15.53	$15.45	$30.98	Charge
3 – ML	-$1.63	$3.19	$1.56	Charge
4 – Built-in	$21.10	$15.44	$36.55	Charge
4 – ML	$0.12	$4.18	$4.3	Charge

6.6 Inference

This chapter reaffirms the initial speculation of the ML algorithm being far more efficient than the built-in algorithm. The results show the vast difference between the performance of the two algorithms. The ML algorithm on both the simulation sets earning the customer a net credit at the end of the billing period while the built-in algorithm results in a charge payable by the customer. This was expected, as mentioned, as the ML algorithm simply better prepared the system to face the challenges. With the ML algorithm in place, this test if repeated during the peak-summer will result in a higher saving for the customer as during the peak-summer, the solar availability for export is more and with the system better prepared, the amount of export could be maximized while minimizing the energy requirement and Demand Charge.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this research collaboration between the electric utility and ASU for the roof-top PV system research, a lot of studies has been carried out by three different researchers. But, this is just the second time the improvement of the built-in control algorithm has been the topic of research. Even previously, a complete revamp of the control strategy was not attempted owing to the difficulty in predicting the load and the PV with great accuracy. This research was specifically to answer the question of "How much improvement could we achieve over the performance of the built-in control algorithm?". The results show that a lot more could be extracted from the same hardware by just improving the control mechanism.

While Chapter 1 through 3 concentrate upon the explaining about the problem on hand and describing the system, the Chapter 4 on the implementation of the ML techniques to forecast load has clearly explained the methods adopted for forecasting the load. The chapter primarily focuses on the arguments supporting the specific features being extracted from the timestamp of the data to be passed into the ML algorithms. These features improved the accuracy of the ML algorithm tremendously while providing consistent results over different scenarios. The results presented at the end of the chapter supports the arguments presented in the chapter.

Chapter 5 of this thesis deals with the prediction of the PV and development of the battery control algorithm. Justification of each and every decision made has been provided while a comparison of the performances of the ML and the built-in algorithms in the results section through tables and plots effectively shows the ML algorithm to be a clear winner. A simple logical modification to the built-in control strategy by effectively utilizing the battery only when necessary not only suits the requirement but also ends up saving money and preventing hardware wear and tear. The Chapter 6 on the economic analysis is a sanity check on the results obtained in Chapter 5. The results are presented in terms of dollars saved instead of the kW units adopted earlier.

The results show the Machine Learning algorithm to reduce the Demand Charge by a factor 10, while generating income by exporting the excess PV into the grid. The investment return period is thus, significantly reduced, with the added benefit of decelerating the 'cycling aging' of the battery by planned charge/discharge cycles. This will encourage people to embrace the new technology and along with efforts from the utilities, this could help the state achieve the renewable penetration target of 2025.

7.2 Future Work

As explained earlier the agenda of this research was to develop a complete control strategy from scratch using ML techniques. Since this is just a first step in such a direction for the problems faced in this specific case, there is scope for improvement. Some of the possibilities are,

- More features like specifying the busy hours of the day could be passed to the ML algorithm.
- Owing to the feedback mechanism, the ML algorithm keeps improving over time. This could be utilized by passing the last 3 years data as training data instead of just 1. This would improve the accuracy significantly.

- The problem could be treated as a Supervised learning Regression problem to better follow the load without the error induced due to the creation of the classification labels.
- Based on research it was observed that Neural Networks algorithms tend to work better in case of time series problems. Thus, this could be a improvement.
- The PV prediction could be improved further by varying the parameters passed to the PVWATTS Calculator by better studying the hardware installed.

The author is currently working on the improvements suggested above. Also, the findings from this thesis presented here will be published in a reputed journal in the near future.

REFERENCES

[1] "U.S. Energy Information Administration - EIA - Independent Statistics and Analysis," U.S. Energy Information Administration (EIA), 21-Dec-2017. [Online]. Available: https://www.eia.gov/state/analysis.php?sid=AZ.

[2] "ASU Solar," Business and Finance, 28-Mar-2018. [On- line]. Available: https://cfo.asu.edu/solar.

[3] P. Etha, A. K. Janjua, V. Scholar, A. Chelladurai, G. Student, and G. G. Karady, "Customer benefit optimization for residential PV with energy storage systems," *2017 IEEE Power & Energy Society General Meeting,* Feb. 2018.

[4] Etha Pavan, "Customer Benefit Analysis and Experimental Study of Residential Rooftop PV and Energy Storage Systems." *Master's thesis dissertation, ASU*, 2017.

[5] "Random Forests Leo Breiman and Adele Cutler," Statistics at UC Berkeley. [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/

[6] "PVWatts," PVWatts Calculator. [Online]. Available: https://pvwatts.nrel.gov/.

[7] E. Cogliani, "The Role of the Direct Normal Irradiance (DNI) Forecasting in the Operation of Solar Concentrating Plants," Energy Procedia, vol. 49, pp. 1612–1621, 2014.

[8] M. Ecker, N. Nieto, S. Käbitz, J. Schmalstieg, H. Blan ke, A. Warnecke und D. U. Sauer. Calendar and cycle life study of Li(NiMnCo)O2- based 18650 lithiumion-Ion batteries. *Journal of Power Sources*; 2014

[9] Basics of MPPT Solar Charge Controller. (n.d.). Retrieved from http://www.leonics.com/support/article2_14j/articles2_14j_en.php

[10] Raspberry Pi 3 Model B. (n.d.). Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b

[11] Bose, B. (2010). Global Warming: Energy, Environmental Pollution, and the Impact of Power Electronics. *IEEE Industrial Electronics Magazine*,4(1), 6-17. doi:10.1109/mie.2010.935860

[12] Johnson, J. (2013, July 10). Random forests. Retrieved from https://shapeofdata.wordpress.com/2013/07/09/random-forests/

[13] He, W. (2017). Load Forecasting via Deep Neural Networks. *Procedia Computer Science, 122*, 308-314. doi:10.1016/j.procs.2017.11.374

[14] Warrior, K. P., Shrenik, M., & Soni, N. (2016). Short-term electrical load forecasting using predictive machine learning models. *2016 IEEE Annual India Conference* (INDICON), 1-6. doi:10.1109/indicon.2016.7839103

[15] Yazdi, F. M. (2009). Application of Neural Networks for 24-Hour-Ahead Load Forecasting. *2009 World Academy of Science, Engineering and Technology International Journal of Electrical and Computer Engineering,* Vol:3, No:2.

[16] Yoon, Y., & Kim, Y. H. (2014). Charge Scheduling of an Energy Storage System under Time-of-Use Pricing and a Demand Charge. *The Scientific World Journal*, 2014.

APPENDIX A

MONTH CLUSTERING DETAILS

A. Month Clustering details

The months of the year are clustered based on the temperature average of the month for the past 100 years in the state of Arizona. The Table A1 shows the clusters.

Table A 1 Month clusters.

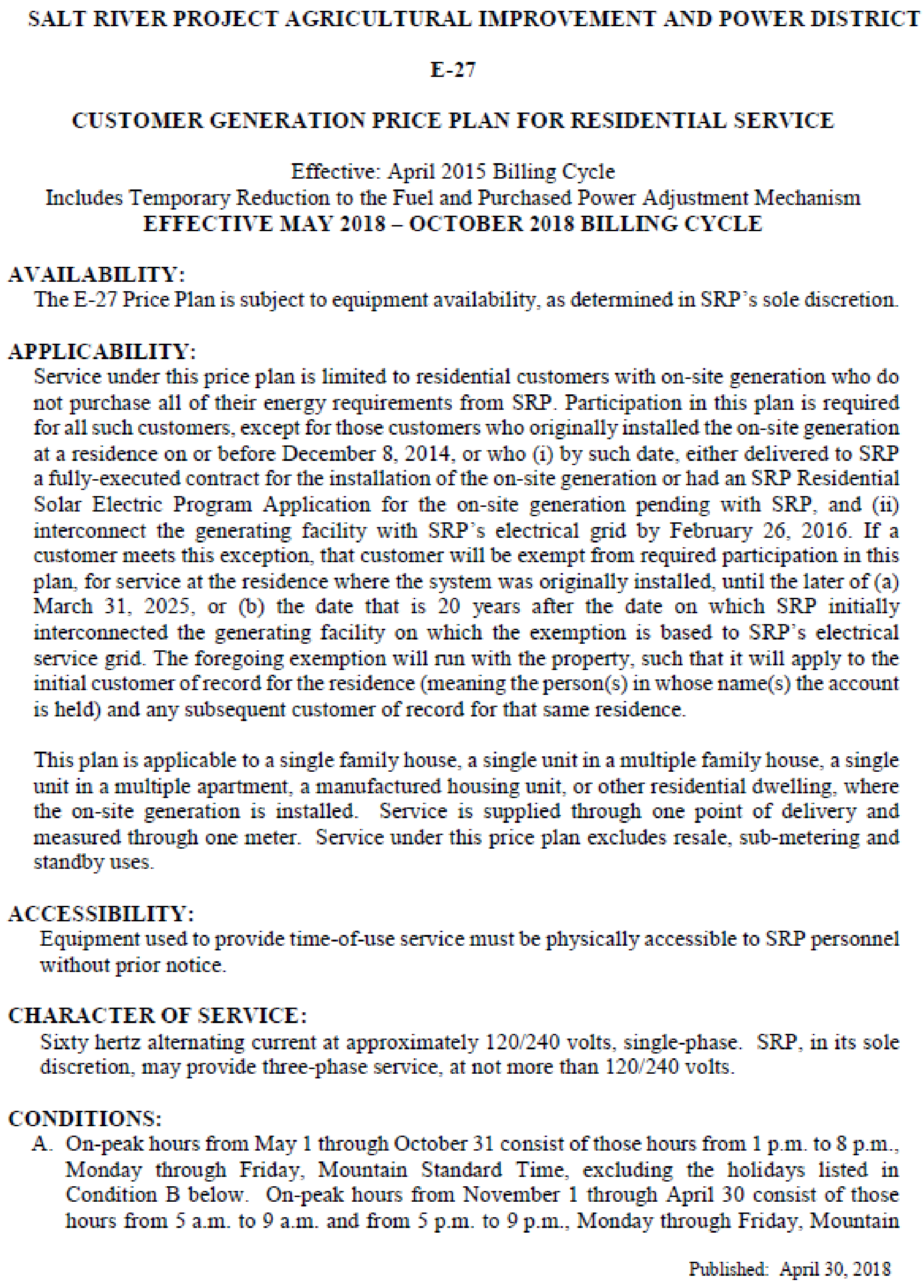
Cluster	Months included
1	January, February and December
2	March and November
3	April, May and October
4	June and September
5	July and August

APPENDIX B

THE E-27 PLAN

B. The E-27 Plan

The E-27 plan of summer 2018 is included here. The Figures B1 through B6 clearly shows in detail the various charges associated with this plan and the conditions specific to the users of the E-27 plan. The images were provided by SRP.



SALT RIVER PROJECT AGRICULTURAL IMPROVEMENT AND POWER DISTRICT

E-27

CUSTOMER GENERATION PRICE PLAN FOR RESIDENTIAL SERVICE

Effective: April 2015 Billing Cycle
Includes Temporary Reduction to the Fuel and Purchased Power Adjustment Mechanism
EFFECTIVE MAY 2018 – OCTOBER 2018 BILLING CYCLE

AVAILABILITY:
The E-27 Price Plan is subject to equipment availability, as determined in SRP's sole discretion.

APPLICABILITY:
Service under this price plan is limited to residential customers with on-site generation who do not purchase all of their energy requirements from SRP. Participation in this plan is required for all such customers, except for those customers who originally installed the on-site generation at a residence on or before December 8, 2014, or who (i) by such date, either delivered to SRP a fully-executed contract for the installation of the on-site generation or had an SRP Residential Solar Electric Program Application for the on-site generation pending with SRP, and (ii) interconnect the generating facility with SRP's electrical grid by February 26, 2016. If a customer meets this exception, that customer will be exempt from required participation in this plan, for service at the residence where the system was originally installed, until the later of (a) March 31, 2025, or (b) the date that is 20 years after the date on which SRP initially interconnected the generating facility on which the exemption is based to SRP's electrical service grid. The foregoing exemption will run with the property, such that it will apply to the initial customer of record for the residence (meaning the person(s) in whose name(s) the account is held) and any subsequent customer of record for that same residence.

This plan is applicable to a single family house, a single unit in a multiple family house, a single unit in a multiple apartment, a manufactured housing unit, or other residential dwelling, where the on-site generation is installed. Service is supplied through one point of delivery and measured through one meter. Service under this price plan excludes resale, sub-metering and standby uses.

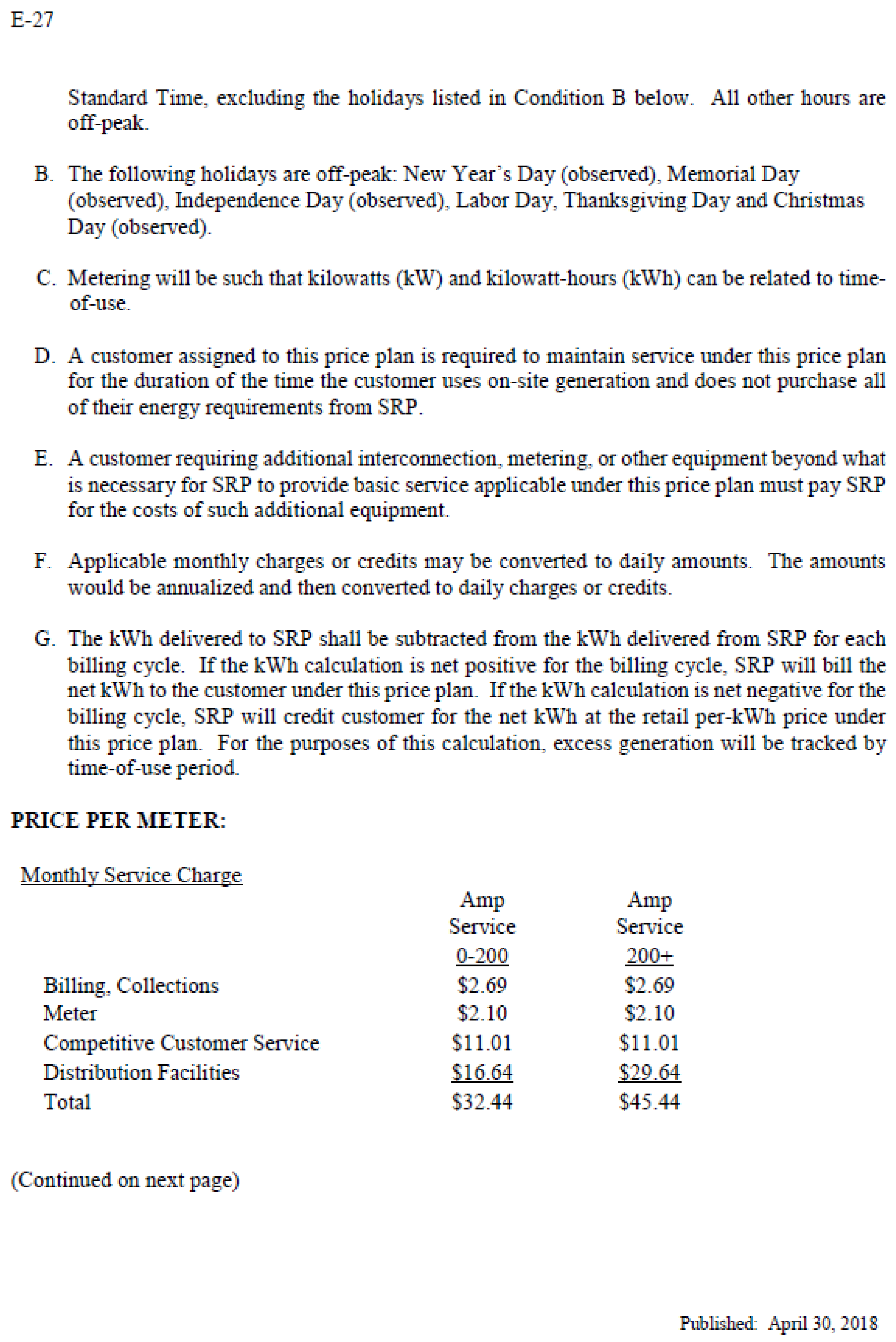
ACCESSIBILITY:
Equipment used to provide time-of-use service must be physically accessible to SRP personnel without prior notice.

CHARACTER OF SERVICE:
Sixty hertz alternating current at approximately 120/240 volts, single-phase. SRP, in its sole discretion, may provide three-phase service, at not more than 120/240 volts.

CONDITIONS:
A. On-peak hours from May 1 through October 31 consist of those hours from 1 p.m. to 8 p.m., Monday through Friday, Mountain Standard Time, excluding the holidays listed in Condition B below. On-peak hours from November 1 through April 30 consist of those hours from 5 a.m. to 9 a.m. and from 5 p.m. to 9 p.m., Monday through Friday, Mountain

Published: April 30, 2018

Figure B1 Page 1 of the E-27 plan.



E-27

Standard Time, excluding the holidays listed in Condition B below. All other hours are off-peak.

B. The following holidays are off-peak: New Year's Day (observed), Memorial Day (observed), Independence Day (observed), Labor Day, Thanksgiving Day and Christmas Day (observed).

C. Metering will be such that kilowatts (kW) and kilowatt-hours (kWh) can be related to time-of-use.

D. A customer assigned to this price plan is required to maintain service under this price plan for the duration of the time the customer uses on-site generation and does not purchase all of their energy requirements from SRP.

E. A customer requiring additional interconnection, metering, or other equipment beyond what is necessary for SRP to provide basic service applicable under this price plan must pay SRP for the costs of such additional equipment.

F. Applicable monthly charges or credits may be converted to daily amounts. The amounts would be annualized and then converted to daily charges or credits.

G. The kWh delivered to SRP shall be subtracted from the kWh delivered from SRP for each billing cycle. If the kWh calculation is net positive for the billing cycle, SRP will bill the net kWh to the customer under this price plan. If the kWh calculation is net negative for the billing cycle, SRP will credit customer for the net kWh at the retail per-kWh price under this price plan. For the purposes of this calculation, excess generation will be tracked by time-of-use period.

PRICE PER METER:

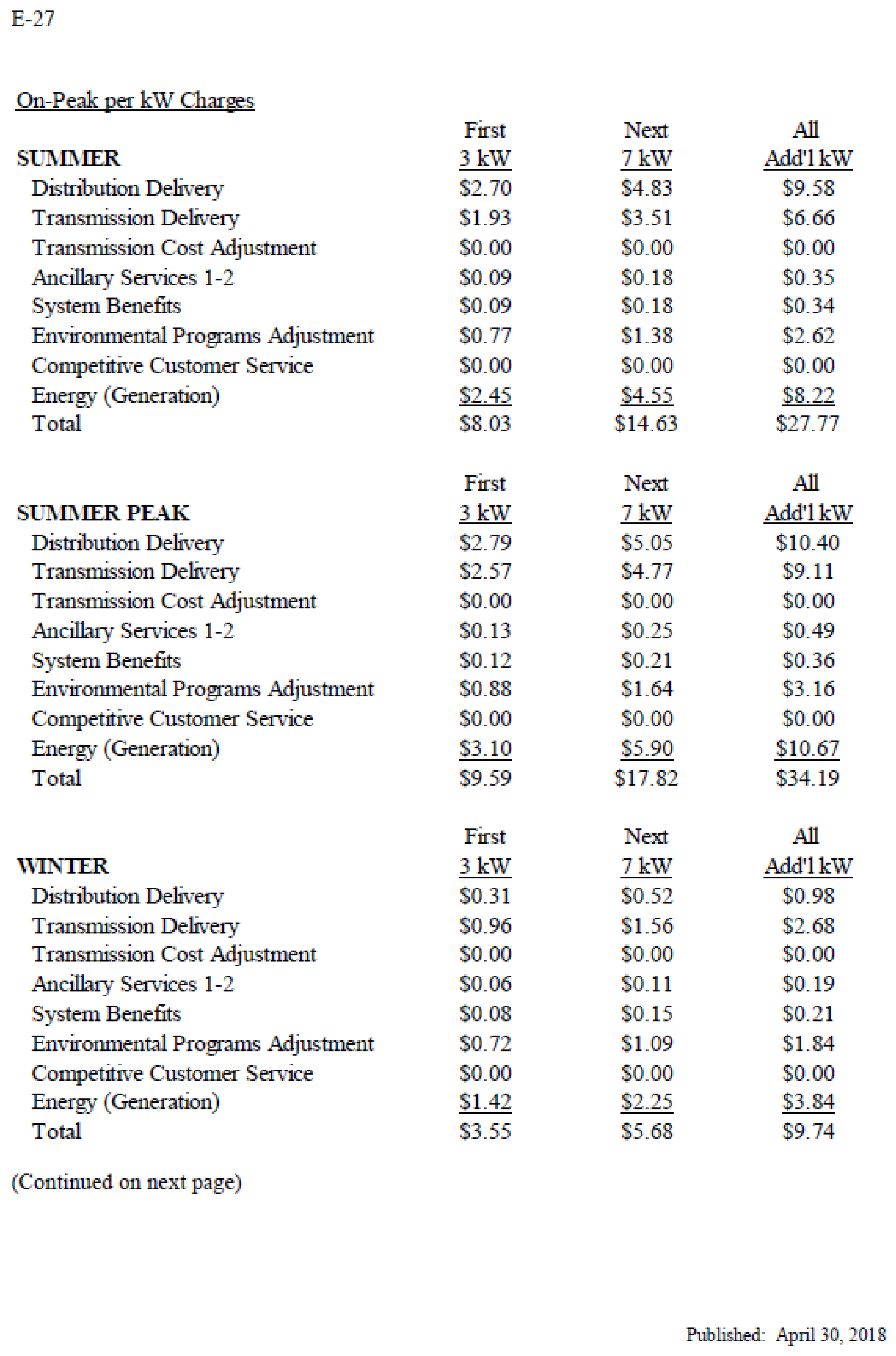
Monthly Service Charge

	Amp Service 0-200	Amp Service 200+
Billing, Collections	$2.69	$2.69
Meter	$2.10	$2.10
Competitive Customer Service	$11.01	$11.01
Distribution Facilities	$16.64	$29.64
Total	$32.44	$45.44

(Continued on next page)

Published: April 30, 2018

Figure B2 Page 2 of the E-27 plan.


E-27

On-Peak per kW Charges

SUMMER	First 3 kW	Next 7 kW	All Add'l kW
Distribution Delivery	$2.70	$4.83	$9.58
Transmission Delivery	$1.93	$3.51	$6.66
Transmission Cost Adjustment	$0.00	$0.00	$0.00
Ancillary Services 1-2	$0.09	$0.18	$0.35
System Benefits	$0.09	$0.18	$0.34
Environmental Programs Adjustment	$0.77	$1.38	$2.62
Competitive Customer Service	$0.00	$0.00	$0.00
Energy (Generation)	$2.45	$4.55	$8.22
Total	$8.03	$14.63	$27.77

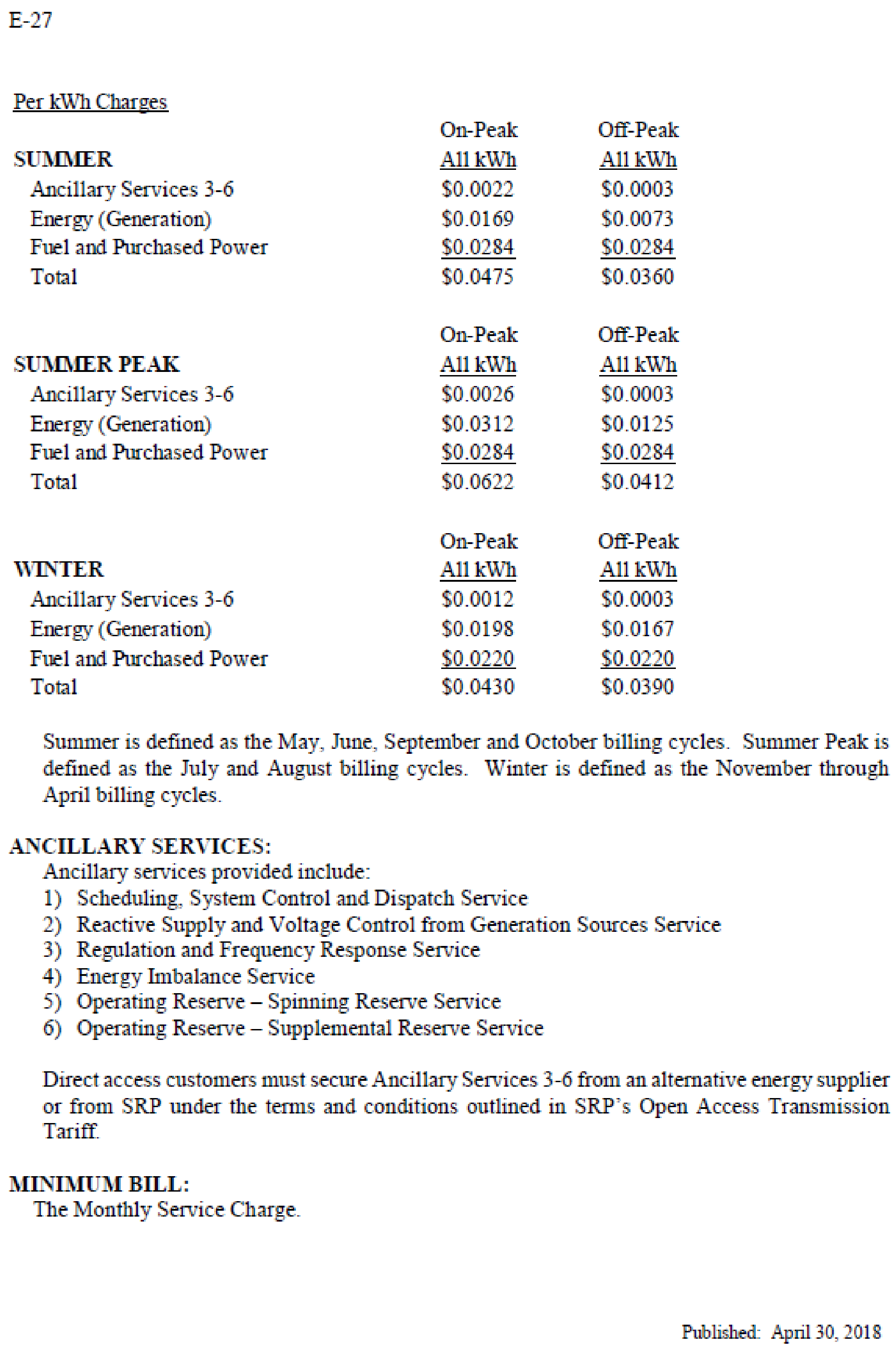
SUMMER PEAK	First 3 kW	Next 7 kW	All Add'l kW
Distribution Delivery	$2.79	$5.05	$10.40
Transmission Delivery	$2.57	$4.77	$9.11
Transmission Cost Adjustment	$0.00	$0.00	$0.00
Ancillary Services 1-2	$0.13	$0.25	$0.49
System Benefits	$0.12	$0.21	$0.36
Environmental Programs Adjustment	$0.88	$1.64	$3.16
Competitive Customer Service	$0.00	$0.00	$0.00
Energy (Generation)	$3.10	$5.90	$10.67
Total	$9.59	$17.82	$34.19

WINTER	First 3 kW	Next 7 kW	All Add'l kW
Distribution Delivery	$0.31	$0.52	$0.98
Transmission Delivery	$0.96	$1.56	$2.68
Transmission Cost Adjustment	$0.00	$0.00	$0.00
Ancillary Services 1-2	$0.06	$0.11	$0.19
System Benefits	$0.08	$0.15	$0.21
Environmental Programs Adjustment	$0.72	$1.09	$1.84
Competitive Customer Service	$0.00	$0.00	$0.00
Energy (Generation)	$1.42	$2.25	$3.84
Total	$3.55	$5.68	$9.74

(Continued on next page)

Published: April 30, 2018

Figure B3 Page 3 of the E-27 plan.


E-27

Per kWh Charges

SUMMER	On-Peak All kWh	Off-Peak All kWh
Ancillary Services 3-6	$0.0022	$0.0003
Energy (Generation)	$0.0169	$0.0073
Fuel and Purchased Power	$0.0284	$0.0284
Total	$0.0475	$0.0360

SUMMER PEAK	On-Peak All kWh	Off-Peak All kWh
Ancillary Services 3-6	$0.0026	$0.0003
Energy (Generation)	$0.0312	$0.0125
Fuel and Purchased Power	$0.0284	$0.0284
Total	$0.0622	$0.0412

WINTER	On-Peak All kWh	Off-Peak All kWh
Ancillary Services 3-6	$0.0012	$0.0003
Energy (Generation)	$0.0198	$0.0167
Fuel and Purchased Power	$0.0220	$0.0220
Total	$0.0430	$0.0390

Summer is defined as the May, June, September and October billing cycles. Summer Peak is defined as the July and August billing cycles. Winter is defined as the November through April billing cycles.

ANCILLARY SERVICES:

Ancillary services provided include:

1) Scheduling, System Control and Dispatch Service
2) Reactive Supply and Voltage Control from Generation Sources Service
3) Regulation and Frequency Response Service
4) Energy Imbalance Service
5) Operating Reserve – Spinning Reserve Service
6) Operating Reserve – Supplemental Reserve Service

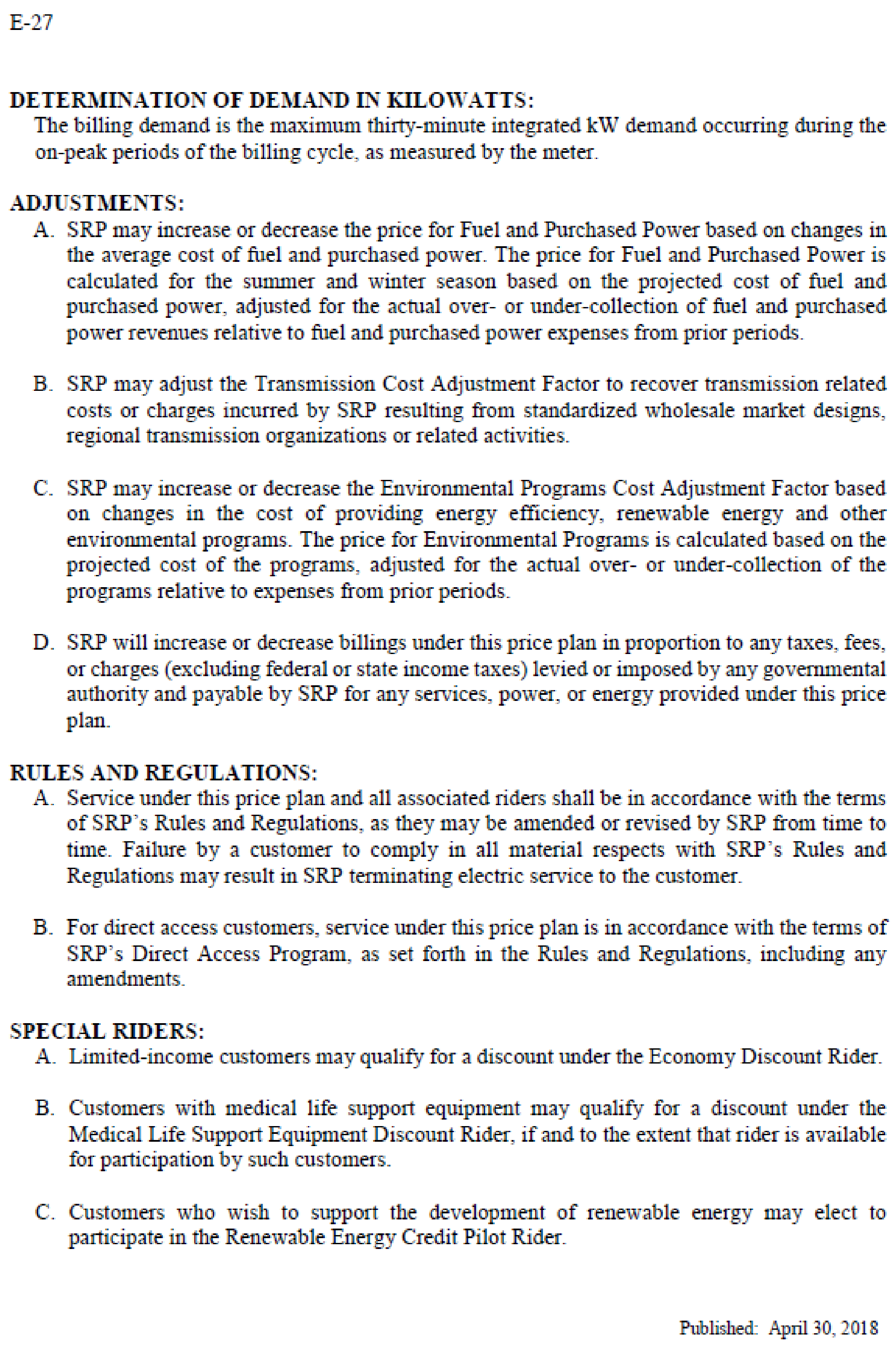
Direct access customers must secure Ancillary Services 3-6 from an alternative energy supplier or from SRP under the terms and conditions outlined in SRP's Open Access Transmission Tariff.

MINIMUM BILL:

The Monthly Service Charge.

Published: April 30, 2018

Figure B4 Page 4 of the E-27 plan.



E-27

DETERMINATION OF DEMAND IN KILOWATTS:

The billing demand is the maximum thirty-minute integrated kW demand occurring during the on-peak periods of the billing cycle, as measured by the meter.

ADJUSTMENTS:

A. SRP may increase or decrease the price for Fuel and Purchased Power based on changes in the average cost of fuel and purchased power. The price for Fuel and Purchased Power is calculated for the summer and winter season based on the projected cost of fuel and purchased power, adjusted for the actual over- or under-collection of fuel and purchased power revenues relative to fuel and purchased power expenses from prior periods.

B. SRP may adjust the Transmission Cost Adjustment Factor to recover transmission related costs or charges incurred by SRP resulting from standardized wholesale market designs, regional transmission organizations or related activities.

C. SRP may increase or decrease the Environmental Programs Cost Adjustment Factor based on changes in the cost of providing energy efficiency, renewable energy and other environmental programs. The price for Environmental Programs is calculated based on the projected cost of the programs, adjusted for the actual over- or under-collection of the programs relative to expenses from prior periods.

D. SRP will increase or decrease billings under this price plan in proportion to any taxes, fees, or charges (excluding federal or state income taxes) levied or imposed by any governmental authority and payable by SRP for any services, power, or energy provided under this price plan.

RULES AND REGULATIONS:

A. Service under this price plan and all associated riders shall be in accordance with the terms of SRP's Rules and Regulations, as they may be amended or revised by SRP from time to time. Failure by a customer to comply in all material respects with SRP's Rules and Regulations may result in SRP terminating electric service to the customer.

B. For direct access customers, service under this price plan is in accordance with the terms of SRP's Direct Access Program, as set forth in the Rules and Regulations, including any amendments.

SPECIAL RIDERS:

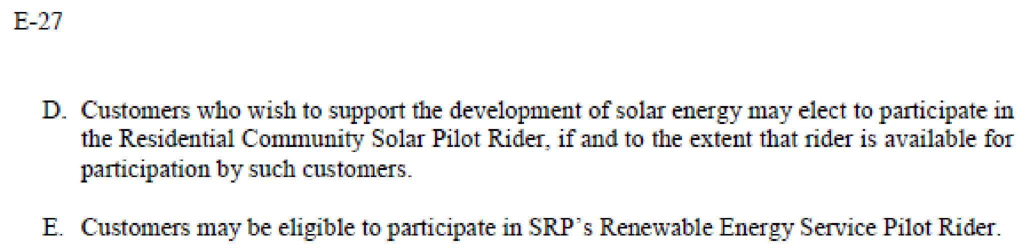
A. Limited-income customers may qualify for a discount under the Economy Discount Rider.

B. Customers with medical life support equipment may qualify for a discount under the Medical Life Support Equipment Discount Rider, if and to the extent that rider is available for participation by such customers.

C. Customers who wish to support the development of renewable energy may elect to participate in the Renewable Energy Credit Pilot Rider.

Published: April 30, 2018

Figure B5 Page 5 of the E-27 plan.


E-27

D. Customers who wish to support the development of solar energy may elect to participate in the Residential Community Solar Pilot Rider, if and to the extent that rider is available for participation by such customers.

E. Customers may be eligible to participate in SRP's Renewable Energy Service Pilot Rider.

Figure B6 Page 6 of the E-27 plan.

APPENDIX C

PYTHON CODES

C. Python Codes

Code to decide the load bank units to switch on.

```
# Importing packages.
import pandas as pd
import numpy as np

# Date ranges for which the data is processed.
# year-month-day hour:minute:second format. The time is the same always. Just
change the date and use.
date_from = '2014-05-1 00:00:00'
# year-month-day hour:minute:second format. The time is the same always. Just
change the date and use.
date_to = '2014-05-1 23:30:00'

data = pd.read_csv('halfHourLoad.csv', encoding='latin-1') # ,

# Change to 24 hours format
data['Time'] = pd.to_datetime(data['Time']).dt.strftime('%H:%M')

# Changing the date and time columns into one and converting them into pandas
recogonized date and time format column.
data['Date'] = data['Date'] + ' ' + data['Time']
data['Date'] = pd.to_datetime(data['Date'], format='%d/%m/%Y %H:%M')

# Dropping the time column now, since it has been combined to the date column.
data = data.drop(['Time'], axis=1)
# Converting the from and to dates interms of row indices of the dataframe.
date_from = (data.loc[data['Date'] == date_from]).index.values
date_to = (data.loc[data['Date'] == date_to]).index.values

# List to hold the number of burners and resistors to be switched on every 15
minutes.
length = int(date_to)+1-int(date_from)
burners = list(np.empty(length))
resistors = list(np.empty(length))
# Act as index for the lists storing the burner and resistor numbers. i is to iterate
between the from and
# to dates and thus may not be from 0 always.
count = 0

# Iterating through the from and to dates.
for i in range (int(date_from), int(date_to)+1):
# Checking the number of burners needed for this load.
   noOfBurners = data['kW (avg)'][i]/0.870
#If the number of burners required is greater than 10, i.e. load is larger than 8800w,
we will need the resistors as well as turning ON all the burners.
   if(noOfBurners>10):
        burners[count]= 10
# This difference is the watts that needs to be supplied by the resistors.
        difference = data['kW (avg)'][i] - (burners[count]*0.870)
# Number of resistor pairs that need to be switched ON.
        resistors[count]= round(difference/0.072)
        count = count + 1
# Continue with the next iteration.
        continue
# If the burners required is less than 10, then switch ON the appropriate number of
burners and
# for the deficit, turn ON the smaller resistors.
    burners[count]= int(noOfBurners)
    difference = data['kW (avg)'][i] - (burners[count]*0.870)
    resistors[count]= round(difference/0.072)
    count = count + 1

# Converting the 2 lists into a dataframe so that it can be written into a csv file.
# We also mention the appropriate headers for the columns. This .csv file serves as
the input to the
# programn that decides the Raspberry Pi commands.
df = pd.DataFrame(data={'880wStep': burners, '72wStep': resistors})
df.to_csv('load.csv')
```

Code to generate the governing code of the Raspberry Pi for larger load bank.

```
# Importing packages
import random
import pandas as pd

# Reading the data that contains the number of loads from each bank to be turned
on.
data = pd.read_csv('load.csv', encoding='latin-1')

# Opening a .py file.
file = open("bigBank.py","w+")

# Writing into the file. Import statements here
file.write("import RPi.GPIO as GPIO\n")
file.write("import time\n")
# This cleanup clears all the set configs. Everything from pin numbering type
(board/BCM), pin settings (high/low) etc. Everything goest otheir default state of
High.
file.write("GPIO.cleanup()\n")

# The reason we want to set the mode here is because there are actually two labels
for all of the pins, Broadcom (BCM) and board (BOARD). The board option will let
you refer to the pin's actual number on the board,
# and the Broadcom number is the actual pin number that the Broadcom chip
considers it to be.
# It seems to be that BCM specification is the *actual* pin number. We'll use
"BOARD".
file.write("GPIO.setmode(GPIO.BOARD)\n")
file.write("\n\n")

# The pins of Raspberry pi we want to trigger. These are the Board numbers and not
the BCM numbers! REMEMBER!!
pinsUsed = (5,33,23,15,3,40,38,18,16,12)
# Initially setting up these pins on the Raspberry Pi.
for pin in pinsUsed:
   string = "GPIO.setup(" + str(pin) + ", GPIO.OUT)\n"
   file.write(string)

file.write("\n\n")
# Initially setting everything low.
for i in pinsUsed:
      string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
      file.write(string)

# Sleep command. Change this depending on the hours left untill the midnight.
file.write("time.sleep(600)\n")
file.write("\n\n")
```

```
#################### ACTUAL TURN ON-OFF STARTS HERE###############
for index, row in data.iterrows():
# Making a duplicate of the list of pins so as to alter it without data loss.
    tempo = list(pinsUsed)
    x = data['880wStep'][index]
# Turn on the appropriate pins. Choose a random pin from the temporary list to turn
on.
# Remove that pin from the temporary list of available pins and repeat the process
till the number of units to be switched ON is achieved.
    for i in range(1, x+1):
        pinChosen = random.choice(tempo)
        string = "GPIO.output(" + str(pinChosen) + ", GPIO.HIGH)\n"
        file.write(string)
        tempo.remove(pinChosen)
# Switching off the ones that have not been turned on in that particular cycle.
    for i in tempo:
        string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
        file.write(string)
# Maintain the set pin config for the next 15 mins.
    file.write("time.sleep(900)\n")
    file.write("\n\n")

# Set all pins to 'Low' as the final config so that till the next load profile comes in, all
the resistors will be in off state
# saving power and wear and tear.
for i in pinsUsed:
        string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
        file.write(string)

# Close the file stream.
file.close()
```

Code governing the larger load bank for a specific case.

```
import RPi.GPIO as GPIO
import time
GPIO.cleanup()
GPIO.setmode(GPIO.BOARD)


GPIO.setup(5, GPIO.OUT)
GPIO.setup(33, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(3, GPIO.OUT)
GPIO.setup(40, GPIO.OUT)
GPIO.setup(38, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)


GPIO.output(5, GPIO.LOW)
GPIO.output(33, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
GPIO.output(15, GPIO.LOW)
GPIO.output(3, GPIO.LOW)
GPIO.output(40, GPIO.LOW)
GPIO.output(38, GPIO.LOW)
GPIO.output(18, GPIO.LOW)
GPIO.output(16, GPIO.LOW)
GPIO.output(12, GPIO.LOW)
time.sleep(600)


GPIO.output(38, GPIO.HIGH)
GPIO.output(5, GPIO.LOW)
GPIO.output(33, GPIO.LOW)
GPIO.output(23, GPIO.HIGH)
GPIO.output(15, GPIO.LOW)
GPIO.output(3, GPIO.HIGH)
GPIO.output(40, GPIO.HIGH)
GPIO.output(18, GPIO.HIGH)
GPIO.output(16, GPIO.LOW)
GPIO.output(12, GPIO.LOW)
time.sleep(900)
```

Code to generate the governing code of the Raspberry Pi for smaller load bank.

```
# Importing packages
import random
import pandas as pd

# Reading the data that contains the number of loads from each bank to be turned
on.
data = pd.read_csv('load.csv', encoding='latin-1')

# Opening a .py file.
file = open("smallBank.py","w+")

# Writing into the file. Import statements here
file.write("import RPi.GPIO as GPIO\n")
file.write("import time\n")
# This cleanup clears all the set configs. Everything from pin numbering type
(board/BCM), pin settings (high/low) etc. Everything goest otheir default state of
High.
file.write("GPIO.cleanup()\n")

# The reason we want to set the mode here is because there are actually two labels
for all of the pins, Broadcom (BCM) and board (BOARD). The board option will let
you refer to the pin's actual number on the board,
# and the Broadcom number is the actual pin number that the Broadcom chip
considers it to be.
# It seems to be that BCM specification is the *actual* pin number. We'll use
"BOARD".
file.write("GPIO.setmode(GPIO.BOARD)\n")
file.write("\n\n")

# The pins of Raspberry pi we want to trigger. These are the Board numbers and not
the BCM numbers! REMEMBER!!
# all the pins of Pi that are connected to the load. Even the 144w ones.
pinsUsed = [3,5,7,11,13,15,19,21,23,29,31,33,35,37,12,16,18,22,24,26,32,36,38,40]
# List of pins connected to 2 resistor clusters i.e. 72w only.
pinsUsed72w = [5,7,11,13,15,19,21,29,31,33,35,37,16,18,22,24,26,32,36,38,40]
# List of pins connected to 2 resistor clusters i.e. 200w only.
pinsUsed200w = [3,12,23]
# Initially setting up these pins on the Raspberry Pi.
for pin in pinsUsed:
   string = "GPIO.setup(" + str(pin) + ", GPIO.OUT)\n"
   file.write(string)

file.write("\n\n")
# Initially setting everything low.
for i in pinsUsed:
      string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
      file.write(string)
```

```
# Sleep command. Change this depending on the hours left untill the midnight.
file.write("time.sleep(600)\n")
file.write("\n\n")

#################### ACTUAL TURN ON-OFF STARTS
HERE##########################################
for index, row in data.iterrows():
# Making a duplicate of the list of pins so as to alter it without data loss.
    tempo = list(pinsUsed72w)
    tempo2 = list(pinsUsed200w)
    x = int(data['72wStep'][index])
# To deal with cases where the load is greater than the load bank capacity.
    if x>29:
        x = 29
# First, when x is greater than 3, then we switch on the 200w cluster first. When
there is no more 200w left or
# when the requirement is below 200w step, then we go to the 72 w.
    while (x>=3 and tempo2):
        pinChosen = random.choice(tempo2)
        string = "GPIO.output(" + str(pinChosen) + ", GPIO.HIGH)\n"
        file.write(string)
        tempo2.remove(pinChosen)
        x = x-3
# This is to make all the unused 144w as 'Low'.
    for i in tempo2:
        string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
        file.write(string)

# Turn on the appropriate pins. Choose a random pin from the temporary list to turn
on.
# Remove that pin from the temporary list of available pins and repeat the process
till the number of units to be switched ON is achieved.
    for i in range(1, x+1):
        pinChosen = random.choice(tempo)
        string = "GPIO.output(" + str(pinChosen) + ", GPIO.HIGH)\n"
        file.write(string)
        tempo.remove(pinChosen)
# Switching off the ones that have not been turned on in that particular cycle.
    for i in tempo:
        string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
        file.write(string)
# Maintain the set pin config for the next 15 mins.
    file.write("time.sleep(900)\n")
    file.write("\n\n")

# Set all pins to 'Low' as the final config so that till the next load profile comes in, all
the resistors will be in off state
# saving power and wear and tear.
for i in pinsUsed:
        string = "GPIO.output(" + str(i) + ", GPIO.LOW)\n"
        file.write(string)

# Close the file stream.
file.close()
```

Code governing the smaller load bank for a specific case.

```
import RPi.GPIO as GPIO
import time
GPIO.cleanup()
GPIO.setmode(GPIO.BOARD)


GPIO.setup(3, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(19, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(29, GPIO.OUT)
GPIO.setup(31, GPIO.OUT)
GPIO.setup(33, GPIO.OUT)
GPIO.setup(35, GPIO.OUT)
GPIO.setup(37, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(32, GPIO.OUT)
GPIO.setup(36, GPIO.OUT)
GPIO.setup(38, GPIO.OUT)
GPIO.setup(40, GPIO.OUT)


GPIO.output(3, GPIO.LOW)
GPIO.output(5, GPIO.LOW)
GPIO.output(7, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(13, GPIO.LOW)
GPIO.output(15, GPIO.LOW)
GPIO.output(19, GPIO.LOW)
GPIO.output(21, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
GPIO.output(29, GPIO.LOW)
GPIO.output(31, GPIO.LOW)
GPIO.output(33, GPIO.LOW)
GPIO.output(35, GPIO.LOW)
GPIO.output(37, GPIO.LOW)
GPIO.output(12, GPIO.LOW)
GPIO.output(16, GPIO.LOW)
GPIO.output(18, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(24, GPIO.LOW)
GPIO.output(26, GPIO.LOW)
GPIO.output(32, GPIO.LOW)
GPIO.output(36, GPIO.LOW)
GPIO.output(38, GPIO.LOW)
GPIO.output(40, GPIO.LOW)
time.sleep(600)


GPIO.output(12, GPIO.HIGH)
GPIO.output(3, GPIO.HIGH)
GPIO.output(23, GPIO.HIGH)
GPIO.output(29, GPIO.HIGH)
GPIO.output(24, GPIO.HIGH)
GPIO.output(31, GPIO.HIGH)
GPIO.output(16, GPIO.HIGH)
GPIO.output(26, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)
GPIO.output(38, GPIO.HIGH)
GPIO.output(36, GPIO.HIGH)
GPIO.output(19, GPIO.HIGH)
GPIO.output(37, GPIO.HIGH)
GPIO.output(18, GPIO.HIGH)
GPIO.output(5, GPIO.LOW)
GPIO.output(7, GPIO.LOW)
GPIO.output(11, GPIO.LOW)
GPIO.output(13, GPIO.LOW)
GPIO.output(15, GPIO.LOW)
GPIO.output(21, GPIO.LOW)
GPIO.output(33, GPIO.LOW)
GPIO.output(35, GPIO.LOW)
GPIO.output(32, GPIO.LOW)
GPIO.output(40, GPIO.LOW)
time.sleep(1800)
```

Code to predict the power requirement from the battery at any point in time.

```
# The import statments for the library used.
import pandas as pd
import numpy as np

# Reading in the load prediction data.
data = pd.read_csv('prediction.csv', encoding='latin-1')
# Dropping the date and other unnecessary columns now, to reduce the size of the
data frame and increase the processing speed.
data = data.drop(['Actual'], axis=1)
data = data.drop(['Deviation'], axis=1)
data = data.drop(['Max Deviation'], axis=1)
data = data.drop(['Unnamed: 0'], axis=1)

# Reading in the PV prediction data.
df = pd.read_csv('PV predict.csv', encoding='latin-1')
# Dropping the date and other unnecessary columns now, to reduce the size of the
data frame and increase the processing speed.
df = df.drop(['Beam Irradiance (W/m^2)'], axis=1)
df = df.drop(['Diffuse Irradiance (W/m^2)'], axis=1)
df = df.drop(['Ambient Temperature (C)'], axis=1)
df = df.drop(['Wind Speed (m/s)'], axis=1)
df = df.drop(['Plane of Array Irradiance (W/m^2)'], axis=1)
df = df.drop(['Cell Temperature (C)'], axis=1)
df = df.drop(['DC Array Output (W)'], axis=1)

# Clustering into Half an hour data.
count = 0
length = int(len(data['Prediction'])/2)
# The list to store the hourly average load value.
halfHourlyLoad = list(np.empty(length))
halfHourPV = list(np.empty(length))
# The following loop converts the 15 minute data into hourly data by calculating the
average of every hour.
for i in range(0,len(data['Prediction']),2):
   halfHourlyLoad[count] = (data['Prediction'][i]+data['Prediction'][i+1])/2
   count = count + 1

count = 0
for i in range(0,24):
   halfHourPV[count] = df['AC System Output (W)'][i]
   halfHourPV[count+1] = df['AC System Output (W)'][i]
   count = count + 2
# Power required by the load after Solar.
deficitPower = np.array(np.zeros(24))
deficitPower = (np.array(halfHourlyLoad) * 200) - np.array(halfHourPV)
```

```
# The peak hours of the day are from 1 pm to 8pm. Thus the intervals pertaining to
this are from interval 25 through 38.
batRequirement =  deficitPower[26:40]
# Stores the watts the battery needs to supply in the half an hour interval.
batSize = np.zeros(14)
for i in range(0,14):
    if(batRequirement[i]>0):
        batSize[i] = batRequirement[i]
```