MICROGRID ANALYTICS FOR GENERATION AND LOAD FORECASTING

A PROJECT REPORT

Submitted by

S S ROHIT (1BM15EE044)

SONAL VARSHNEY (1BM15EE053)

SUNIDHI (1BM15EE057)

in partial fulfilment for the award of the degree

of Bachelor of Engineering

in

ELECTRICAL AND ELECTRONICS

under the supervision of

Dr Prakash D B

Assistant Professor, BMSCE

B.M.S. COLLEGE OF ENGINEERING

MAY 2019

B.M.S. COLLEGE OF ENGINEERING

BENGALURU 560019

BONAFIDE CERTIFICATE FROM SUPERVISOR

Department: Electrical and Electronics

Candidates Degree Registered for: BE

Candidate Details:

Sl. No.	Student Name	USN	Signature
1.	S S Rohit	1BM15EE044	
2.	Sonal Varshney	1BM15EE053	
3.	Sunidhi	1BM15EE057	

Certified that this project titled "Microgrid Analytics for Generation and Load Forecasting" has been carried out by the candidates to my satisfaction. The final year dissertation report was thoroughly scrutinized and corrected by me. All the corrections have been incorporated by the students. The work is original and the project report is the final one of high standard. I duly certify the same.

(Signature)
Dr P Meena
Head of Department
Electrical and Electronics
B.M.S. College of Engineering

(Signature)
Dr Prakash D B, Supervisor
Assistant Professor
Electrical and Electronics
B.M.S. College of Engineering

Date:

Seal:

(Signature of External Evaluator)

ACKNOWLEDGEMENTS

We express our deep sense of gratitude to our institution for providing us with the opportunity to undertake this project. It was an excellent experience to work on data analytics for Cares Renewables.

We would also like to convey our gratitude to Dr Prakash D B, Assistant Professor, Department of Electrical and Electronics, BMSCE, who guided and encouraged us throughout, providing knowledge of the project. We are thankful to Dr P Meena, Head of Department, Department of Electrical and Electronics, for helping us reach out to the industries and providing her valuable knowledge and time.

We would extend our special gratitude to all the teaching staff of the Department of Electrical and Electronics, BMSCE, who have been with us throughout our courses and have imparted valuable knowledge which has helped us with the project.

We would like to thank Dr B V Ravishankar, Principal, BMSCE and Dr Ravishankar Deekshit, Vice Principal, BMSCE for providing us this platform and supporting us throughout the curriculum of this project.

This project provided us with a wonderful platform to expand our academic knowledge in the field of Data Analytics and Data Science. It allowed us to grow technically and professionally.

ABSTRACT

Micro-grids are a rapidly growing sector of smart grids which will be an essential component in the trend towards distributed electricity generation. It becomes essential for the providers of such services to give an insight of the advantages of installing solar panels and other renewable sources and assure the customer of the returns. It also becomes important for the supplier to analyse the performance of the panels if installed in a particular coordinate and the load requirements of the customer. This helps the service provider suggest suitable power ratings that can be installed to the customer.

The prediction of power generation is a complicated task for the micro-grids since most DG (Distributed Generation) sources are renewable energy sources whose power generation varies largely with external conditions like sunshine, temperature etc. Forecasting renewable generation is a challenging task and its relevance increases rapidly with more penetration of renewable energy sources in the power grid.

Given the rise of smart electricity meters and the wide adoption of electricity generation technology like solar panels, there is a wealth of electricity generation data available. With the advent of Artificial Intelligence (AI) and Data Science, the wealth of data available can be routed, collected, organised, stored and utilised in a thorough manner. Using various AI techniques, the data collected can be processed and the results obtained can be utilised to get insights on the performance of the system and assist in making the system robust. The service provider can provide an application to monitor demand and generation. The consumer can visualise this data and can get predictions of the load requirements for the next few hours.

In this project, a Data Analytics based forecasting model for power generation prediction of solar power plant and load side prediction of the consumer has been proposed. Short-term forecasting over an hour or a day requires non-linear predictive models. Machine learning algorithms such as neural networks are inherently non-linear and are suitable for accurate forecasting. Various algorithms such as Neural Networks, Decision Trees, Conditional Restricted Boltzmann Machines (CRBM) and Long Short-Term Memory (LSTM) for forecasting short-term demand have been compared.

CONTENTS

Part II **Generation Side Analytics**

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Machine Learning

Machine Learning is an application of Artificial Intelligence (AI) that provides system the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access the data and use it to learn for themselves.

To predict power generation from past data using machine learning, various methods of Machine Learning can be employed.

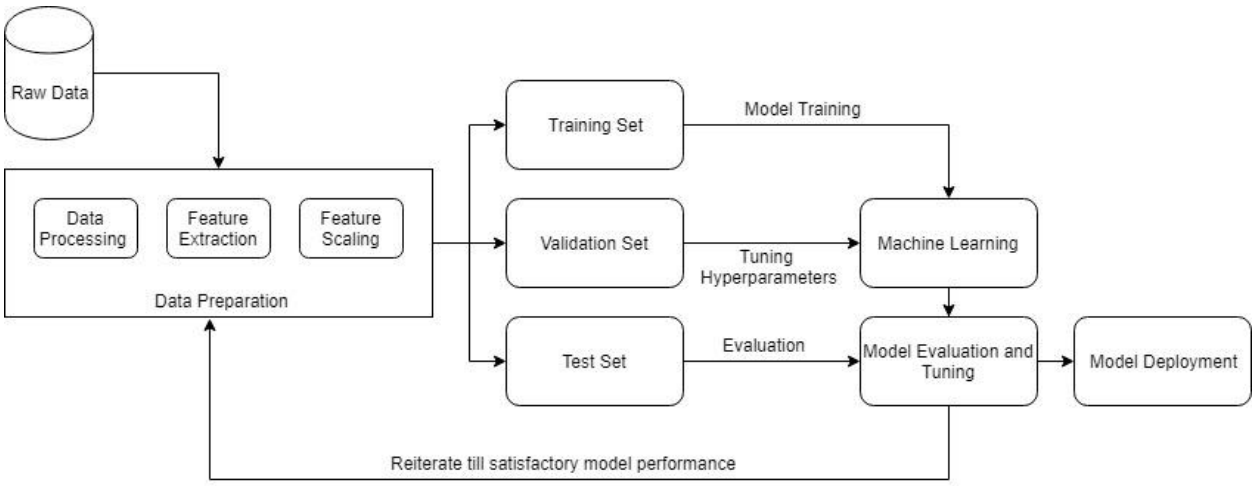

Figure 1.1 Block Diagram of Machine Learning

Data Pre-processing

Data pre-processing is an important step in machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

The input data being fed to the system is raw and has to undergo preliminary processing before the model is implemented on it. For example, sometimes, neural networks fail to predict time-series data and provide erroneous predictions. In such cases, the data has to be converted to a suitable format so that the model can operate on the data.

Data pre-processing includes cleaning, instance selection, normalization, scaling, feature extraction, etc.

Since the dataset being worked on is time-series data, it is necessary to have continuous data in order to produce accurate results. Unfortunately, there are gaps in data, irregular time steps of recordings, or removed data points that often need to be filled for data analysis. Interpolation is one such technique for statistical models useful for filling in missing data. It denotes the process of computing the intermediate values of a function from a set of given or tabular values of that function. Linear interpolation, a very simple form of interpolation, is basically the rendering of a straight line between two or more points. However, it is not very precise. Polynomial interpolation is a generalised form of linear interpolation. If a set of data contains *n* known points, then there exists exactly one polynomial of degree *n*-1 or smaller that passes through all of those points. The polynomial graph can be thought of as filling in the curve to account for data between the known points. This methodology is polynomial interpolation.

Model Implementation

An implementation of a given data model is a physical realisation of the components that constitute that model. There are various models in machine learning that can be used for prediction. Statistical models like Logistic Regression and Autoregressive Moving Average (ARMA) and AI based models such as Neural Networks and Long Short-term Memory (LSTM) are used. The model to be used for a certain data depends on the type and format of input data and the nature of prediction to be done by the model.

Hyperparameters express higher-level properties of the model such as its complexity and rate of learning. These parameters are generally set before the training process begins as they cannot be learned. These parameters are determined by experimentation considering the loss as the objective function. The number and diversity of hyperparameters is specific to each model. They are tuned in order to make the model more optimal.

The dataset is split into two parts - train and test. The train data is generally 80% of the total data. The test data is the remaining 20%. Note that since the data being dealt with are time-series data,

they have to be at continuous intervals of time. During the process of training, the data is split into train set and a validation set to determine the loss over every iteration. This loss is reduced after every iteration using an appropriate optimizer function such as the gradient descent.

Evaluation

The main objective function behind these models is to reduce the mean absolute error in order to obtain the best accuracy. Mean Absolute Percentage Error is a measure of prediction accuracy and is also used as a loss function in many models. It is given by the formula:

$$\mathrm{M} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

With each iteration, the Mean Absolute Error should reduce to obtain optimal solution.

The test data is predicted. For data validation, mean average errors of the actual data and predicted data are calculated and compared. Plots of these errors, input test data and predicted test data are obtained. These graphs are then analysed and inferences are made. Conclusion inferred from the graphical output are fed back to the model by making suitable changes to various model parameters in order to optimise the solution.

1.2 Project Scope

In this project, a Microgrid analytics-based approach for predicting power generation of a solar power plant along with its load side demand has been implemented. Over the past few years the demand for renewable energy has surged and with the advent of cutting-edge technologies such as Artificial Intelligence (AI) and Machine Learning (ML), has enabled us to improve the efficiencies of the working of these plants.

These technologies have made use of in order to augment the existing electrical technologies of power converters. Using these techniques, it is desired to predict the generation of power from solar and other renewable sources. It is also desired to develop models which can predict load side demand. Various models include statistical approach such as the ARMA (Autoregressive Moving Average) and ARIMA (Autoregressive Integrated Moving Average), regression-based approach that include linear and logistic regression and finally the AI approach that include ANN (Artificial

Neural Network), CRBM (Conditional Restricted Boltzmann Machine) and the LSTMs (Long-Short-Term Memory).

The first part of the project deals with the load side analytics where the input data used is the power consumption at a given instant of time. Based on this previous data, it is possible to predict the future consumption on an hourly or daily basis. The first step involves determining a model which provides the most optimal and accurate results. In order to do so, a comparative study has to be carried out with the existing models, which have been further explained.

The second part of the project is the generation side analytics with input data as previously generated values of power and also the various factors affecting power generation such as irradiation, humidity and wind speed. The visualisation of the relationship between these factors and the generation is done using libraries such as matplotlib and orange. Using this, the future generation can be predicted.

The third part is the development of the web application to provide a graphical user interface. The focus is to develop a website for service providers and consumers that is interactive and provides information in very simple visual format. A bunch of RESTful APIs have been developed where service providers and consumers can integrate their system with the developed backend, thereby reusing this framework for deployment and visualisation.

From the existing literature, a comparative analysis has been done in order to validate the final approach for the data-sets.

1.3 Tools and Libraries Used

In order to carry out this research and development, a number of tools and packages have been utilized. Most of these tools are either in Python programming language or JavaScript. The packages used are split into two parts

- For data synthesis, analysis and machine learning application
- For model deployment onto a server, for package management, version control and GUI (Graphical User Interface).

Data synthesis is the process in which the dataset is analysed and pre-processed by conducting a number of evaluations on the model, such as NaN detection and interpolation. For this process,

numpy and pandas have been used for Python as libraries. Also, these two libraries play an important role in the process of data engineering.

The next set of libraries include building the model by training it on the synthesised dataset. For this purpose, keras running a TensorFlow backend has been used. The difference between keras and TensorFlow is that TensorFlow is coded using symbolic style and hence, the graph is built only at compile time, whereas keras is a wrapper with high level API and follows imperative style of programming. Symbolic style is more efficient when compared to imperative style due to its run time reusability, whereas imperative style is more flexible for building models.

To visualize the data, matplotlib is used and its GUI version is utilized to display interactive plots. For the purpose of visualisation, Chart.js - a wrapper around JavaScript - has also been used.

Once the model is developed and tested to work with desired accuracies, it becomes important to deploy the model so that it can be utilized for production environment and finally, sold as a product. The technology stack is developed using Docker Containers, making it easy for deployment purpose. For the backend, Flask is used, which is a Python framework with REST enabled APIs. It is important to understand the usage of Flask in the tech industry. Big companies, such as Netflix and YouTube, use Python powered Flask for their backend to handle huge amounts of traffic along with plethora of AI models running a number of analysis for user interface development. The RESTful APIs that are exposed enable users/third party systems to interface with the developed system to deploy and view useful data. The model is deployed using keras serving. As soon as API call/ HTTP request is made, the data from the server is loaded into the database (NoSQL). This data is then passed through the model for future predictions. The model evaluates the predicted output and finally returns the JSON response that is displayed in the front end. For implementing this, MongoDB is used as database. For data representation and front end, React.js is used and for building visualisations and charts, Chart.js is used.

CHAPTER 2
LITERATURE SURVEY

2.1 Survey based on Research Papers

In the world of Data analytics, there is a plethora of models that provide varied results when subjected to multiple constraints due to the variety of data that is available. There are many models specifically for time series prediction. Different kinds of data provide different results when applied to different models.

It becomes necessary to conduct a comprehensive literature survey before diving into model selection. The base paper for this research is "Short-term electrical load forecasting using predictive machine learning models" [1]. This paper mainly deals with prediction of load side demand using mainly three approaches viz. ANN, CRBM and Decision Trees. The dataset used is of an institutional building with typical loads like fans, AC, computers, lighting and machines such as motors. The forecasting using these models was done on MATLAB.

Also, it becomes important to understand the effects of various external factors such as wind humidity and human needs. This has been explained in the paper "Factor Affecting Short Term Load Forecasting" [2]. In this paper, it can be seen that for time-series prediction, weather data is very essential. Weather is the most important independent variable for load forecasting. Factors like temperature, humidity and precipitation influence the use of electrical appliances, hence affecting the load requirement.

Short-term load forecasting at the load level needs to account for a range of variations in the data. This can be achieved only by adopting non-linear models as the data tend to show non-linear patterns. Popular time series forecasting models include ARMA, ARIMA and machine learning models include Neural Networks and Support Vector Machines. This can be seen in "Power load forecasting using support vector machine and ant colony optimization" [3].

"Short-Term Load Forecasting: Similar Day-Based Wavelet Neural Networks" [4] explains the requirement of non-linear models for non-linear patterns exhibited by the data. Non-linear models like ANN were implemented in this paper and it could be seen that these models were effective and accurate.

Deep learning algorithms, including Neural Networks and CRBM, have been employed in the recent times to achieve more robust models that are sensitive to weather and patterns in the existing data. In “Deep learning for estimating building energy consumption” [5], CRBM is used for load side forecasting for a building. To evaluate the models, Root Mean Squared Errors (RMSEs) of various models like CRBM and FCRBM (Factored CRBMs) were compared. From this paper, CRBM model has been implemented in this project.

From “Reinforcement learning with long short-term memory” [6], it can be observed that quantities with long-term dependencies cannot be predicted by non-Markovian RL. Since the data in this project has seasonal variations, Markovian RL was another option. There are methods of using Reinforcement learning with LSTMs to predict future data and are very robust as they continuously learn from the input data. LSTM model was implemented in this paper which was incorporated into this project.

LSTM architecture would allow the predictions to depend on information from long ago. The model-based system could then learn the mapping from (inferred) environmental states to actions, as in the Markovian case, using standard techniques such as Q-learning. These were the observations made from “Reinforcement learning with hidden states” [7].

For generation side model implementation, there are number of literature surveys that have been conducted. Based on the model proposed in the paper titled “Neural network-based estimation of maximum power generation from PV module using environmental information” [8], a model for predicting solar radiation using temperature, humidity wind speed and wind direction has been designed. From the paper titled “Solar Generation Prediction using the ARMA Model in a Laboratory-level Microgrid” time series modelling has been implemented.

From the paper titled “Forecast of hourly average wind speed with ARMA models in Navarre (Spain)” [9], the modelling of ARMA and its autocorrelation and partial autocorrelation methods have been utilized for the dataset utilised for the project. In this paper, they have used ARMA model for prediction of wind speed. The error considered for comparison in this paper is Root Mean Squared Error (RMSE). The same method has been utilised for comparison of models for generation side predictions. The method of identifying the best model among ARMA has also been implemented from this paper for the dataset used in this project.

From the paper “Smart energy management system for optimal microgrid economic operation” [10], it was possible to analyse various Microgrid generator elements and the proposed models for implementing solar generation prediction. The major constraints for generation side prediction,

such as the maximum output power limit, cost effectiveness and the state of charge limits were analysed in this paper.

Taking into account wind power as a constituent of the Microgrid, system variables were extracted from the paper titled “Current methods and advances in forecasting of wind power generation” [11]. This will enable further research and development into to the field of Microgrid analytics. There are two methods which can be used for wind power forecasting. The first method is based on analysis of historical time series of wind and the second method uses forecasted values from a numerical weather prediction (NWP) model as an input. The various steps involved in wind power forecasting were explained in detail. This paper has built a strong foundation for further research in the field of Microgrids.

In order to bring the system to production level, the paper titled “A study of the relationship between weather variables and electric power demand inside a smart grid/smart world framework” [12] gives information between the various correlation factors that enabled to build the framework of the model developed in this project. In this paper, the effect of precipitation, air temperature, relative humidity, average wind speed, average wind direction and pressure on the load curve were studied and it was found that these parameters largely affect the usage of power. It was also observed that the day being a working day or a holiday also played an important role in determining the load curve. From this, a few aspects could be taken into consideration and parameters like relative humidity, wind speed, wind direction, solar irradiation and air temperature could be incorporated in this project.

The paper titled “Short-term load forecast of microgrids by a new bilevel prediction strategy” [13] enabled to implement short term radiation forecasting and has also given a pathway for further research into load side predictions.

As the amount of data increases, the features that can be extracted from the data are more and hence, more robust and sensitive models are needed to make faithful predictions. This is where deep learning comes into picture. LSTMs have proven to be very robust and also producing accurate results even with the presence of a lot of variations in the data. This was observed from the paper titled “Applying LSTM to time series predictable through time-window approaches” [14]. The implementation of LSTM in this project to predict load curve was based on this paper. As the dataset available was large, LSTM, by prior literature survey, was proven to be more appropriate and suitable for the data collected for this project.

LSTM has not been employed in any of the research papers for load forecasting, although similar datasets have been worked on. Similarities between the previous datasets on which LSTM was implemented and the data collected for this project prompted the use of LSTM for load forecasting.

2.2 Industry Interaction

As the main objective of this project is to assist the service providers and their users on matters related to generation using renewable sources and load side requirements, industry interactions were crucial. It was essential to get insights about the GUI and technology requirements from some of the renowned service provider.

A number of interactions with the industry throughout the course of the project proved very resourceful. The data for generation was provided by Cares Renewables which is a clean technology-based consultancy service provider. Their systems are installed on the rooftop of B.M.S. College of Engineering. The generation data is available online at Enphase Energy which provides a data warehouse for such systems. Industry experts gave insights to the various existing methods of solar generation prediction. They also suggested the use of deep learning techniques as they are less immune to variations as compared to statistical models and additionally, they are sensitive to seasonal data.

The next industry interaction was with Aparna Renewables. They mentioned the requirement for a GUI to make it easier for the service provider to visualize the prediction curves. They gave the requirements in GUI for the deployment of models. This GUI-enabled application would help customers predict the generation and match it with the load, thereby assisting the user in making an estimate about cost.

CHAPTER 3
DESCRIPTION OF MODELS

3.1 Logistic Regression

Logistic Regression is a supervised classification problem and is generally conducted when the dependent variable is dichotomous (binary). It is a predictive analysis used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic Regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

In order to map predicted values to probabilities, sigmoid function is used. It maps any real value into a value between 0 and 1. It is mathematically and graphically described by $S(z)=1/(1+e^{-z})$.

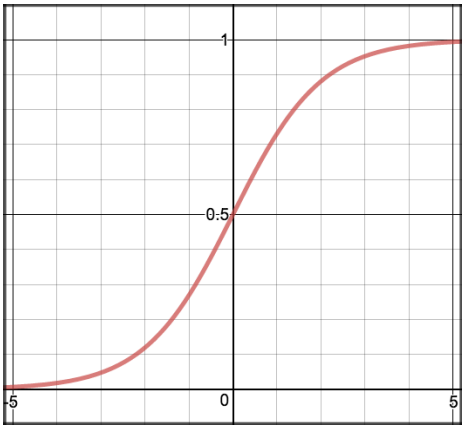

Figure 3.1 Sigmoid Function

The current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class, a threshold value or tipping point is selected above which all values are classified into class 1 and below which all values are classified into class 2 hence creating a decision boundary. Using the determined sigmoid function and decision boundary, a prediction function can be written. Gradient Descent is used to minimise the cost function and final step is to assign class labels to predicted probabilities. Logistic regression is a linear method, but the predictions are transformed using the logistic function.

3.2 Neural Networks

Neural networks are a powerful algorithm used to learn complex nonlinear functions to fit the given data. A neural network takes an input, passes through multiple hidden layers and outputs a prediction based on the combination of all the inputs. These networks are trained iteratively to achieve optimal solutions by techniques like Gradient Descent. In each cycle of training, an error metric is calculated between the actual and predicted result which is then back propagated through the network. Each neuron's weight is then adjusted relative to how much they contribute to the total error. The process is repeated till the network error drops below an acceptable threshold.

Neuron

A neuron takes a group of weighted inputs, applies activation function according to the data and returns an output.

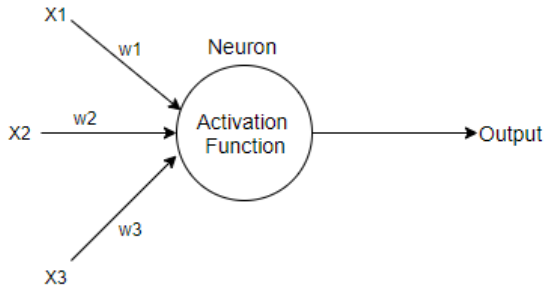

Figure 3.2 Neuron

Inputs to a neuron can be features of the dataset or the outputs of previous layer neurons. The neuron then applies an activation function to the sum of the weighted inputs and then passes on the result to all the neurons of the next layer.

Layers

The input layer holds all the data the model trains on. Each neuron represents a unique attribution of the dataset. There are hidden layers between the input and output layer where the activation function is applied. There can be multiple hidden layers. Traditionally, hidden layers are fully-connected layers where all neurons of the previous layer are connected to all the neurons of the next layer. The final layer of the network returns an output based representing the model's prediction.

Activation Functions

Activation function decides whether a neuron should be activated or not by calculating weighted sums and further adding bias. It is used to add non-linearity to the output of a neuron to make it capable of learning more complex tasks. Some common activation functions are ReLU (Rectified Linear Unit), Sigmoid and Tanh.

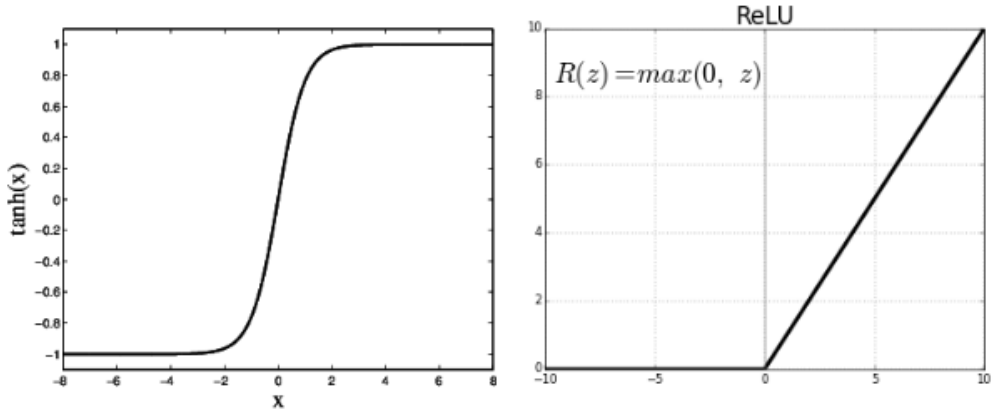

Figure 3.3 Tanh and ReLU Activation Functions

Procedure

A typical neural network is depicted in Fig 3.3 with input features denoted by x and the output as y. Weights of each connection are given by θ and the activation function by a.

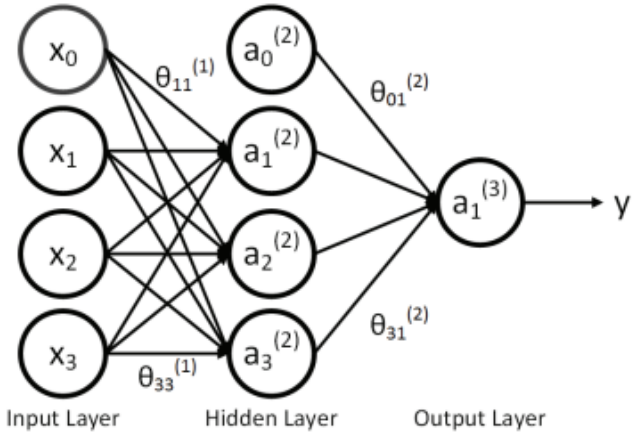

Figure 3.4 Neural Network

Forward propagation occurs when the network is exposed to the training data. The input data is passed through the network in such a way that all the neurons apply their transformation to the information they receive from the neurons of the previous layer and send it to the neurons of the next layer. When the data has crossed all the layers the final layer will be reached with a result of

label prediction for those input examples. Loss function is used to estimate the loss and to compare the prediction result in relation to the correct result. Once the loss has been calculated, backpropagation occurs. Starting from the output layer, that loss information propagates to all the neurons in the hidden layer that contribute directly to the output. This process is repeated, layer by layer, until all the neurons in the network have received a loss signal that describes their relative contribution to the total loss.

3.3 LSTM

Long-Short-Term Memory (LSTMs) are a type of neural network that can retain and remember data for a long time but also can discard (forget) data if necessary. This is done with the help of various gates which regulate the flow of information in the network.

LSTM passes data as it propagates forward. The gates regulate the flow of data by deciding which portion of the data is to be retained and which part of it is to be discarded.

A typical LSTM network comprises of multiple memory blocks called cells. The flow of data is decided by three gates - input, output and forget - that each cell has.

The inputs to an LSTM cell from the previous cell are the cell state and the hidden state of the previous cell. It also takes another input which is fresh data via the input gate. The output of the cell is decided by the output gate. The data being propagated to the next cell is the cell state and the hidden state of the current cell. A cell state acts as a highway that transports relative information along the sequence chain.

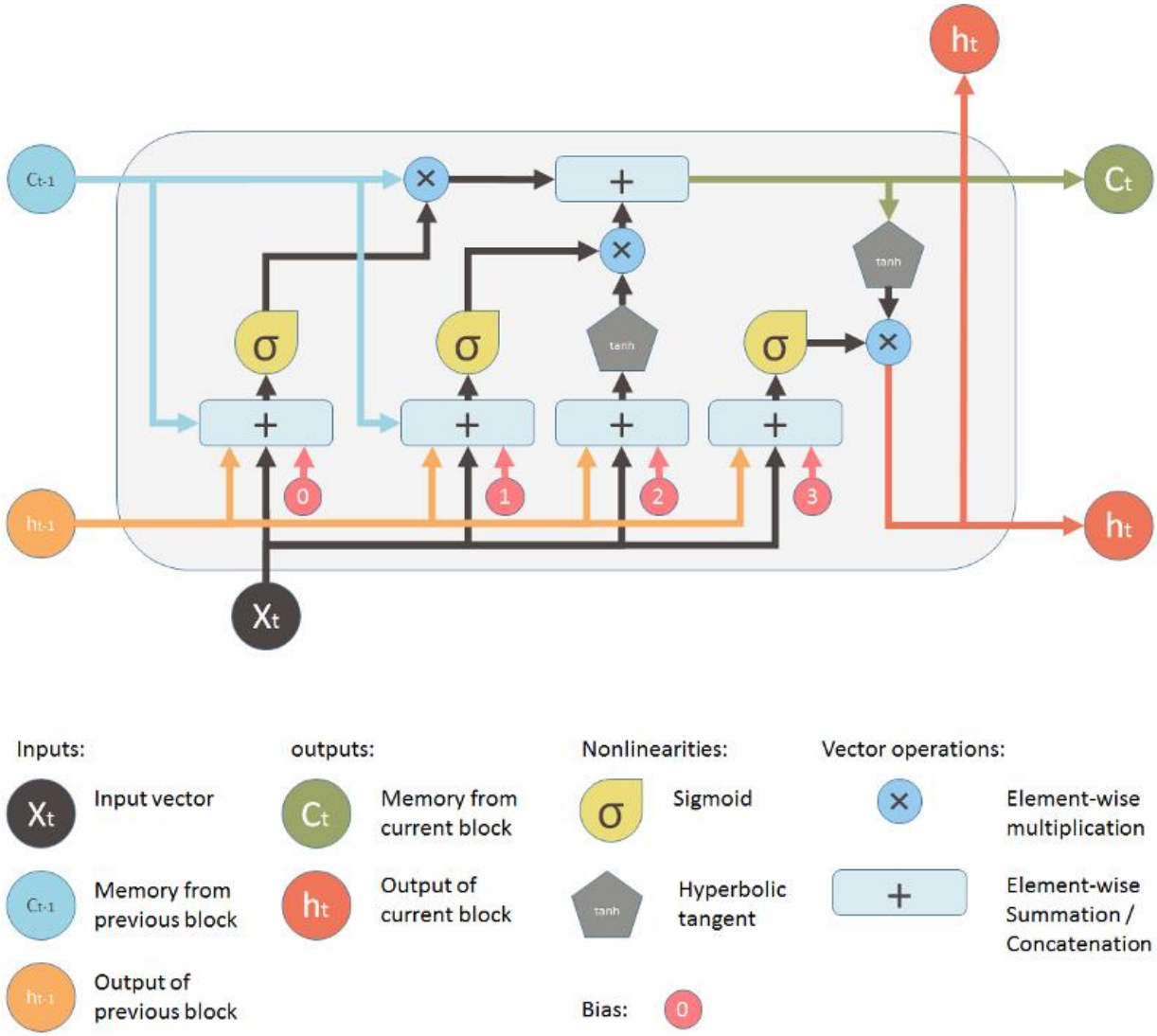

Figure 3.5 LSTM Cell

Forget Gate

The forget gate is responsible for removing unwanted data from the cell state. This is essential for optimizing the LSTM network.

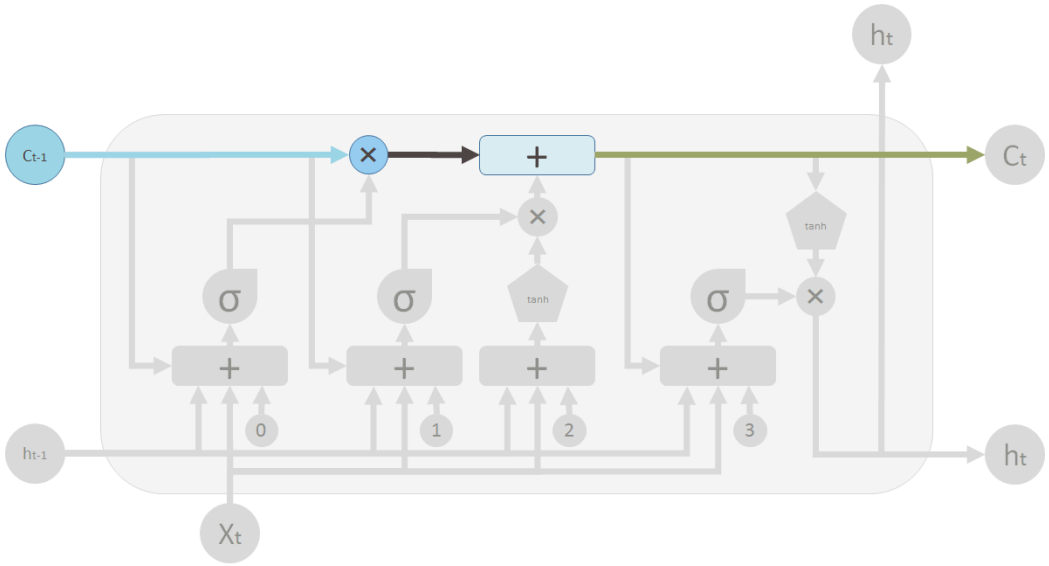

Figure 3.6 Forget Gate

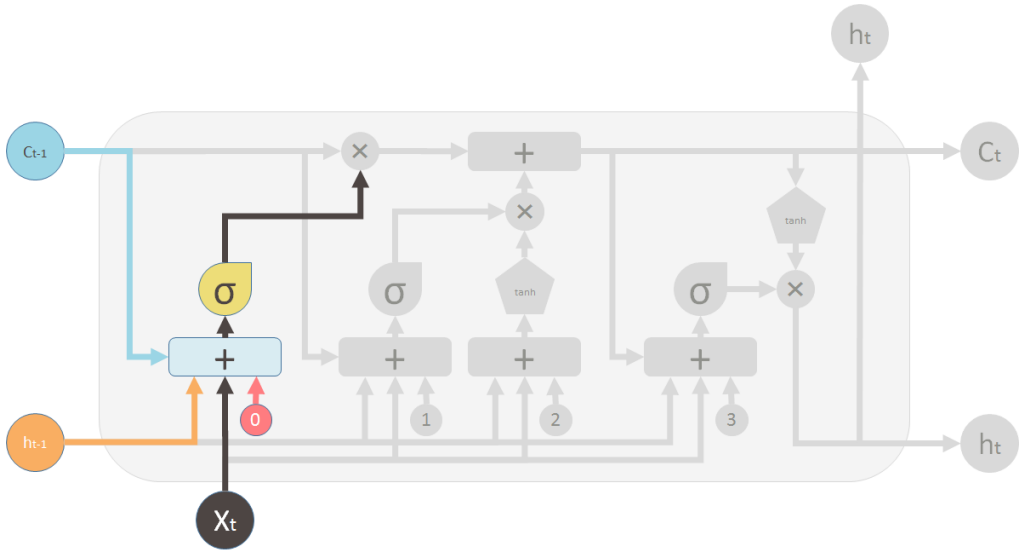

Figure 3.7 Calculation of Forget Gate Filter Value

Forget gate takes two inputs – the hidden state of the previous cell and the input to the current cell. These are multiplied with the weight matrices and bias is added. The sigmoid function is applied to this value. The sigmoid function outputs a vector that ranges from 0 to 1. This is the value that decides if the data needs to be forgotten completely (0) or needs to be retained in the cell

completely (1) or a part of the data has to be retained and the other part needs to be forgotten (between 0 and 1, excluding 0 and 1). This is multiplied with the cell state.

Input Gate

The input gate is responsible for the addition of information to the cell state. The addition of information or data to the cell state is a three-step process.

1. Regulating the value to be added to the cell by involving a sigmoid function
2. Creating a vector containing all possible values that can be added to the cell state via a tanh function
3. Multiplying the value of the regulatory filter (sigmoid gate) with the created vector (tanh gate) and adding this useful data to the cell state using an addition function.

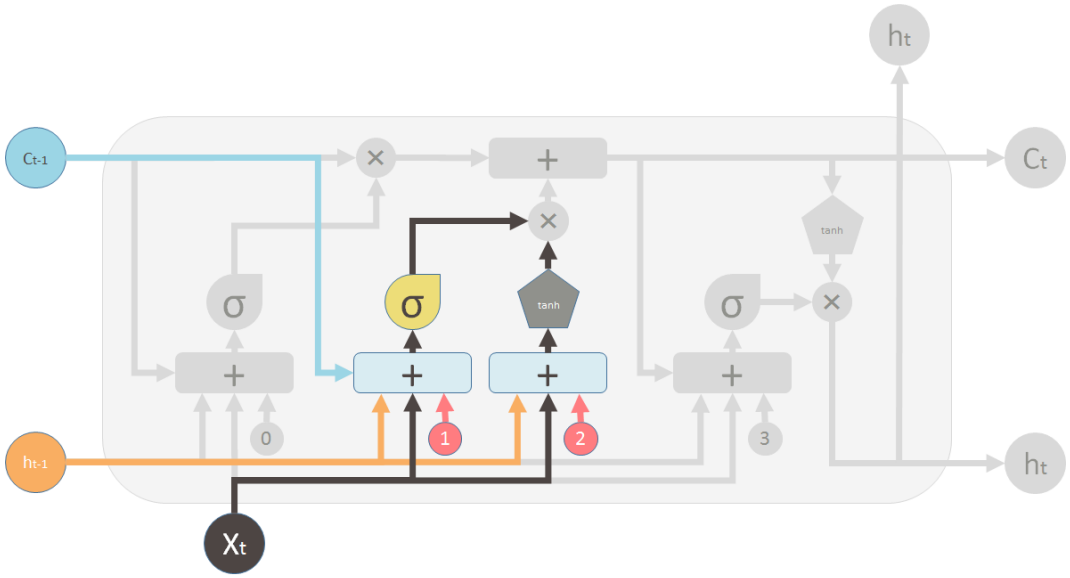

Figure 3.8 Input Gate

This ensures that only the important information is added to the cell state and the redundant data is discarded.

Output Gate

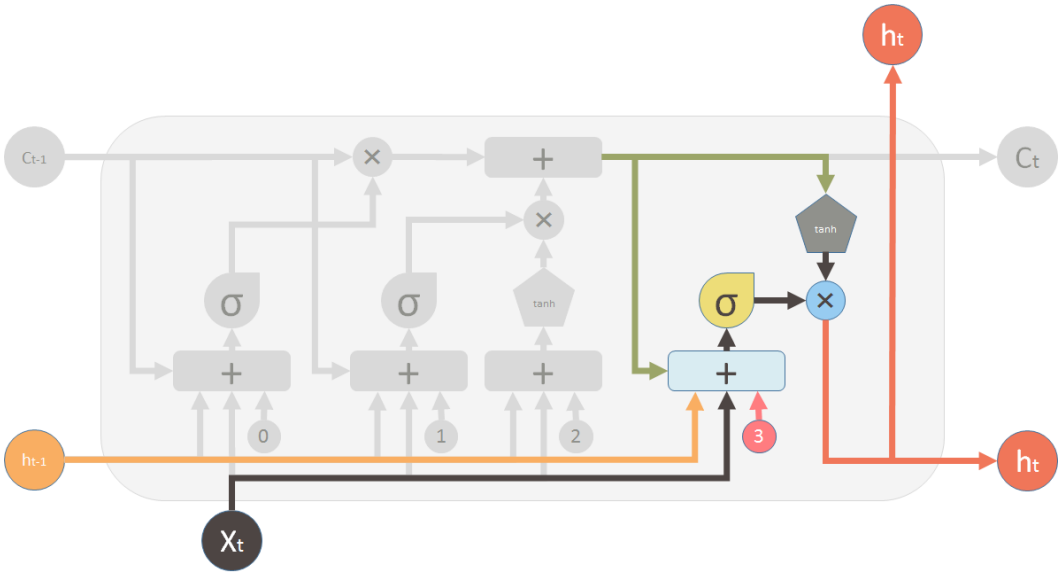

Figure 3.9 Output Gate

The output gate selects the information that had to be taken from the cell state and displayed as the output. This is done in three steps.

1. Creating a vector by applying tanh function, thereby scaling the data to the range -1 to +1
2. Making a filter using the hidden state of the previous cell and the input to the current cell such that the values that need to be output from this cell can be regulated
3. Multiplying the value of this regulatory filter to the created vector and passing it as the output of the current cell.

3.4 ARMA

Auto Regressive (AR)

Autoregression (AR) model uses the weighted sum of past values to predict future values. An AR(1) regressive process is the first order process which means that the current value is based on the immediately preceding value. It's used for forecasting when there is some correlation between values in a time series and the values that precede and succeed them. Generalising, AR(p) regressive process that predicts based on 'p' number of immediately preceding values. An AR(p) model is defined by the following equation:

$$Y_t = \delta + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots\ldots\ \varphi_p Y_{t-p} + A_t$$

where

- Y_{t-1}, Y_{t-2}… are past series values
- A_t is white noise (randomness)
- $\delta = (1 - \sum \Phi_i)\ \mu$

Moving Average (MA)

Moving Average (MA) model is a common approach for modelling univariate time-series data. It specifies the output variable depending linearly on the current and various past values with errors. Contrary to AR, MA models are always stationary. The autocorrelation function (ACF) of an MA(q) process is zero at lag q+1 and greater. Therefore, the appropriate maximum lag for the estimation is determined by examining the sample ACF to see where it becomes insignificantly different from zero for all lags beyond a certain lag, which is designated as the maximum lag q.

MA(q) is given by the equation:

$$X_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \ldots\ldots \theta_q\varepsilon_{t-q}$$

where

- μ is the mean of the series
- $\theta_1 \ldots \theta_q$ are the parameters
- $\varepsilon_{t-1 \ldots} \varepsilon_{t-q}$ are white noise error terms.

The ARMA Model

An ARMA model, or Autoregressive Moving Average model, is used to describe weakly stationary stochastic time series in terms of two polynomials, one for autoregression model and the second for the moving average model. It is referred to as ARMA(p,q) model and the equation is given by

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}.$$

where

- φ is AR model parameters
- θ is MA model parameters
- ε is white noise error term.

Estimation and forecasting of univariate time-series model using Box-Jenkins method has three parts - model identification, estimation and diagnostic checking.

Model Identification

By using plots of Autocorrelation and Partial Autocorrelation functions of the dependent time-series, it is possible to decide which AR-MA model needs to be implemented. Autocorrelation is the correlation of a signal with a delayed copy of itself as a function of delay. The analysis of Autocorrelation Function (ACF) is a mathematical tool for finding repeated patterns or identifying missing fundamental frequency in a signal. Partial Autocorrelation Function (PACF) gives the partial correlation of a time series with its own lagged values, controlling for the values at all shorter lags, unlike ACF which does not control for other lags. Comparison of plot of sample ACFs vs lags of time-series data with theoretical ACFs and PACFs leads to selection of appropriate model.

Estimation

Estimation uses computational algorithms to arrive at coefficients that best fit the selected model. To estimate p and q of ARMA, there are various methods can be used, namely Yule Walker procedure, Method of Moments and Maximum Likelihood Method. Yule Walker method has been used in this project. It fits an AR model to the windowed input data by minimising the forward prediction error.

Model	ACF	PACF
AR(p)	Spikes decay towards zero	Spikes cut off to zero
MA(q)	Spikes cut off to zero	Spikes decay towards zero
ARMA(p,q)	Spikes decay towards zero	Spikes decay towards zero

Table 3.1 Selection of Short-Term Forecasting Model

Diagnostic Checking

Diagnostic Checking tests whether the estimated model conforms to the specification required. If the estimation is inadequate, the steps are repeated to build a better model. The prime criterion for model selection is based on AIC and BIC.

ARMA vs ARiMA

The main contrast between ARMA and ARiMA is the differencing. ARMA is a stationary model. If not, then the stationarity is achieved by taking a series of differences. The 'i' in ARiMA stands for integration. It is a measure of how many non-seasonal differences are required to make the model stationary. A model with a 'd'th difference to fit and ARMA(p,q) model is called an ARIMA process of order (p,d,q).

3.5 Decision Tree

A Decision Tree is a flowchart structure where various results are computed from the root node to the leaf node. All nodes other than the root and leaf nodes are called decision nodes. Decision trees are constructed using algorithms that identify ways to split data based on various conditions. There are mainly two types of decision trees:

- Classification type where the decision variable is categorical
- Regression type where the decision variable is continuous

Classification and Regression Tree (CART) is an umbrella term used to represent both the given procedures.

Decision trees classify examples by sorting them down the tree from the root to a leaf node which provides the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes. This top-down induction of decision trees is an example of greedy algorithm and is the most common strategy used.

Decision Tree using XGBoost

XGBoost (eXtreme Gradient Boosting) is an implementation of gradient boosted decision trees which is used for increasing speed and optimising the performance of a model. Gradient boosting

is an approach where new models are created that predict the errors of prior models. These are then added together to make the final prediction. It uses a gradient descent algorithm to minimize the loss when adding new models. XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

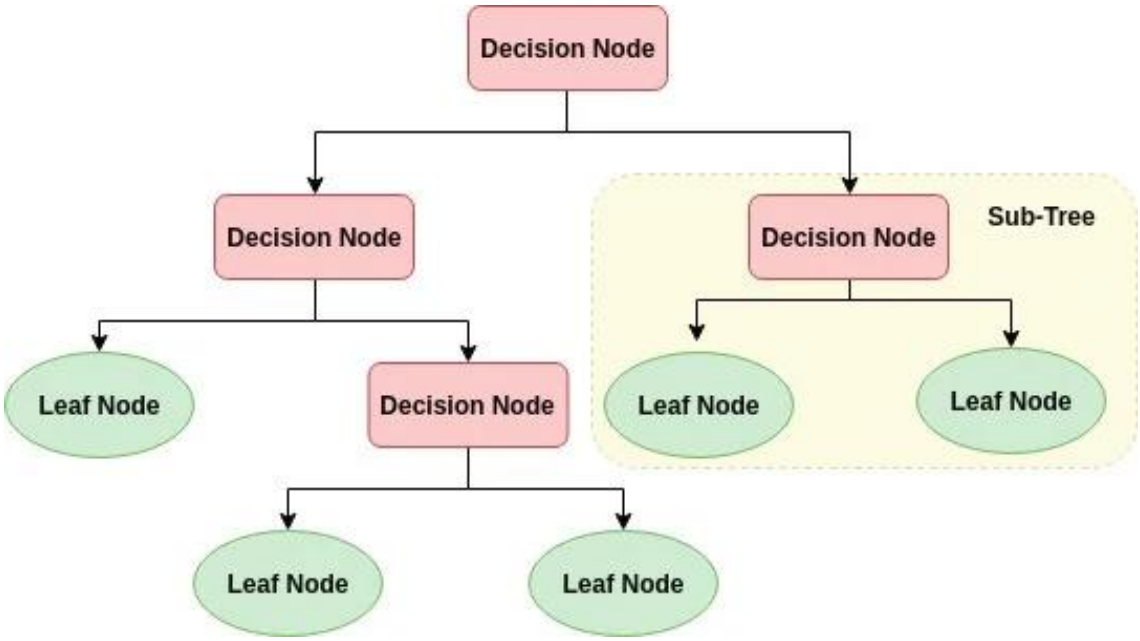

Figure 3.10 A General Decision Tree Representation

3.6 CRBM

Conditional Restricted Boltzmann Machines (CRBMs) are energy-based learning models extended from Restricted Boltzmann Machines (RBMs) used to model complex time series data. CRBMs can capture the intricate dependencies of a time series and accurately estimate the probability distribution of the input data. A Restricted BM is a parameterized generative model which represents a probability distribution. Learning a BM means adjusting the BM parameters such that the probability distribution represented fits the training data as well as possible. Boltzmann machines consist of two types of units, visible and hidden neurons, which can be thought of as being arranged in two layers. After successful learning, it provides a closed-form representation of the distribution underlying the training data.

A CRBM consists of three layers, a visible layer V, a hidden layer H and a conditional layer U as shown in Figure 3.11.

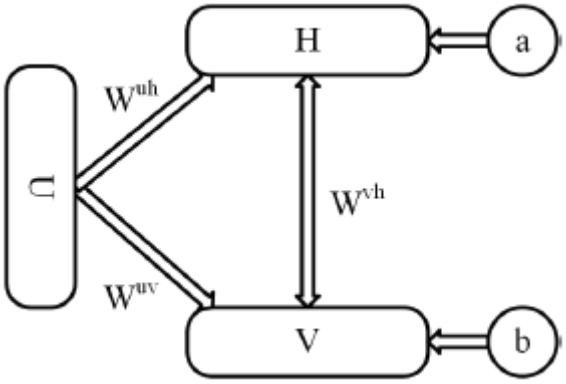

Figure 3.11 CRBM Representation

Inputs are given to the visible layer from which reconstructed outputs are obtained. The distinct features of the input data are modelled by the hidden layers. There are no connections between the nodes in the same layer, while the nodes in the visible and hidden layers are connected symmetrically through bi-directional weights, denoted by W^{vh}. Through weights W^{uv} and W^{uh}, the conditional layer gives inputs to the visible and hidden layers dynamically. a and b represent the biases to the hidden layer and visible layer respectively.

PART I
LOAD SIDE ANALYTICS

CHAPTER 4
DATA ANALYSIS AND MATHEMATICAL MODEL

4.1 Data Analysis

Load Data

The data used for the Load Side analysis requires the power demand over a certain period of time. Power consumed can either be directly obtained (active power) or calculated by scalar multiplication of the current vector and the voltage vector.

The data was collected over a period of one year from the Bosch unit that consumes the power generated by the solar grid installed in B.M.S. College of Engineering. Figure 4.1 shows the format of the data collected.

vltgA	vltgB	vltgC	currA	currB	currC	currN	F	PFA	PFB	PFC	VTHD	ITHD	sample_timestamp	eventId	eventTime
2138	2141	2131	2632	1073	2676	500	4900	86	90	90	5	22	3/18/2016 16:14	4	3/18/2016 16:14
2140	2145	2137	2720	1117	2529	529	4900	91	92	92	4	14	3/18/2016 16:17	0	3/18/2016 16:17
2131	2132	2129	3058	1382	2955	382	4900	90	92	92	5	18	3/18/2016 16:18	2	3/18/2016 16:18
2132	2133	2130	3058	1382	2926	382	4900	90	92	92	6	18	3/18/2016 16:20	3	3/18/2016 16:20
2123	2131	2128	2661	1000	2220	632	4900	92	92	92	5	12	3/18/2016 16:29	1	3/18/2016 16:29
2133	2137	2132	2545	1027	2347	496	4900	0	0	0	5	14	3/18/2016 16:29	5	3/18/2016 16:29
2153	2150	2147	1970	1014	2000	352	5000	91	92	91	4	16	3/18/2016 16:36	0	3/18/2016 16:36

Figure 4.1 Representation of Dataset for Load Side Analytics

This data represents a multivariate time series of power-related variables that in turn could be used to model and even forecast future electricity consumption. The values of phase voltages, phase currents, power factors in each phase, frequency, Total Harmonic Distortion (THD) in current and voltage are collected over a period of time.

datetime	Power (Kw)
18-03-2016 16:30	115.0825926
18-03-2016 16:45	108.8862382
18-03-2016 17:00	102.6898839
18-03-2016 17:15	97.3304694
18-03-2016 17:30	91.9710549
18-03-2016 17:45	86.6116404

Figure 4.2 Power Data for Load Side Analytics

As seen in the Figure 4.1, the data obtained is at irregular intervals. This type of data cannot be used for networks such as LSTM, as they require continuous data. Initially, the missing values or deleted values are obtained using interpolation techniques. Here, we have used polynomial piecewise interpolation. The unit which collects the load side data records eventID as 5 when the power factor is equal to 0. So, the voltage and current for eventID 5 is utilised for the calculation of power since these are the instantaneous peak values recorded. The power factor for the preceding eventID 3 is considered to obtain non-zero value. The power is then calculated and recorded in a generalised format at 15-minute intervals. The power consumption for each timestamp is calculated as

$$P = V_R I_R \cos \phi_R + V_Y I_Y \cos \phi_Y + V_B I_B \cos \phi_B$$

Weather Data

It was observed while conducting literature survey that weather plays an important role in demand for electric power. Factors like ambient temperature, solar radiation, wind speed, wind direction and relative humidity logged at finite intervals need to be introduced into the model to increase the real-time effectiveness of the model.

The weather data is collected from the weather station located at B.M.S. College of Engineering. The parameter values have been logged at 5-minute interval and the average value for 1 hour has been taken.

Report	Date/Time	7:AT	8:RH	9:WD	10:SR	11:WS
RPT2	08-01-2014 01:00	35.8	-47	31	77.5	117.5
RPT2	08-01-2014 02:00	34	-47	31	77.9	110.9
RPT2	08-01-2014 03:00	35.3	-47	31	77.4	116.5
RPT2	08-01-2014 04:00	38.4	-47	31	76.9	116.5
RPT2	08-01-2014 05:00	39	-47	31	77.6	124.9
RPT2	08-01-2014 06:00	39	-47	31	78.1	129.7
RPT2	08-01-2014 07:00	41.4	-47	31	78.1	117.3
RPT2	08-01-2014 08:00	42.2	-47	31	78.1	118
RPT2	08-01-2014 09:00	43.7	-47	31	77.5	110.8

Figure 4.3 Weather Data

AT stands for ambient temperature and the unit is degree Celsius. RH stands for relative humidity and is given in percentage. WD stands for wind direction and the unit is degrees from north in clock direction. SR stands for solar radiation. WS stands for wind speed and the unit is m/hr.

This data is interpolated and values for every 15 minutes are obtained. These data are then fed to the model as inputs.

4.2 Mathematical Model

Demand is calculated using the following equation subjected to constraints time and event ID:

$$P = V_R I_R cos\ \phi_R + V_Y I_Y cos\ \phi_Y + V_B I_B cos\ \phi_B$$

Voltage, current and power factor values of each phase are obtained from the dataset and are operated on as the given equation.

Constraints

The major constraint in this model is the eventID. The eventID 3 represents the event when voltage and current are at their peak values. But the power factor is zero at this point. Hence, if power is calculated using all the parameters from this eventID, the power obtained will be zero (formula for power is $P = V\ I\ cos\ \phi$). To obtain a non-zero value of power, the power factor of eventID 3 is considered.

Objective Function

In our model, the objective is to minimise error. There are various ways of calculating the error. In this project, we have used Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). These two need to be minimised in order to obtain an efficient system. Hence, these two are the objective functions of the mathematical model.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| * 100$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|A_i - F_i|$$

CHAPTER 5
IMPLEMENTED METHODS AND COMPARATIVE STUDIES

5.1 Implemented Methods

Models for three kinds of predictions have been implemented. Predictions are based on

- one-hour previous data
- two-hour previous data
- three-hour previous data

Depending on this, various parameters for the models are selected, such as number of nodes for a neural network or for a decision tree.

For a 24-hour period, since data is being collected at regular 15-minute intervals, there will be a dataset of 96 points.

5.1.1 Neural Networks

The designed model uses one hidden layer of optimum size. Features are extracted from the time-series data and are given as input to the nodes of the input layer.

For a three-hour based prediction, the number of input nodes required would be 12 points. The hidden layer uses over 200 nodes which is selected after the tuning of hyperparameters. This network is trained to finally obtain the power consumption. The training data is then split into a cross-validation set and a test set. This is done so that the model is not affected by high variance or bias. Bayesian Regularisation, the training method used, provides the required weights.

Results

The data input to the model was taken from 24/03/2016. Figure 5.1 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 3 hours data. It can be observed that the prediction accuracy is satisfactory. Figure 5.2 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 3 hours data input.

Figure 5.1 One hour prediction (1300 to 1400 hours) with three hours previous data (ANN)

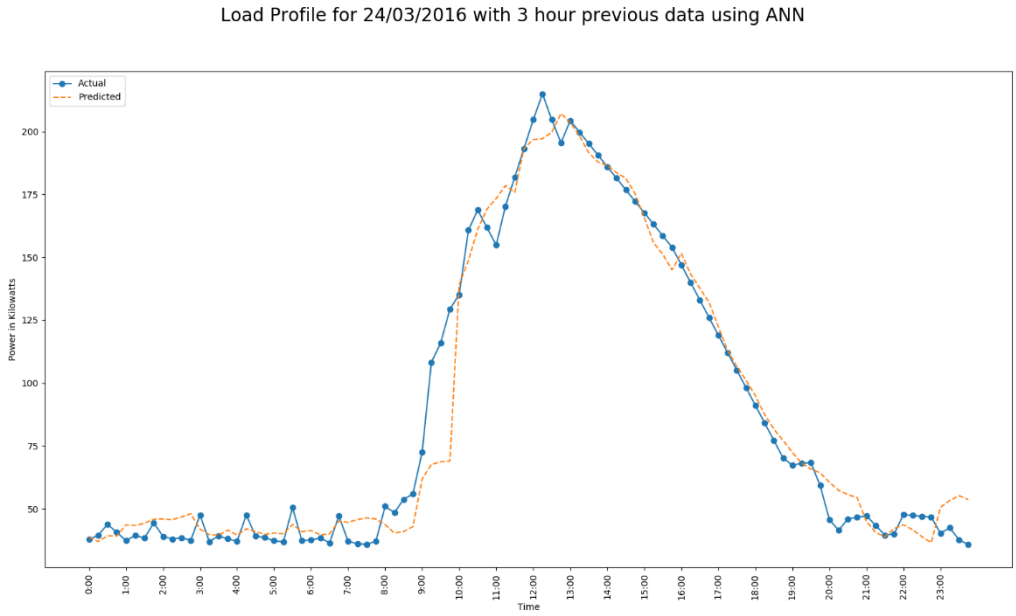

Figure 5.2 Full day prediction with three hours previous data (ANN)

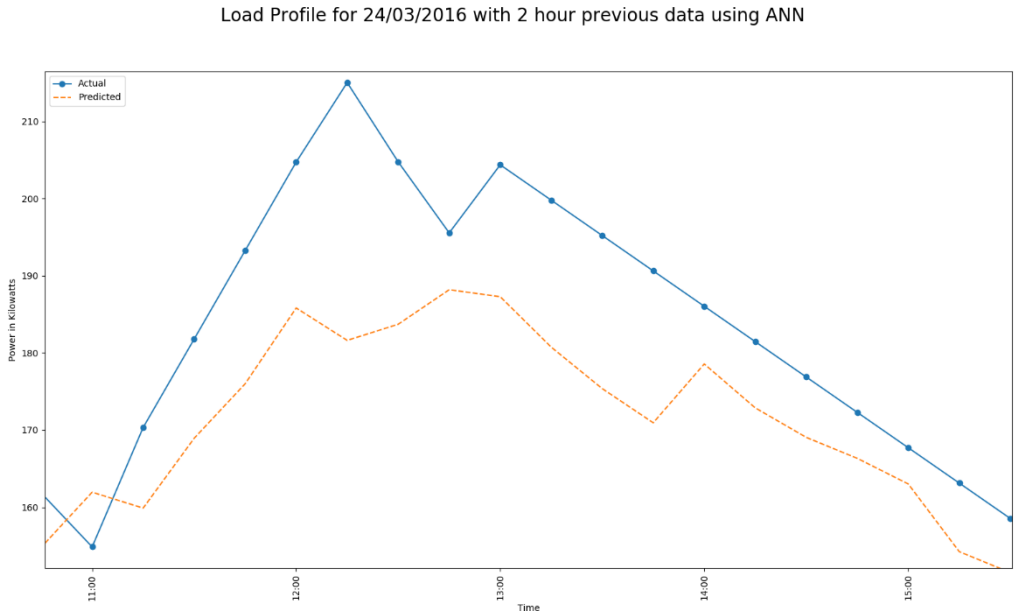

Figure 5.3 One hour prediction (1300 to 1400 hours) with two hours previous data (ANN)

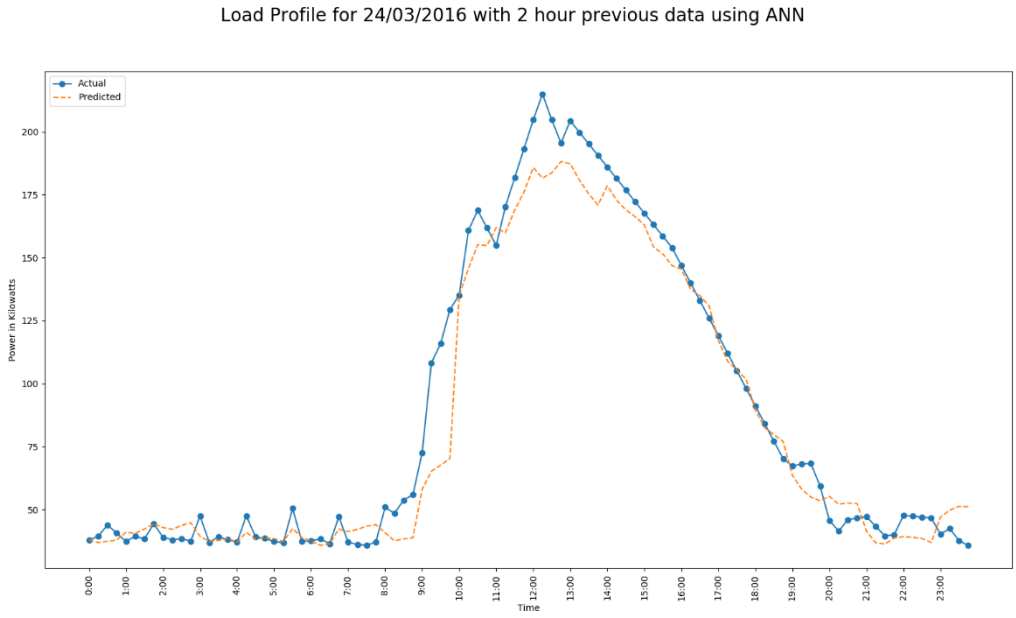

Figure 5.4 Full day prediction with two hours previous data (ANN)

Figure 5.3 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 2 hours data. Figure 5.4 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 2 hours data input.

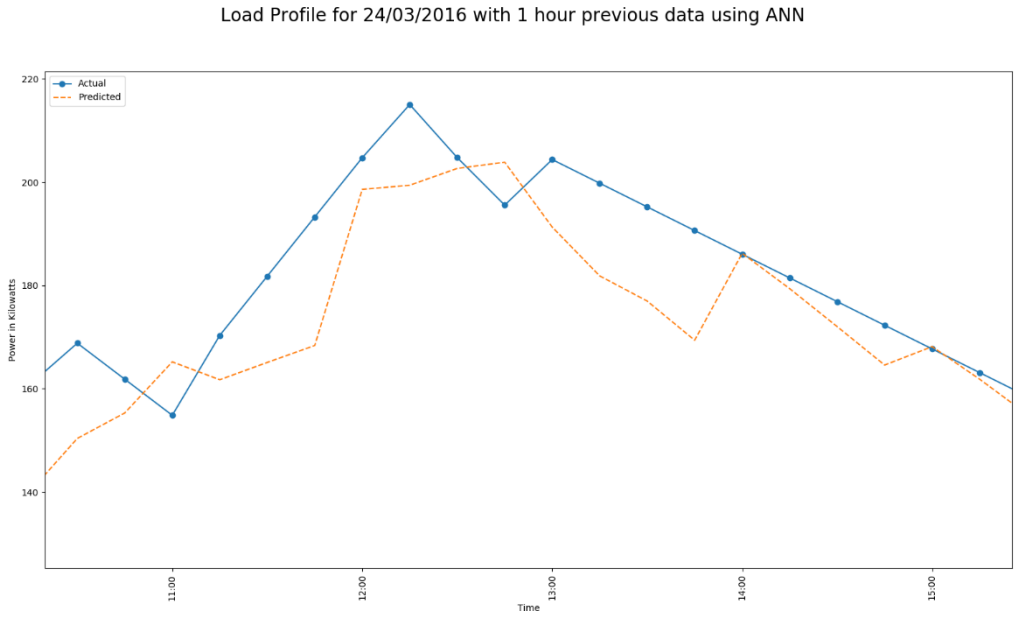

Figure 5.5 One hour prediction (1300 to 1400 hours) with one hour previous data (ANN)

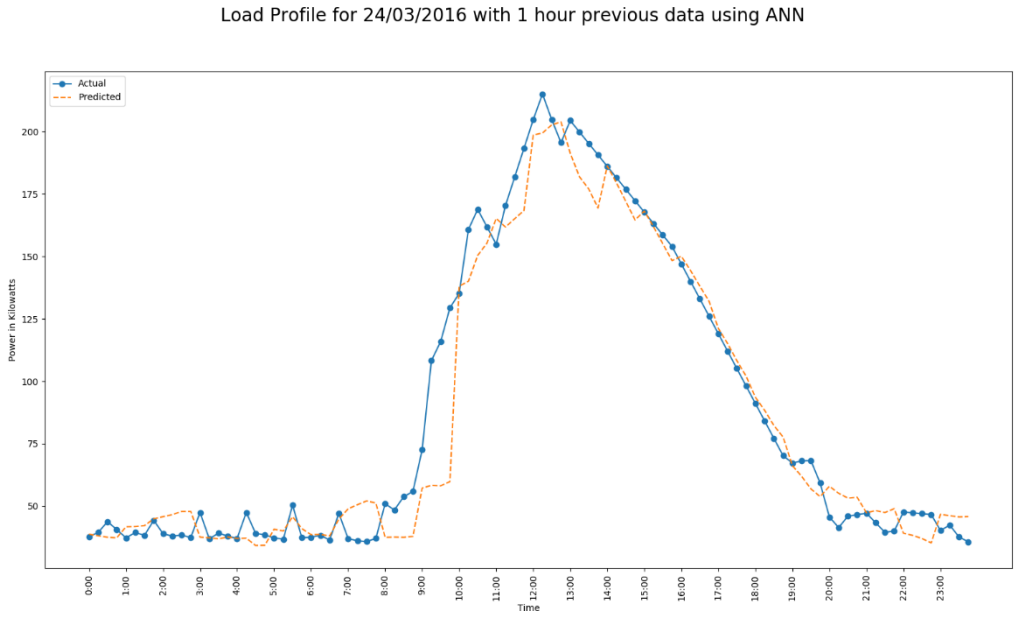

Figure 5.6 Full day prediction with one hour previous data (ANN)

Figure 5.5 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 1 hour data. Figure 5.6 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 1 hour data input.

7.1.2 CRBM

To speed up learning of this model, chunks of frames of the dataset are assembled into mini-batches. This model uses two hidden layers, each layer having up to 500 units. The conditional layer consists of autoregressive units. This layer is connected with previous five data points and the model is trained for 1000 epochs. After initialising the weights and bias, conditional probabilities are calculated using Gibbs Sampling. In this model, the feature corresponding to the power consumption during the same time and the day of the previous week is excluded as the previous data is already considered. Once the positive and negative phase of a mini-batch is complete, weight updates are calculated. This is done using contrastive divergence. This process is done for all the batches to finish the training of the model.

Results

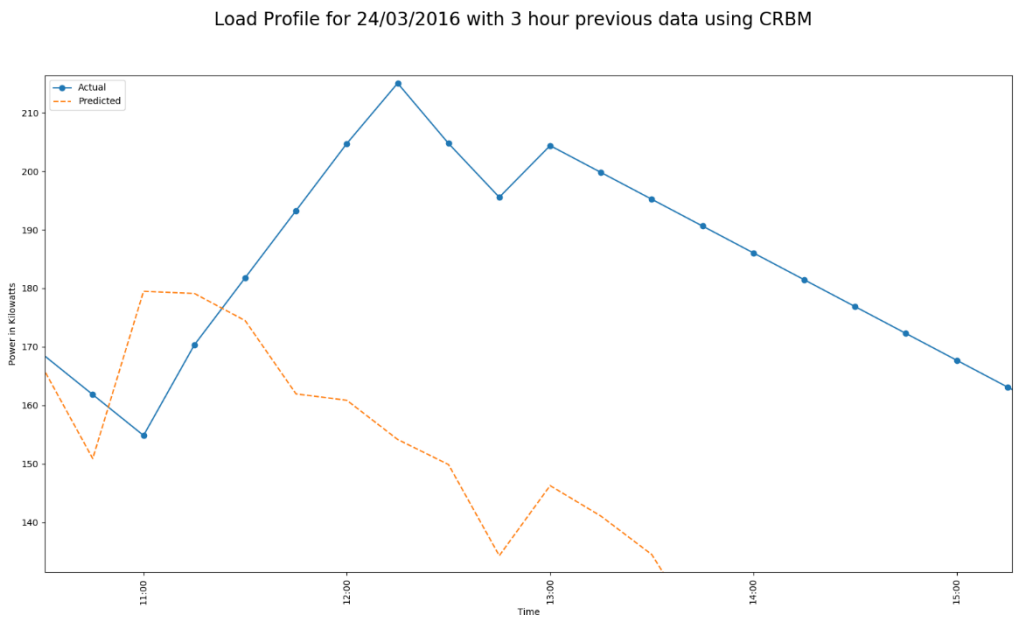

Figure 5.7 One hour prediction (1300 to 1400 hours) with three hours previous data (CRBM)

Figure 5.7 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 3 hours data. Figure 5.8 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 3 hours data input.

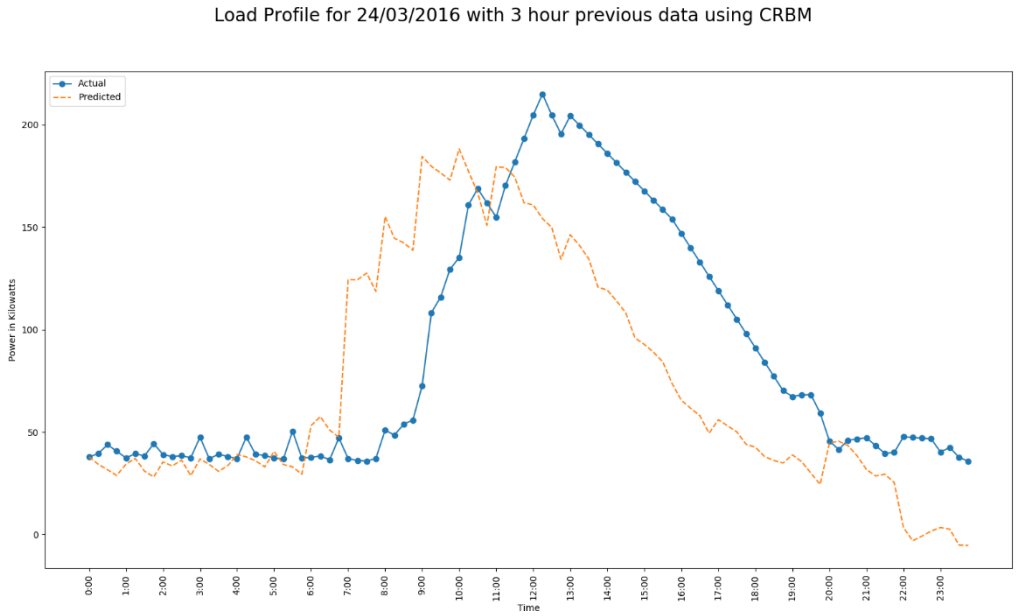

Figure 5.8 Full day prediction with three hours previous data (CRBM)

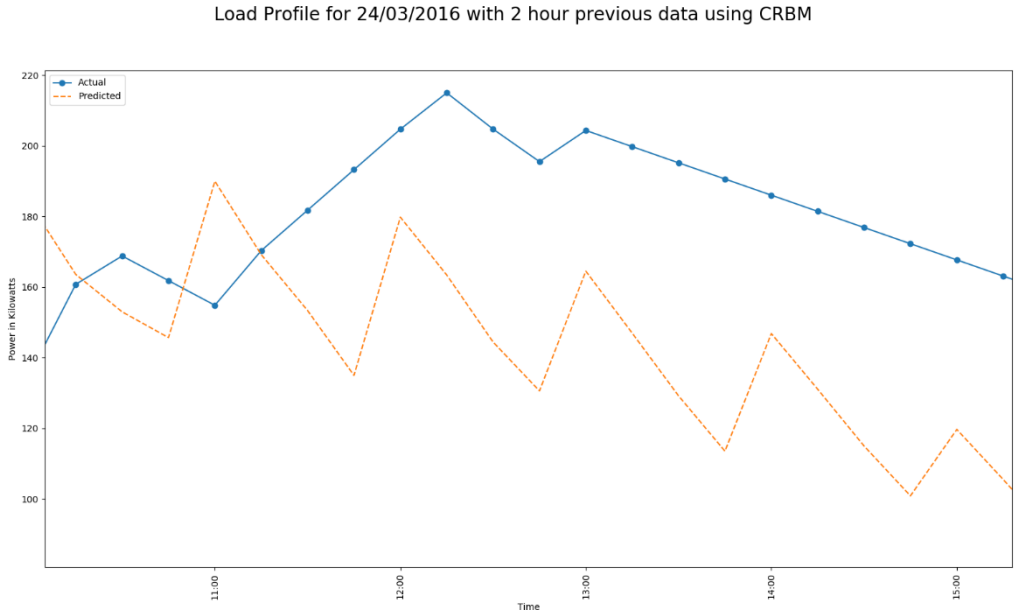

Figure 5.9 One hour prediction (1300 to 1400 hours) with two hours previous data (CRBM)

Figure 5.9 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 2 hours data. Figure 5.10 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 2 hours data input.

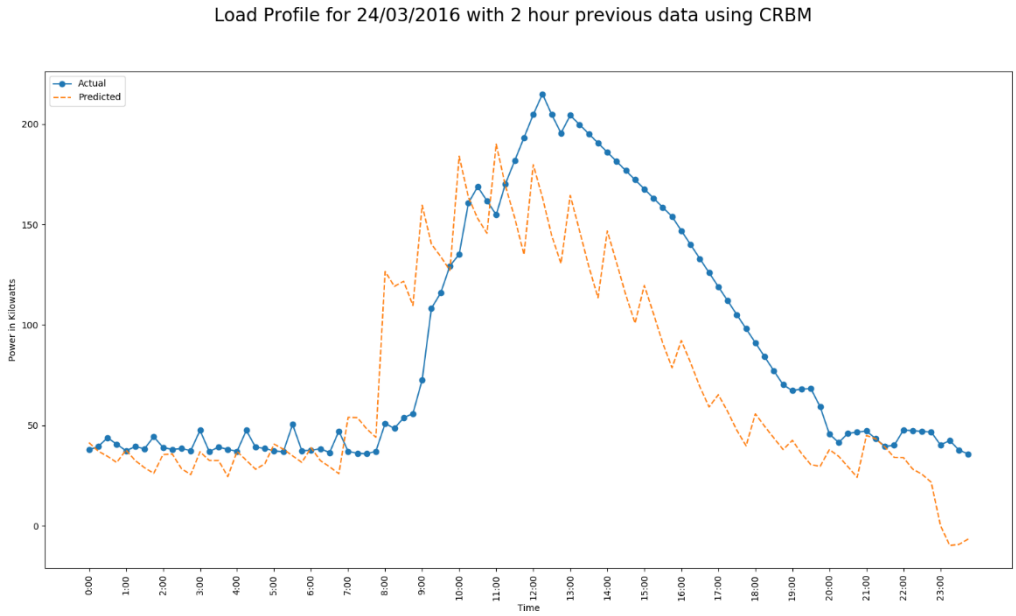

Figure 5.10 Full day prediction with two hours previous data (CRBM)

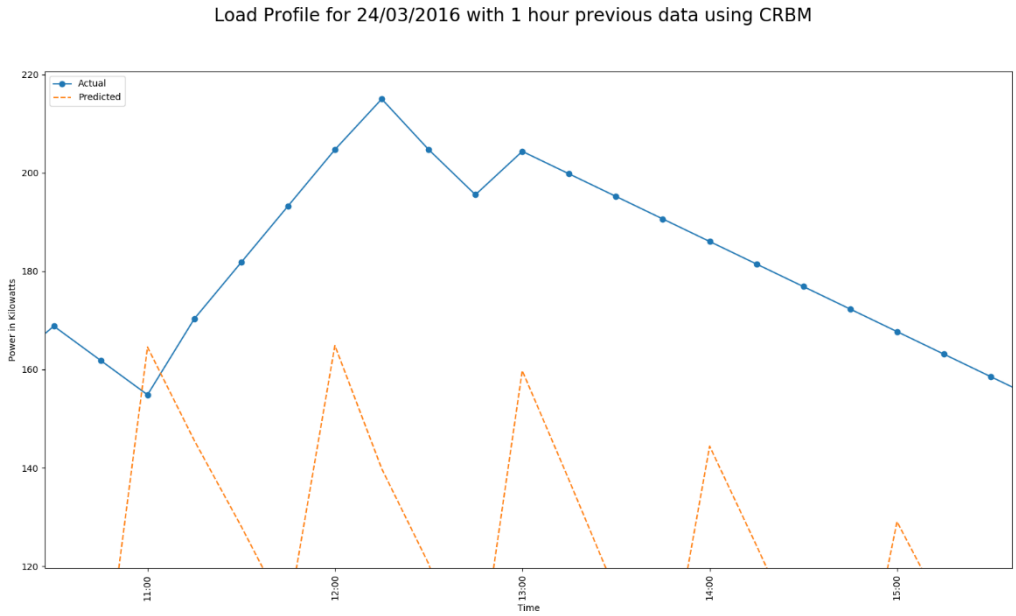

Figure 5.11 One hour prediction (1300 to 1400 hours) with one hour previous data (CRBM)

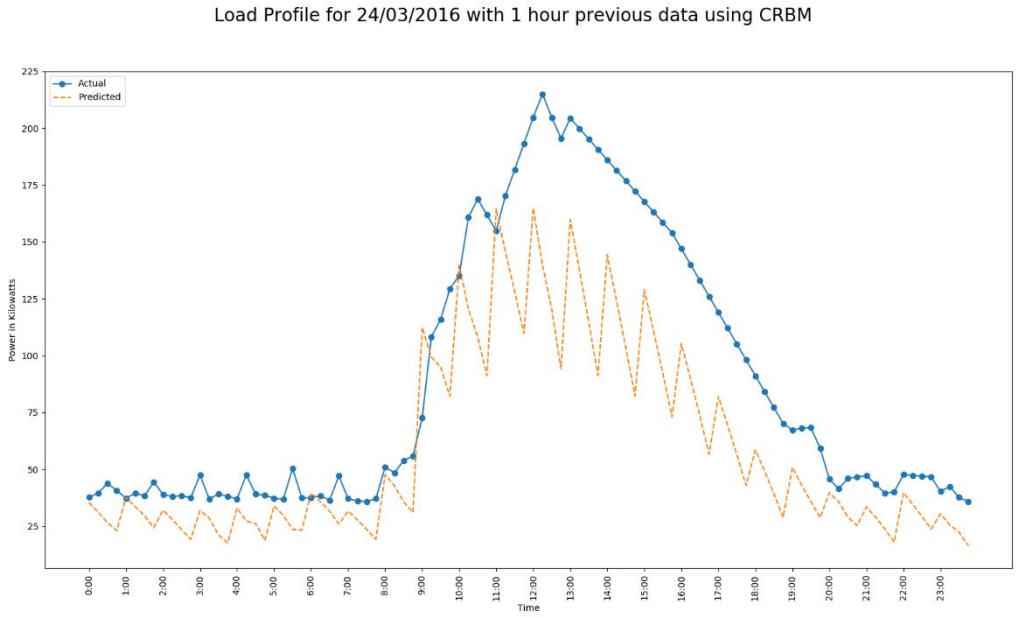

Figure 5.12 Full day prediction with one hour previous data (CRBM)

Figure 5.11 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 1 hour data. Figure 5.4 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 1 hour data input.

5.1.3 LSTM

The input data for LSTM model included load data as well as weather data. This model gave us the least MAPE value proving it to be the most accurate in all three predictions. For a three-hour previous based prediction, 12 data points were given as input. The LSTM layer included 200 nodes followed by a hidden dense layer with 100 nodes. ReLU activation function was used for the complexity as the output of the neurons. The model was compiled using Adam optimiser to obtain 3-4% MAPE.

Results

Figure 5.13 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 3 hours data. Figure 5.14 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 3 hours data input.

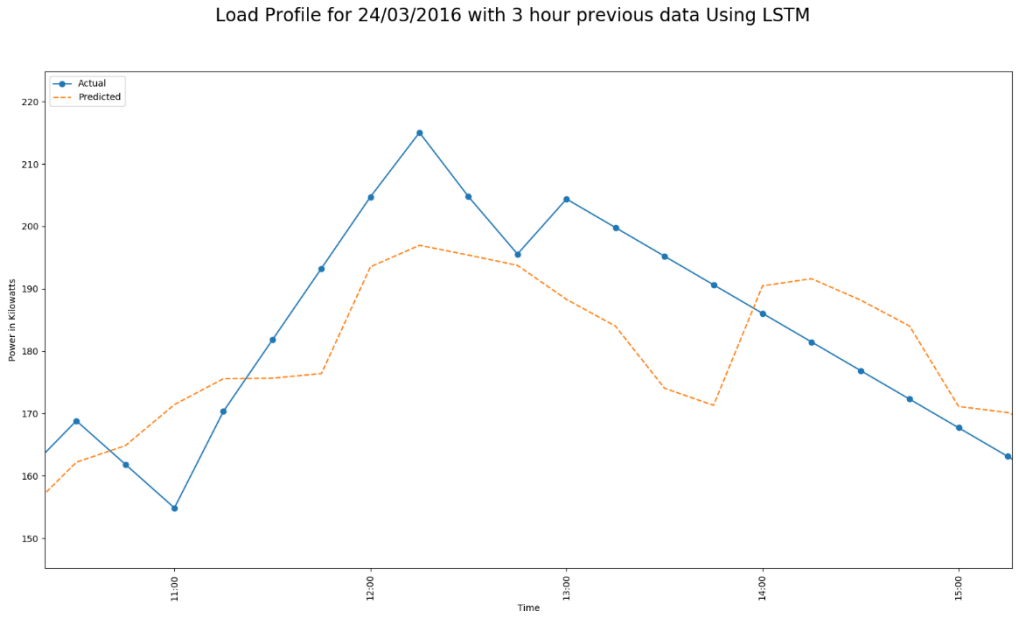

Figure 5.13 One hour prediction (1300 to 1400 hours) with three hours previous data (LSTM)

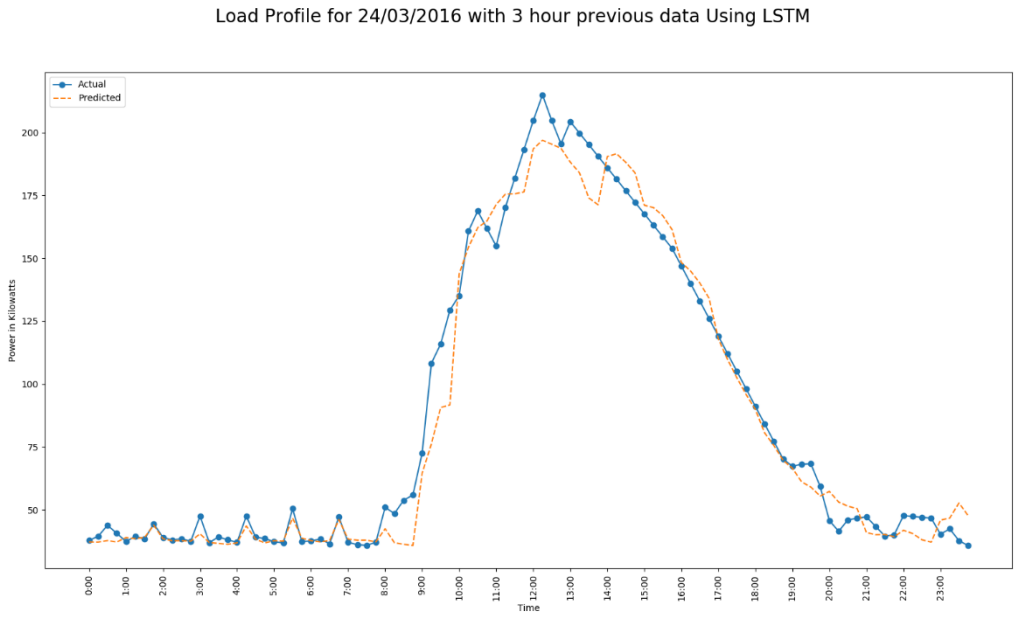

Figure 5.14 Full day prediction with three hours previous data (LSTM)

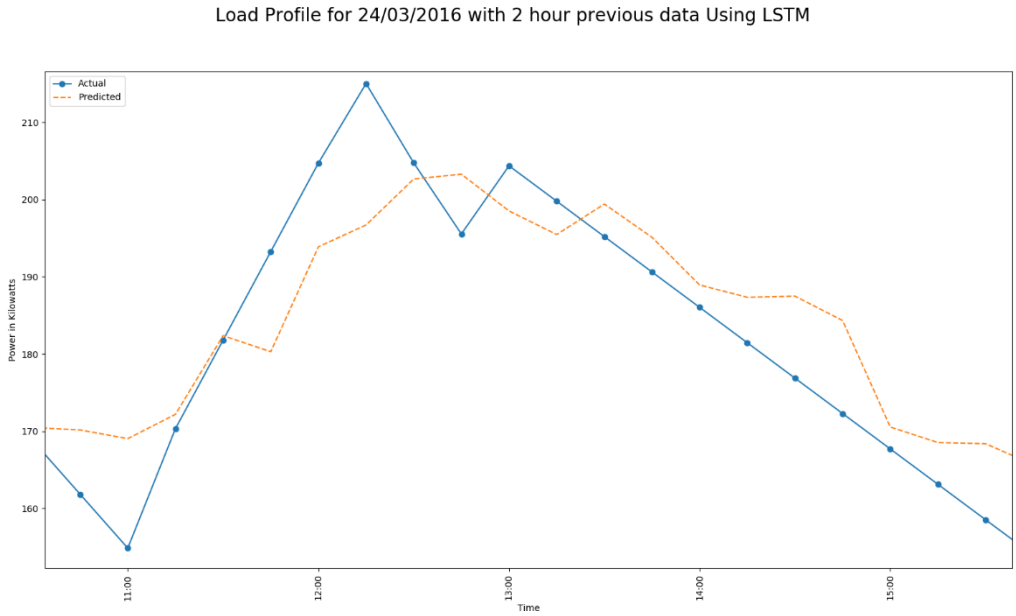

Figure 5.15 One hour prediction (1300 to 1400 hours) with two hours previous data (LSTM)

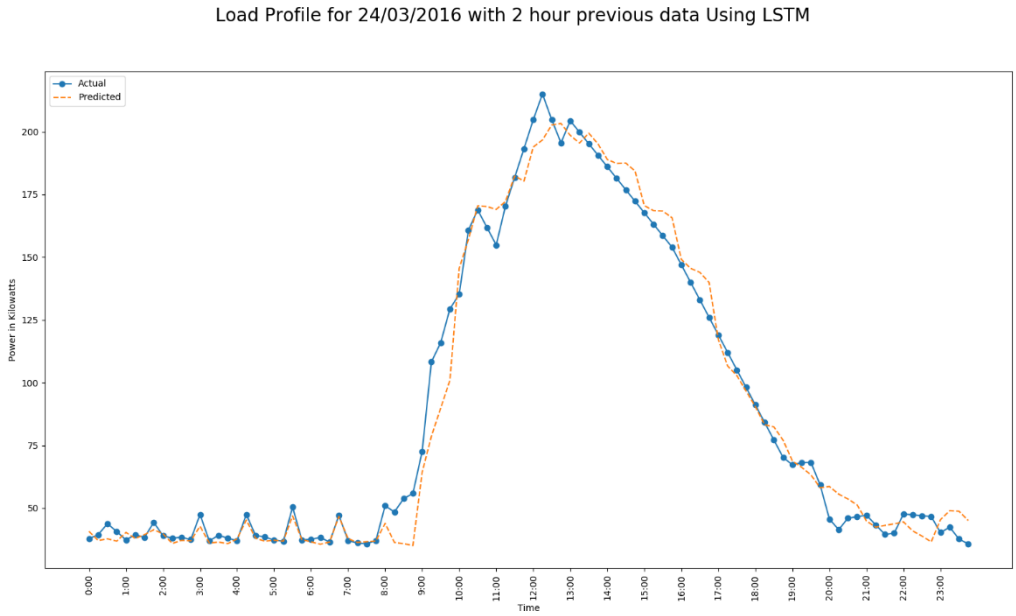

Figure 5.16 Full day prediction with two hours previous data (LSTM)

Figure 5.15 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 2 hours data. Figure 5.16 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 2 hours data input.

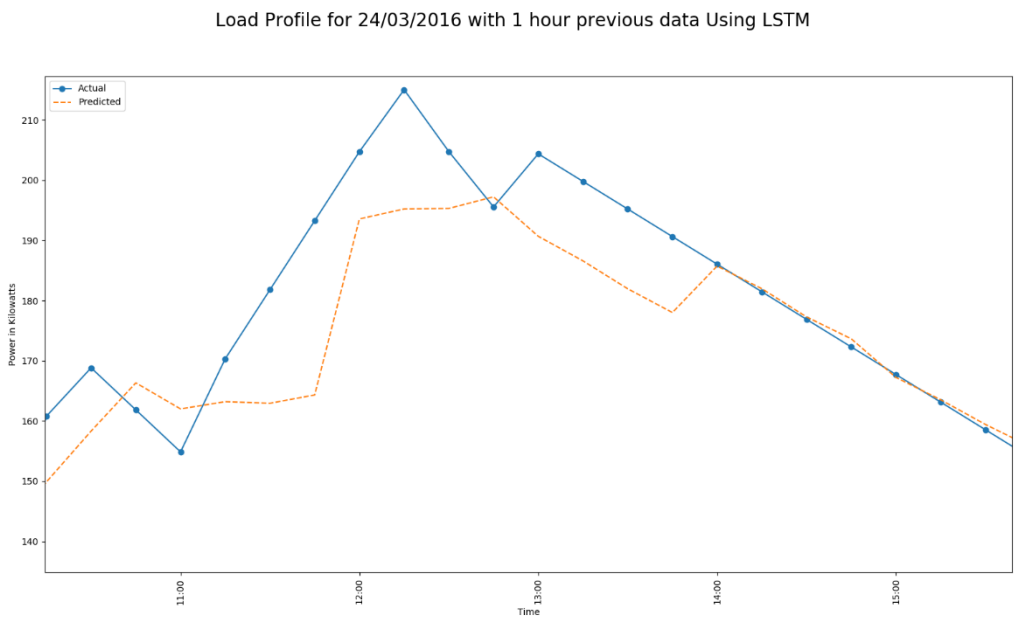

Figure 5.17 One hour prediction (1300 to 1400 hours) with one hour previous data (LSTM)

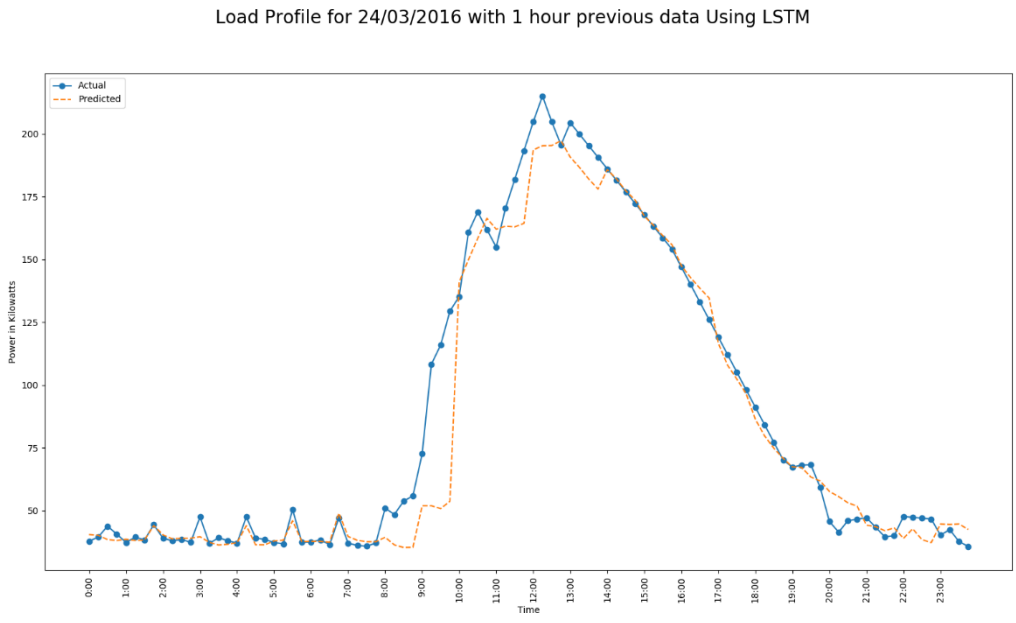

Figure 5.18 Full day prediction with one hour previous data (LSTM)

Figure 5.17 shows the graph obtained for 1 hour (1300 to 1400 hours) prediction using previous 1 hour data. Figure 5.18 shows the graph obtained for predictions from 0000 to 2359 hours on 24/03/2016 with previous 1 hour data input.

5.2 Comparative Studies

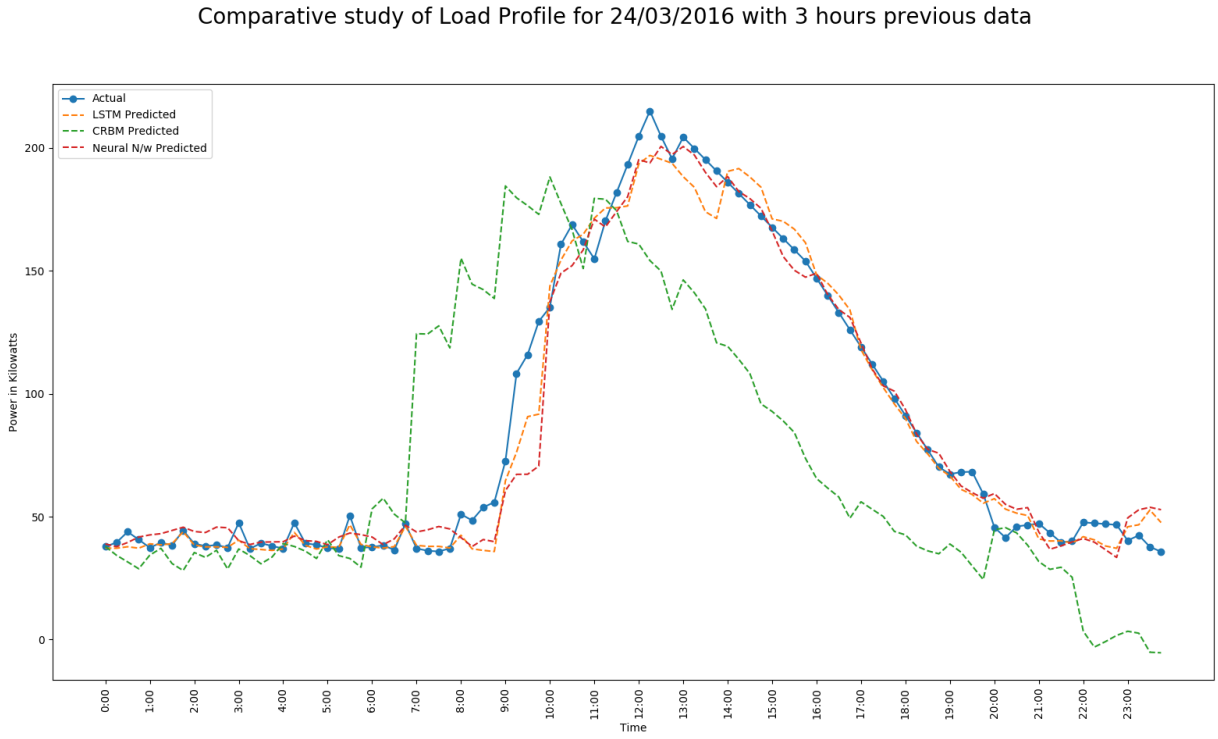

Figure 5.19 Comparative study of the models

Time (Hours)	Actual Power (kW)	Predicted Power (kW)		
		NN	CRBM	LSTM
01:00	37.36	42.78	34.59	39.02
05:00	37.21	38.71	40.56	37.87
09:00	73.07	59.88	184.53	64.59
13:00	204.45	200.72	146.60	188.59
17:00	119.66	120.90	56.29	118.42
21:00	46.71	53.56	38.39	50.53

Table 5.1 Predicted Power with 3 hours previous data

Time (Hours)	Percentage Error		
	NN	CRBM	LSTM
01:00	-14.50749465	7.414346895	-4.443254818
05:00	-4.031174415	-9.002956195	-1.773716743
09:00	18.0511838	-152.5386616	11.60530998
13:00	1.824406945	28.29542675	7.757397897
17:00	-1.03626943	52.95838208	1.03626943
21:00	-14.66495397	17.81203168	-8.178120317

Table 5.2 Percentage Error with 3 hours previous data

24/03/16	NN		CRBM		LSTM	
	MAPE (%)	MAE (kW)	MAPE (%)	MAE (kW)	MAPE (%)	MAE (kW)
	5.5454	11.049	499.5	3686.06	5.5454	5.1948

Table 5.3 Errors on 24/03/2016

Tables 5.1, 5.2 and 5.3 provide the comparison between the predicted powers and the errors obtained with different models. Figure 5.19 shows the predictions provided by all the models. It can be found that LSTM provides the best results for this dataset. The accuracy in predictions using LSTM proves to be the highest among the tested models. It can also be seen in Table 5.3 that the percentage error obtained for CRBM predictions is very high. Hence, CRBM is not recommended for such datasets.

It can be seen from the comparative studies that LSTM is successful in learning the seasonality of the data. It successfully retains the necessary data required to take into account the seasonality while avoiding extreme variations that might shift the model drastically. Hence, LSTM proves to be the best model for this dataset, which is a seasonal time-series data.

PART II

GENERATION SIDE ANALYTICS

CHAPTER 6
DATA ANALYSIS AND MATHEMATICAL MODEL

6.1 Data Analysis

We have obtained two set of time-series data, one from NASA and the other from SolarAnywhere.

NASA

This data set has 32,000 data points with features such as humidity, temperature, wind speed and solar irradiation. Solar irradiation is collected at intervals of 5 minutes for a duration of 1 year.

SolarAnywhere

This data set has 8,000 data points with the feature of Global Horizontal Irradiance (GHI). Solar irradiation is collected at intervals of 1 hour for a duration of 1 year.

These time-series data need to be normalized before passing to some models such as neural networks.

The graphs for daily, monthly and yearly irradiation and the correlation between radiation versus temperature and radiation versus windspeed are obtained using visualization tools.

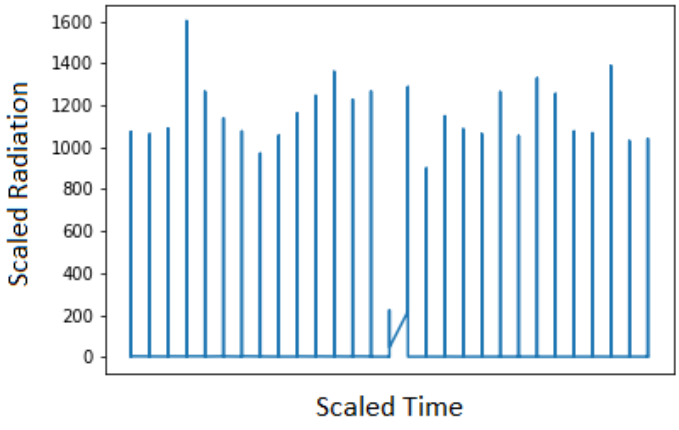

Figure 6.1 Graphical Representation of Radiation in One Month

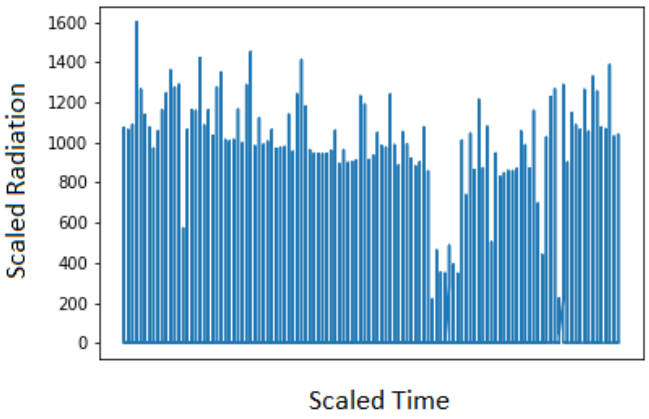

Figure 6.2 Graphical Representation of Radiation in One Year

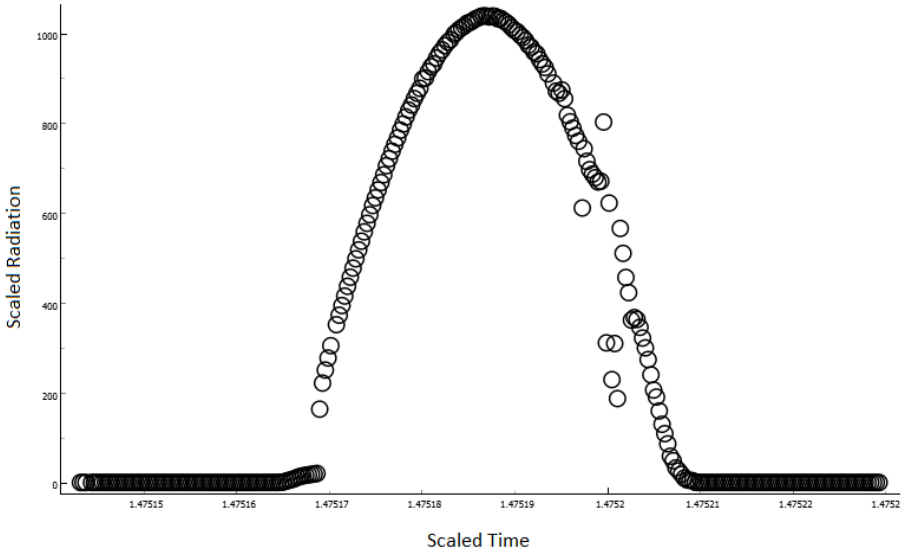

Figure 6.3 Graphical Representation of Radiation in One Day

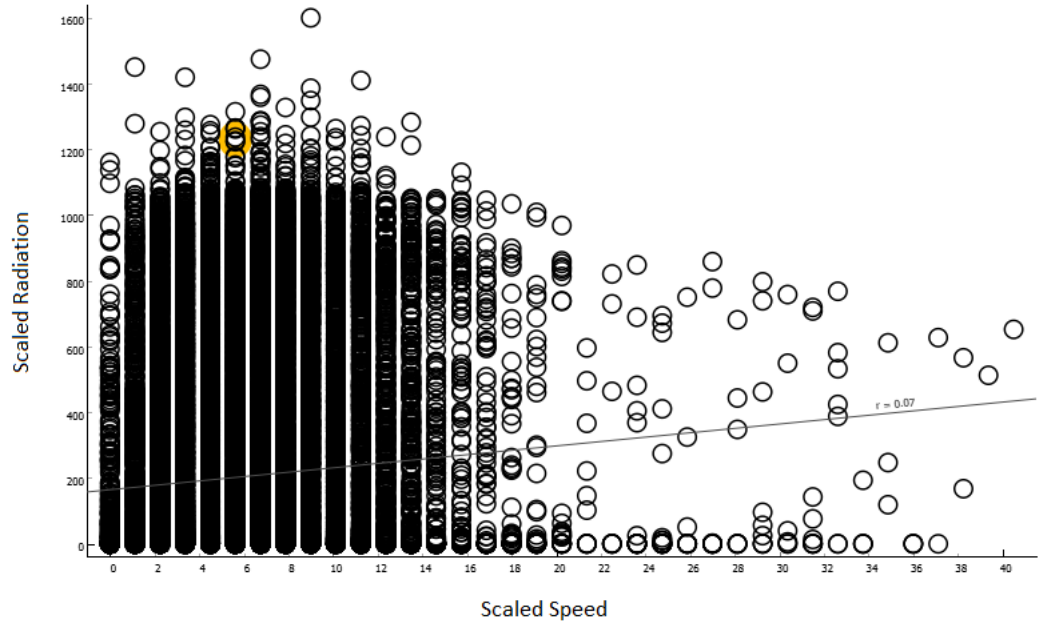

Figure 6.4 Plot of Radiation vs Speed

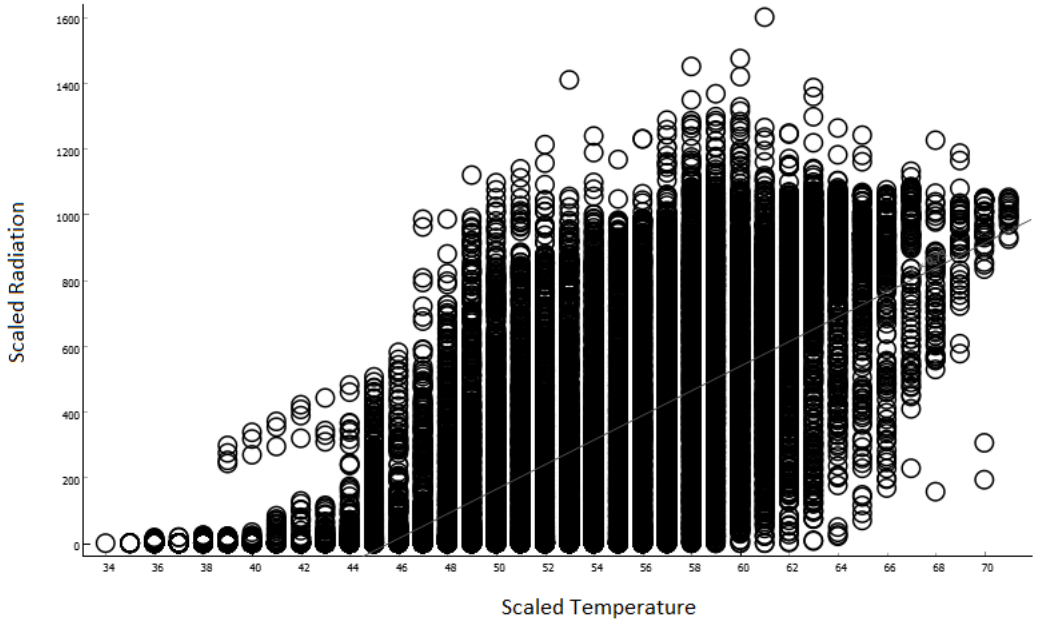

Figure 6.5 Plot of Radiation vs Temperature

In Figure 6.4, it is observed that when the windspeed increases initially, the irradiance increases. But after a certain windspeed, the irradiance starts decreasing because of obvious drop in temperature.

In Figure 6.5, it is observed that with the increase in temperature, irradiance increases. But at very high temperature, the performance of the solar panel deteriorates due to overheating, hence not fully utilising the available irradiance.

6.2 Mathematical Model

The main objective is to identify the most suitable model with the least value of error. Hence, the objective function for model evaluation is Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). The formulas for the same are as follows.

$$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left|\frac{e_t}{y_t}\right|$$

where

- $e_t = y_t - y^{p}_{t}$
- n represents the number of data points
- y_t is the actual value for data point t
- y^{p}_{t} is the predicted value for data point t

CHAPTER 7
MODELS IMPLEMENTED AND COMPARATIVE STUDIES

7.1 Models Implemented

Based on the literature survey, a few models suitable for time-series data with seasonality have been implemented. Short-term forecasting has been conducted using Logistic Regression, ANN, XGBoost and ARMA. The predictions have been made for 1 hour and 3 hours. Long-term forecasting has been conducted using LSTM. The prediction has been made for 2 days.

7.1.1 Logistic Regression

On implementing this method, a very high mean squared error was obtained, which resulted in poor output. This is because the input data taken was time-series data with seasonal trends and logistic regression fails to capture these trends.

7.1.2 Artificial Neural Network

The data available is not in a suitable format to be input to this model. Initially, data cleansing is performed in order to get data which is suitable to input to this model. This includes removing NaN values and then scaling or normalising the data to a particular range of say 0 to 1. This data is then split into train and test (usually 80% and 20%) before initialising the model.

The model built consists of a 4 layered neural network with 6 input nodes, 20 nodes in hidden layer 1, 10 nodes in hidden layer 2 and 1 output node. Each hidden layer uses ‘sigmoid’ as its activation function. The optimizer used for this model is ‘adam’, also known as adaptive gradient algorithm. The output of each node lies between 0 and 1. The model is trained with train data and target variable being radiation with 100 epochs. The test data is fed to the model where the model adjusts the weights by back propagation, thereby predicting the output. Then, cross validation is performed with the expected and predicted values using mean squared error as loss function.

```
Epoch 97/100
23490/23490 [==============================] - 1s 53us/step -
loss: 39074.1439 - acc: 8.5143e-05 - val_loss: 37375.3285 -
val_acc: 0.0000e+00
Epoch 98/100
23490/23490 [==============================] - 1s 53us/step -
loss: 39075.9999 - acc: 0.0000e+00 - val_loss: 37763.4566 -
val_acc: 0.0000e+00
Epoch 99/100
23490/23490 [==============================] - 1s 53us/step -
loss: 39074.3791 - acc: 0.0000e+00 - val_loss: 36713.7091 -
val_acc: 0.0000e+00
Epoch 100/100
23490/23490 [==============================] - 1s 54us/step -
loss: 39077.3001 - acc: 0.0000e+00 - val_loss: 37720.4731 -
val_acc: 0.0000e+00
```

Figure 7.1 Output of ANN method with Mean Squared Error

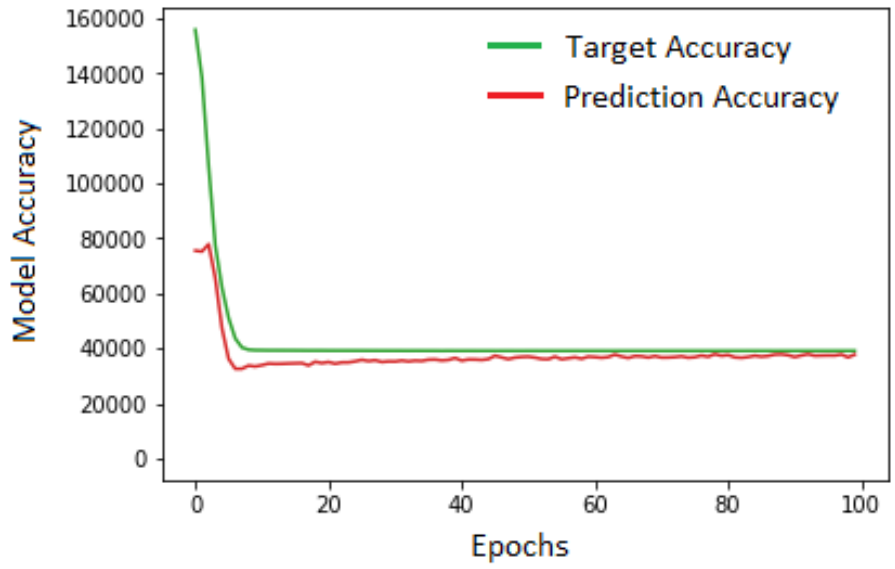

Figure 7.2 Graphical representation of Target Accuracy and Prediction Accuracy

Figure 7.1 shows the errors obtained for ANN. Figure 7.2 shows the graph for target accuracy and prediction accuracy using ANN. For this model, a mean squared error of 37000 was obtained which proved more efficient than logistic regression but not decision tree.

7.1.3 Decision Tree using XGBoost

XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

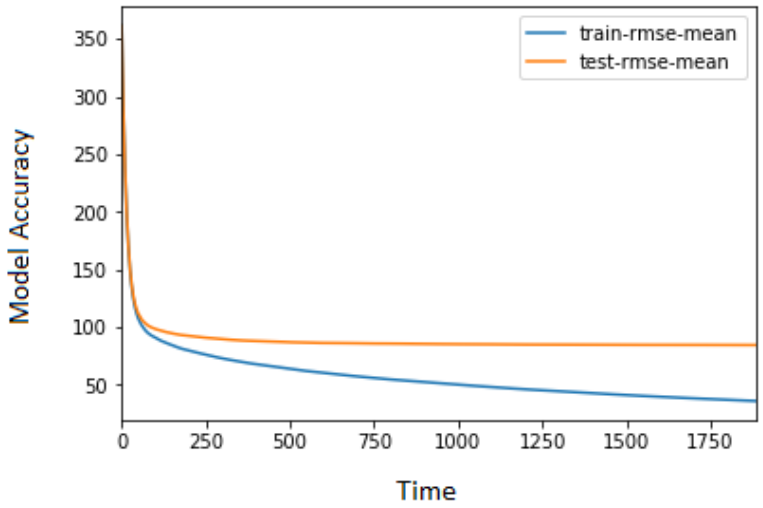

Figure 7.3 Graphical representation of Training Model and Test Model

For the built model, a mean squared error of 7000 was obtained which proved to be more efficient than ANN model. Figure 7.3 shows the RMSE graphs for train model and test model data.

7.1.4 ARMA

Based on Table 3.1, AR and ARMA model were selected for the NASA dataset and for the SolarAnywhere dataset respectively.

Using NASA Data

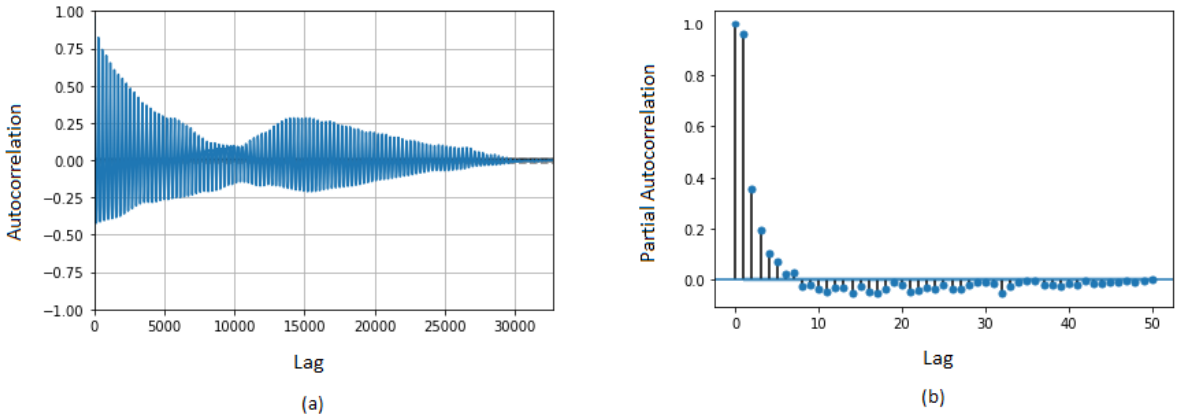

Figure 7.4 ACF and PACF (a) Autocorrelation Function where function decays to zero and (b) Partial Autocorrelation Function where function spikes down to zero

From the autocorrelation graphs shown in Figure 7.4, it is observed that the ACF graph is decaying to zero whereas the PACF graph cuts-off to zero. Hence, AR model is suitable for this data.

Figure 7.5 represents solar radiation data for a one hour ahead prediction. The blue points represent predicted data and it is observed that there is a bias close to 25. By implementing this bias into the code, it is possible to get better accuracy. The best values for parameters p and q were chosen after trying several values and seeing which gave better results.

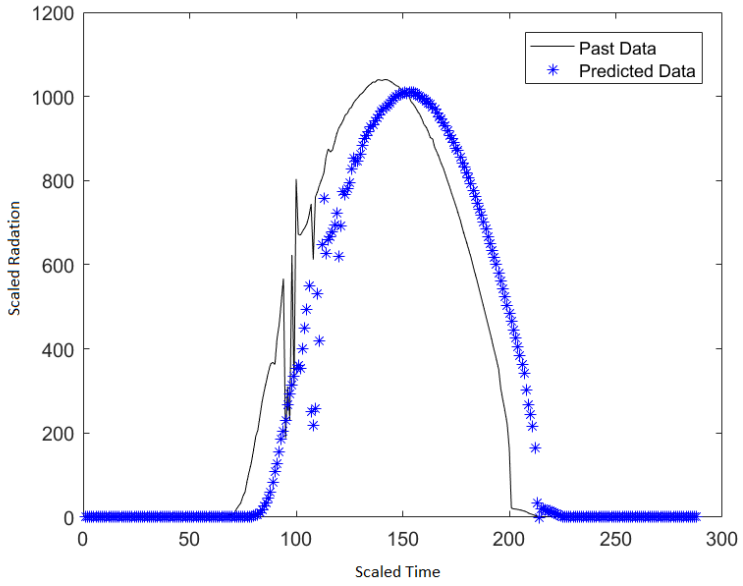

Figure 7.5 Graphical representation of past data and predicted data using AR model (one hour ahead prediction)

Using SolarAnywhere Data

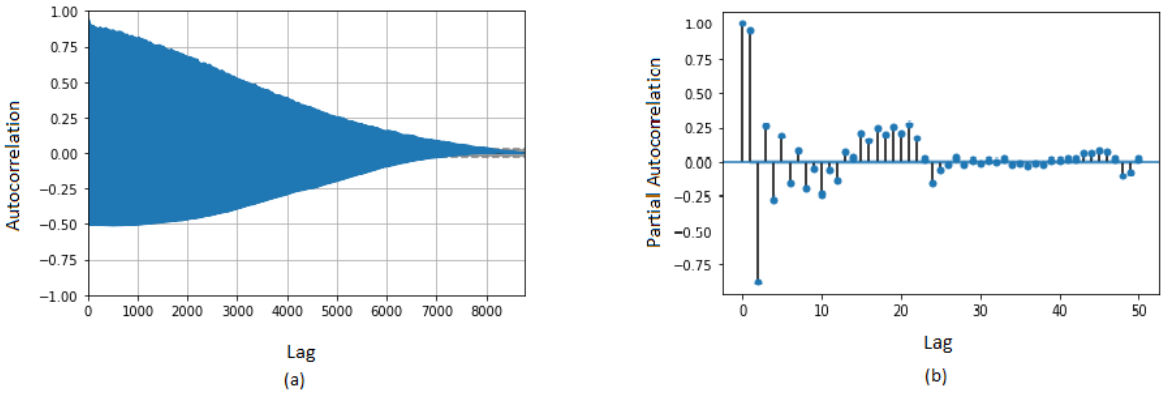

Figure 7.6 ACF and PACF (a) Autocorrelation Function where function decays to zero and (b) PACF where function decays to zero

From Figure 7.6, it is observed that both ACF graph and PACF graph decay to zero. Hence, ARMA model is suitable for this data.

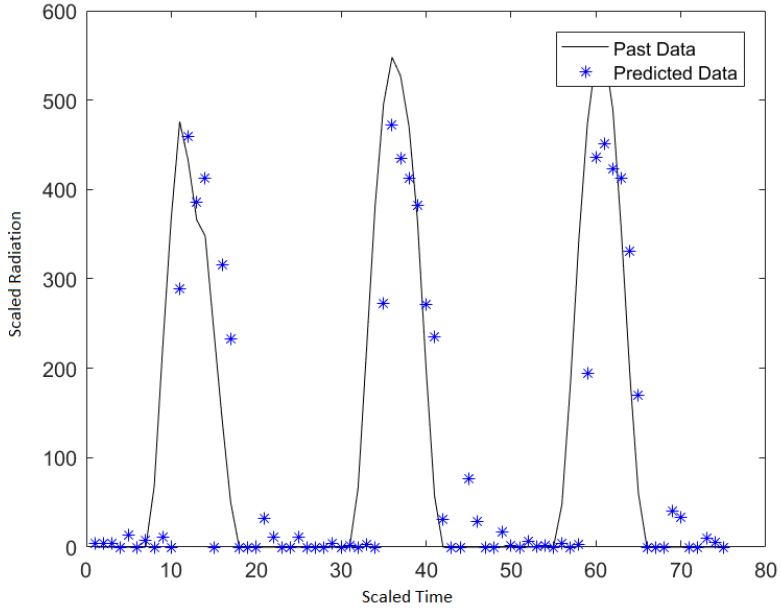

Figure 7.7 Graphical representation of past three days data with a prediction interval of three hours

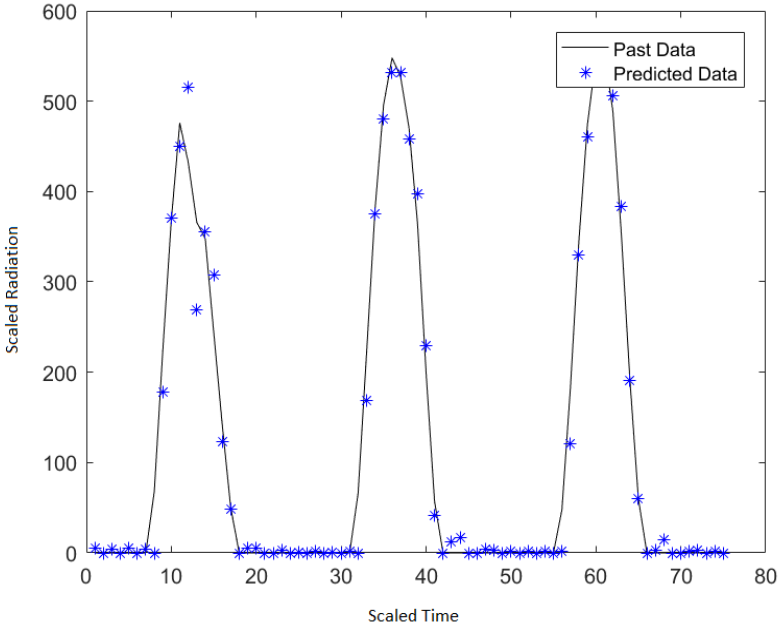

Figure 7.8 Graphical representation of past three days data with a prediction interval of one hour

Figure 7.7 represents solar radiation data for a three hour ahead prediction using past three days data. Figure 7.8 represents data for a one hour ahead prediction using past 3 days data. The best values for parameters p and q were chosen after trying several values and seeing which gave better results.

7.1.5 LSTM

This designed model for generation side prediction consists of 200 LSTM layers and 100 hidden dense layers. It takes 12 inputs and produces 12 outputs. The results obtained were for 2 days.

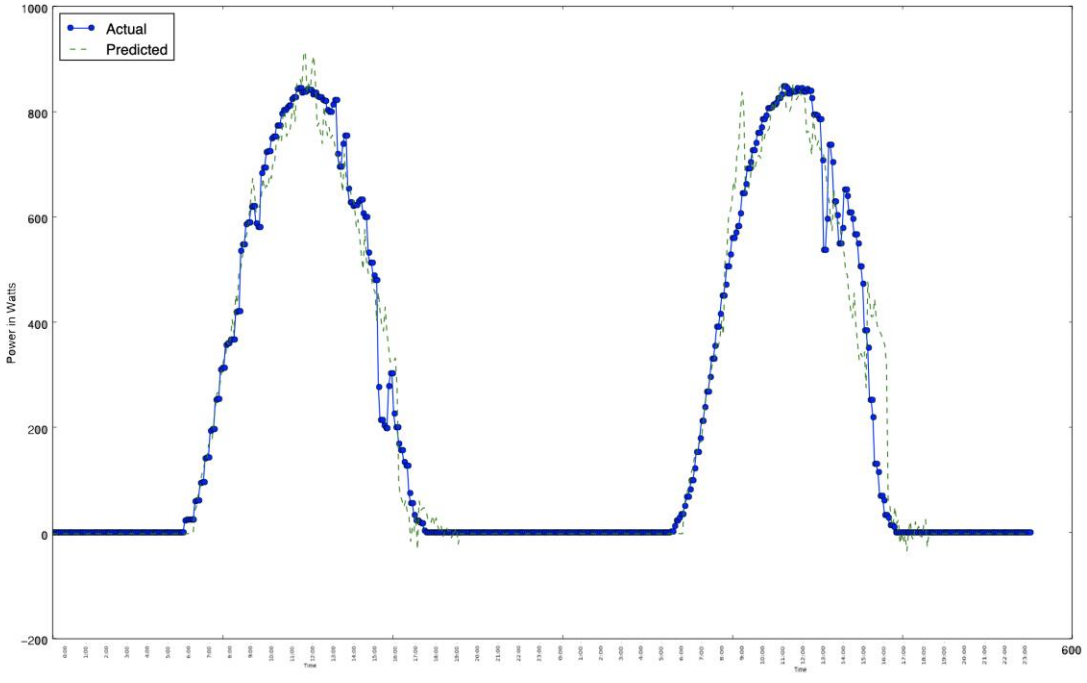

Figure 7.9 Graphical representation of prediction for 2 days with LSTM

Figure 7.9 shows the graph obtained for 2 days prediction using LSTM. The accuracy obtained is satisfactory.

7.2 Comparative Studies

The first model implemented was Logistic Regression. However, a high mean square error was obtained and it was concluded that this method was not suitable for data with seasonal trends. The Decision Tree method using XGBoost proved to be a more efficient method with the least mean squared error.

For short term forecasting, ARMA model was implemented. Based on the autocorrelation and partial autocorrelation functions, results were executed in either AR or ARMA model and by selecting the appropriate parameters required, better results were achieved. Since this was short term forecast, the graph for a one hour ahead prediction using past three days data was obtained.

From comparative studies, ARMA proved to be the best models for generation side forecasting. The errors obtained were the least for this model.

For long-term forecasting, LSTM was implemented. The results obtained were satisfactory. Hence, it can be deduced that LSTMs are suitable for long-term forecasting of seasonal time-series data.

CHAPTER 8
WEB APPLICATION AND VISUALISATION

From the industry, valuable insights about their requirements with regards to utilization of this technology were received. Naturally, it became important to develop a graphical user interface (GUI) that enables users to analyse and deploy models for gaining insights on the data.

Our web service includes mainly three parts viz. frontend, backend and database. The developed backend is based on a Python-based server service known as Flask that enables deployment of RESTful APIs where the users can integrate their system with ours. The APIs are as follows.

- Deploy Model
- Predict Load
- Predict Generation

Frontend GUI is built on React.js which is a JavaScript based framework for frontend development. Fetch calls have been used in order to fetch data from the backend and display it on graphs. For development of graphs, Chart.js has been used which is, again, a JavaScript based framework integrated with React.

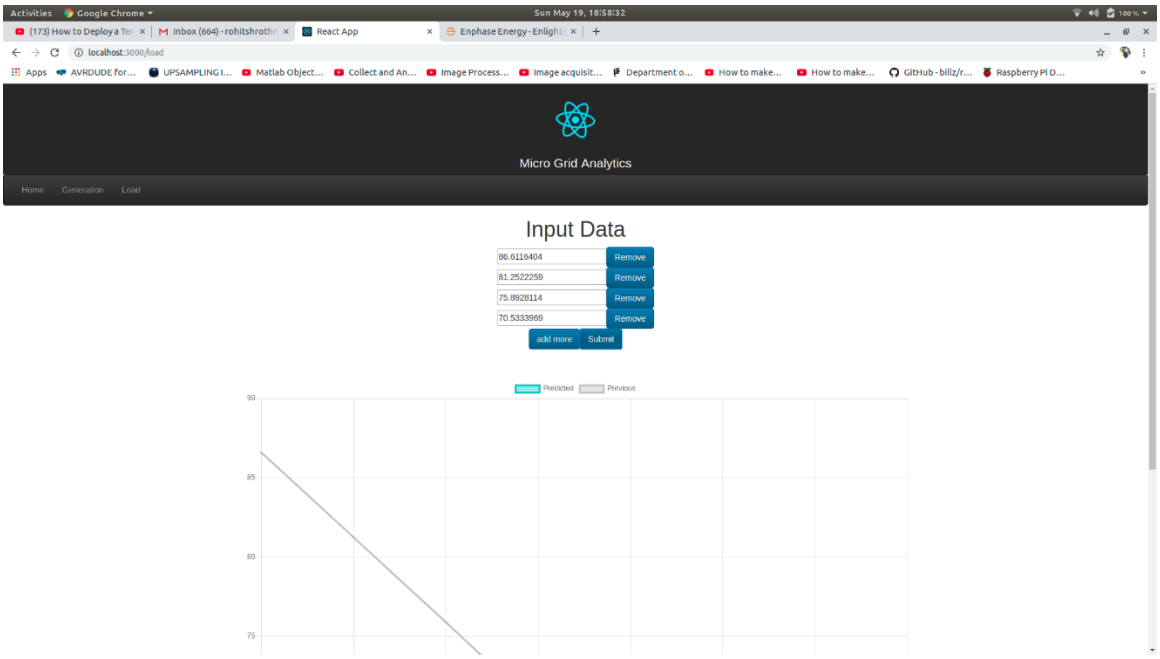

Figure 8.1 Load Side on Web App

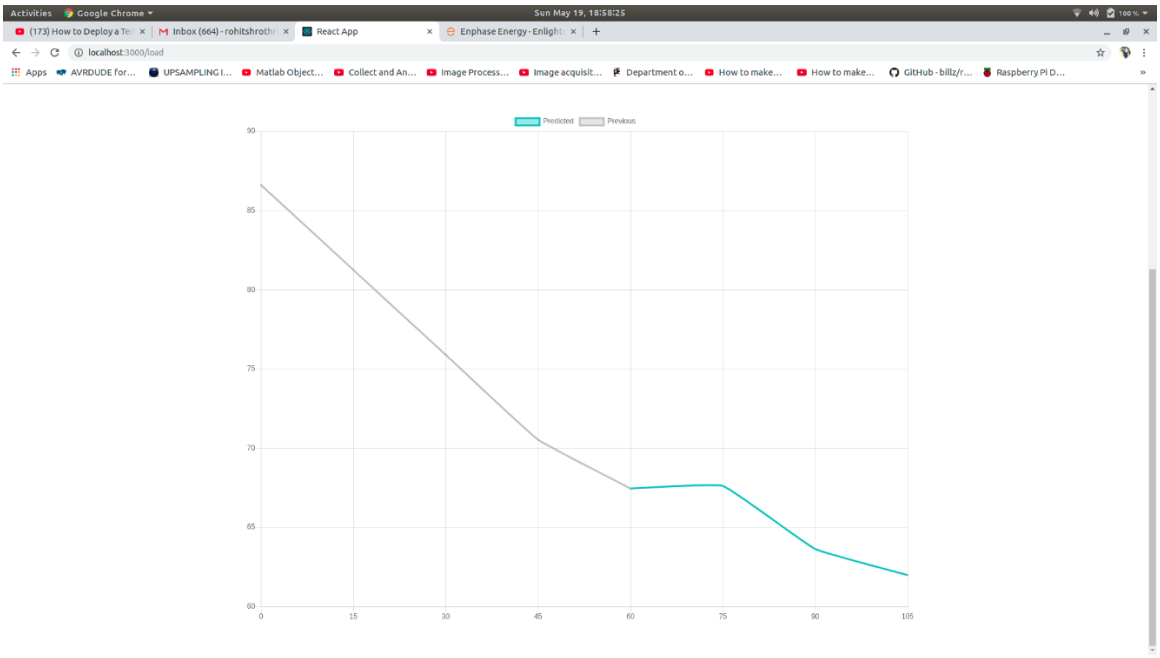

Figure 8.2 Graphical Representation of Load Side Prediction on Web App

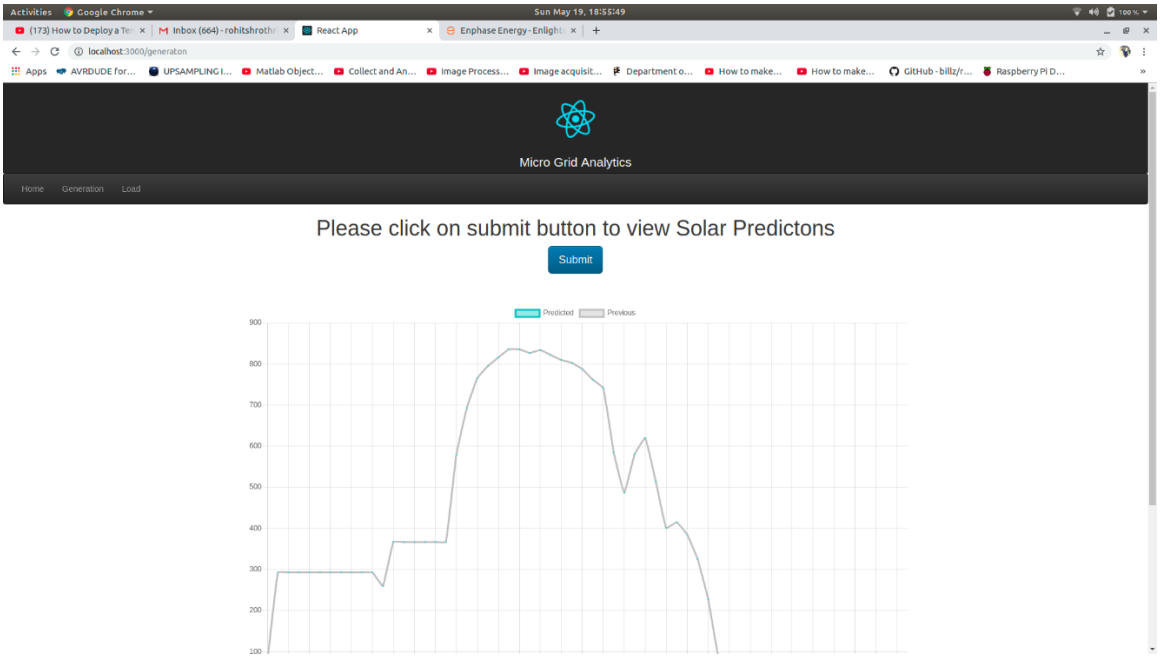

Figure 8.3 Generation Side on Web App

Python threads have been used to call API from the Enphase system setup in the laboratory environment at 15-minutes intervals and save the data into a MongoDB database.

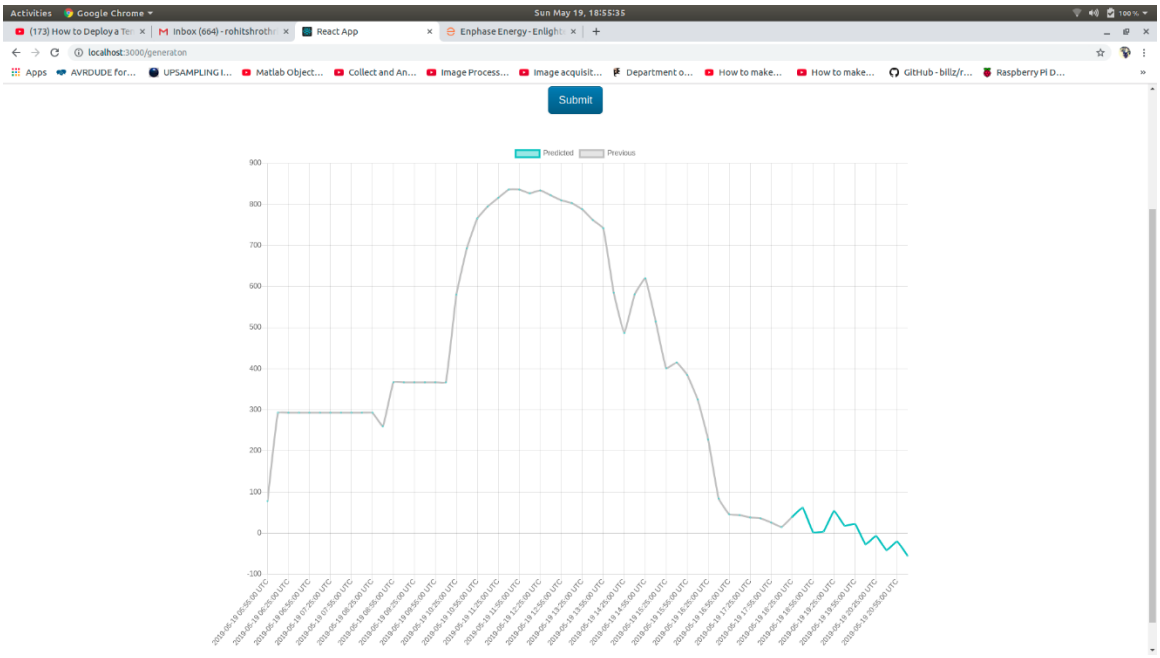

Figure 8.4 Graphical Representation of Generation Side Prediction on Web App

For easy deployment of the developed system, all the web services have been developed on docker containers with environment setup. These containers can be easily deployed on Amazon web services or any other cloud-based service with ease. This entire application shall be available as an Open Source Project on github.com/randr97.

This application can be used by various service-based company that provide renewable technology. It would enable customers to get more insights about their systems and also predict the cost and return on investment of their system. They can also make an informed decision between selecting conventional source and non-conventional source of energy. Overall, it is a profit to both customers and service providers, thereby increasing the use of renewable sources of energy.

CHAPTER 9
CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

Load Side

CRBM model exhibited the highest MAPE value with very poor accuracy in predictions. This model was not suitable for the available dataset.

Neural Networks proved to be an efficient model for some part of the predictions. However, considering over a large period of time, LSTMs obtained the least MAPE value giving most accurate results throughout.

Generation Side

Logistic regression model obtained a high mean square error and it was concluded that this method was not suitable for data with seasonal trends.

Compared to Neural Networks and Decision Tree using XGBoost, ARMA model had the least error values and was most suitable for the short-term forecasting executed in this project.

Based on the autocorrelation and partial autocorrelation functions, results were executed in either AR or ARMA model and by selecting the appropriate parameters required, better results were achieved. Since this was short term forecast, the graph for a one hour ahead prediction using past three days data was obtained.

9.2 Future Scope

The project has been designed to predict the power generated by solar plant. This can be extended to wind energy and tidal energy as well. This will make the microgrid more robust and versatile.

Website, at its current stage of development, is serving pre-deployed models and the user can test these models only for a specific data. This can be further extended by providing flexibility to the users to deploy their own models. The models that are deployed are served using flask backend but this could lead to server overloading and crashing when a plethora of users hit the server. In

order to make up for this issue, these models can be served using Google's production level API known as TensorFlow serving.

Cache Clearing

The database that is being used is MongoDB which is a NoSQL database. It becomes important to optimize the database when more features are added with multiple schemas. In order to do so, when users log out or when the choose to exit the platform, the cache that is stored should be cleared as it improves the query performance. It is very important to delete unwanted data from the database to make sure that the server is able to cater to multi-client requests without crashing.

Authorisation

The data in the website needs to be made safe. This makes it necessary to employ methods of authentication and authorisation. Basic Authentication or Open Authorisation (OAuth) can be used for authorisation. The advantage of Open Authorisation over Basic Operation is that it is more secure. Credentials can be hacked in case of basic authorisation. This is not the case with OAuth. OAuth helps in creating a secure passage to access the REST API. OAuth 2.0 is the most suitable form of authentication for this purpose.

Data Security

The data present is the system is very crucial and is highly subjected to cyber-attacks. It becomes very important to safeguard this data by using a highly secure authentication technique such as SSO. This data has to be stored in encrypted form which can be decrypted only when the user is authorised. The system is highly prone to DOS (Denial of Service) attacks which would disable the entire system from parsing any client request. It is very important to safeguard the servers from such attacks which can be made possible by rejecting multiple requests from a single system and also detecting VPNs that are used to deploy such attacks.

REFERENCES

[1] Warrior, Karun P., M. Shrenik, and Nimish Soni. "Short-term electrical load forecasting using predictive machine learning models." *2016 IEEE Annual India Conference (INDICON)*. IEEE, 2016.

[2] Muhammad Usman Fahad and Naeem Arbab, “Factor Affecting Short Term Load Forecasting,” Journal of Clean Energy Technologies, vol. 2, No. 4, Oct. 2014.

[3] Dongxiao Niu, Yongli Wang and Desheng Dash Wu, “Power load forecasting using support vector machine and ant colony optimization,” Expert Systems with Applications 37, pp. 2531–2539, 2010.

[4] Y. Chen et al., “Short-Term Load Forecasting: Similar Day-Based Wavelet Neural Networks,” in IEEE Transactions on Power Systems, vol. 25, no. 1, pp. 322-330, Feb. 2010.

[5] Elena Mocanu et al., “Deep learning for estimating building energy consumption,” Sustainable Energy, Grids and Networks, vol. 6, Pages 91–99, June 2016

[6] Bakker, Bram. "Reinforcement learning with long short-term memory." *Advances in neural information processing systems*. 2002.

[7] L.-J. Lin and T. Mitchell. Reinforcement learning with hidden states. In Proc. of the 2nd Int. Conf. on Simulation of Adaptive Behavior. MIT Press, 1993.

[8] Hiyama, Takashi, and Ken Kitabayashi. "Neural network based estimation of maximum power generation from PV module using environmental information." IEEE Transactions on Energy Conversion 12.3 (1997): 241-247

[9] Torres, Jose Luis, et al. "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)." Solar Energy 79.1 (2005): 65-77

[10] Chen, Changsong, et al. "Smart energy management system for optimal microgrid economic operation." *IET renewable power generation* 5.3 (2011): 258-267.

[11] Foley, Aoife M., et al. "Current methods and advances in forecasting of wind power

generation." Renewable Energy 37.1 (2012): 1-8

[12] Hernández, Luis, et al. "A study of the relationship between weather variables and electric power demand inside a smart grid/smart world framework." Sensors 12.9 (2012): 11571-11591

[13] Amjady, Nima, Farshid Keynia, and Hamidreza Zareipour. "Short-term load forecast of microgrids by a new bilevel prediction strategy." *IEEE Transactions on smart grid* 1.3 (2010): 286-294.

[14] Gers, Felix A., Douglas Eck, and Jürgen Schmidhuber. "Applying LSTM to time series predictable through time-window approaches." *Neural Nets WIRN Vietri-01*. Springer, London, 2002. 193-200.