1-1-2019

Supervised Machine Learning Techniques for Short-Term Load Forecasting

Harish Amarasundar
University of Denver

Supervised Machine Learning Techniques for Short-Term Load Forecasting

Abstract

Electric Load Forecasting is essential for the utility companies for energy management based on the demand. Machine Learning Algorithms has been in the forefront for prediction algorithms. This Thesis is mainly aimed to provide utility companies with a better insight about the wide range of Techniques available to forecast the load demands based on different scenarios. Supervised Machine Learning Algorithms were used to come up with the best possible solution for Short-Term Electric Load forecasting. The input Data set has the hourly load values, Weather data set and other details of a Day. The models were evaluated using MAPE and R2 as the scoring criterion. Support Vector Machines yield the best possible results with the lowest MAPE of 1.46 %, a R2 score of 92 %. Recurrent Neural Networks univariate model serves its purpose as the go to model when it comes to Time-Series Predictions with a MAPE of 2.44 %. The observations from these Machine learning models gives the conclusion that the models depend on the actual Data set availability and the application and scenario in play

Document Type

Thesis

Degree Name

M.S.

Department

Electrical Engineering

First Advisor

Mohammad A. Matin, Ph.D.

Second Advisor

George Edwards, Ph.D.

Third Advisor

Mohammed Mahoor, Ph.D.

Keywords

Electric energy, Load forecasting, Machine learning, Neural networks, Time series predictions, Utility companies

Subject Categories

Electrical and Computer Engineering | Engineering

Supervised Machine Learning Techniques for Short-Term Load Forecasting

---

A Thesis

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Harish Amarasundar

August 2019

Advisor: Dr. Mohammad A. Matin

Author: Harish Amarasundar
Title: Supervised Machine Learning Techniques for Short-Term Load Forecasting
Advisor: Dr. Mohammad A. Matin
Degree Date: August 2019

Abstract

Electric Load Forecasting is essential for the utility companies for energy management based on the demand. Machine Learning Algorithms has been in the forefront for prediction algorithms. This Thesis is mainly aimed to provide utility companies with a better insight about the wide range of Techniques available to forecast the load demands based on different scenarios. Supervised Machine Learning Algorithms were used to come up with the best possible solution for Short-Term Electric Load forecasting. The input Data set has the hourly load values, Weather data set and other details of a Day. The models were evaluated using MAPE and R2 as the scoring criterion. Support Vector Machines yield the best possible results with the lowest MAPE of 1.46 %, a R2 score of 92 %. Recurrent Neural Networks univariate model serves its purpose as the go to model when it comes to Time-Series Predictions with a MAPE of 2.44 %. The observations from these Machine learning models gives the conclusion that the models depend on the actual Data set availability and the application and scenario in play.

Acknowledgements

This Thesis is submitted to the Daniel Felix Ritchie School of Engineering Computer Science at University of Denver as a fulfillment of the requirements for the Master of Science degree in Electrical Engineering. The research has been conducted at the Department of Electrical and Computer Engineering, which is part of the Daniel Felix Ritchie School of Engineering Computer Science at University of Denver.

Firstly I would like to thank my advisor Dr. Mohammad A. Matin, who supports my Master's program as my advisor. I would like to express my gratitude to Dr. George Edwards, who is always giving me valuable advice on research. Finally, I would like to thank my other committee members: Dr. Mohammed Mahoor, Dr. Yun-bo Yi for your attendance, support and questions. I gained valuable knowledge, ideas and inspiration from all those great people.

Finally, I owe many thanks to my parents, friends and family for all their love, understanding and supports in my pursuits, for periods where I have been apart away from them. Thank you!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electric Energy

Electrical Energy is a form of energy that results from the flow of Electric Charge. Electrical Energy is not a material resource that can be stored and reused later, it has to be generated and transferred based on the Demand on an instantaneous basis. Electrical Energy is used all over the world on a regular basis that help power the devices that run on Electricity. Electricity to this day is the most important invention because it serves as the baseline for all inventions to come. It is safe to say that Electrical Energy has become an important resource in this modern generation. Electrical Energy is generated from Electric Power Generation stations from resources like Natural Gas, Solar, Coal, Fossil Fuels, Nuclear, Hydroelectric, Wind turbines, Geothermal, Biomass and other sources as shown in Fig 1.1 [1]. Electrical Energy is distributed based on a statistic or a demand from the consumers.

That is where Load forecasting comes into play which gives the utility companies with meaningful information. The Utility Companies make use of these Data and prediction algorithms which provides them a better sense of the Load Demand for future consumption. Predicted Load Demand allows the Utility Companies to efficiently allocate resources and meet the supply the Demand of the consumers.

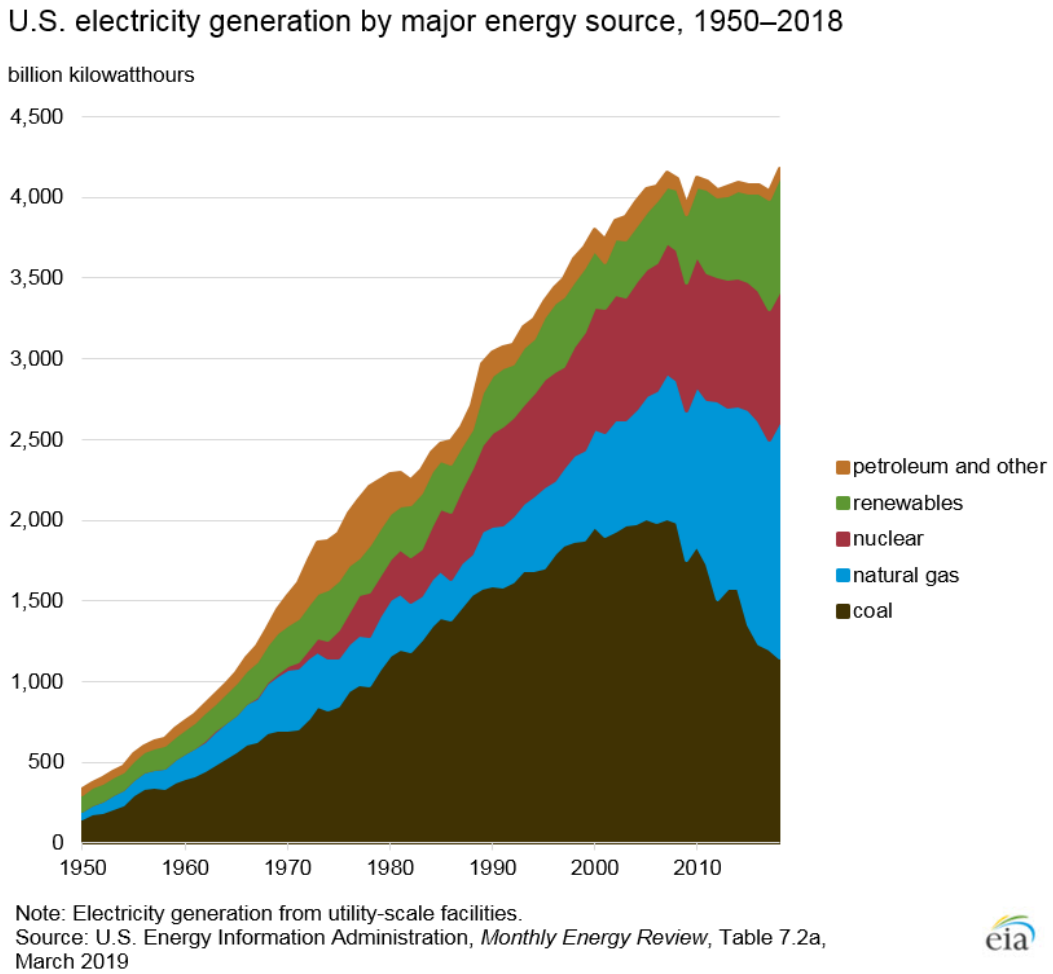

Figure 1.1. Sources of Electrical Energy - USA [1]

Load Forecasting

The Term Load Forecasting can be defined as the way or the methods by which Utility/Energy Companies predict the Power/Energy required by the consumers both residential and commercial and supply the required Load Demand's for Short-term, Mid-Term and on a Long-Term Basis.

Need for Load Forecasting

Global electricity demand is projected to increase by 85% in 2040 as living standards rise, economies expand and the need for electrification of society continues. Electricity demand forecasting plays an important role in load allocation and planning for future generation facilities and transmission augmentation. Load demand in a given season is subject to a range of uncertainties, including underlying population growth, climate change and economic conditions. In addition, historical data are of importance in demand forecasting. There is also a rising need for Power Suppliers to build their bidding strategies with their competitors so that later the consumers can derive a plan to maximize their utilities using electricity purchased from pool [2, 3].

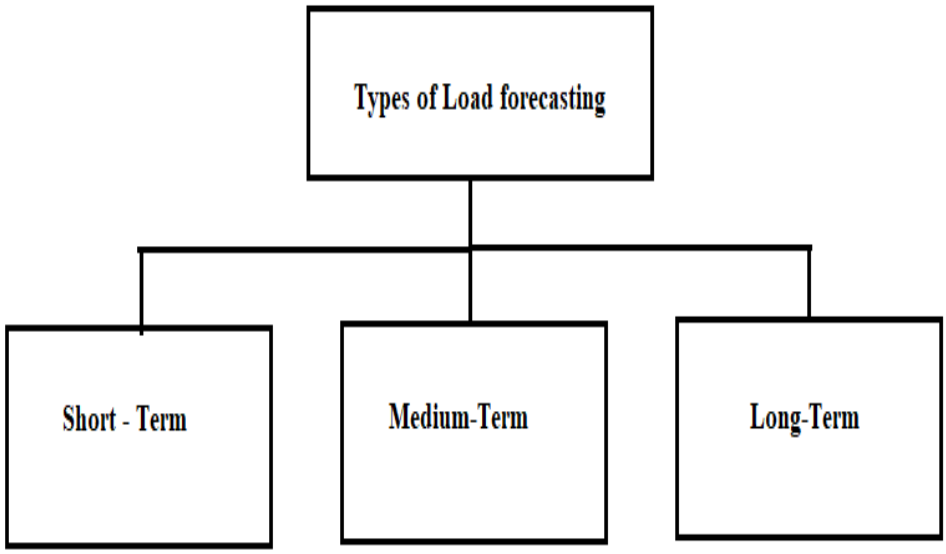

Figure 1.2. Types of Load Forecasting

Types of Load Forecasting

There are three different types of Load Forecasting, they are:

- Short-Term: Short-term forecasts are usually from one hour to one week. They play an important role in the day-to-day operations of a utility such as unit commitment, economic dispatch and load management. A short term electricity demand forecast is commonly referred to as an hourly load forecast.

- Medium-Term: Medium-term forecasts are usually from a few weeks to a few months and even up to a few years. They are necessary in planning fuel procurement, scheduling unit maintenance and energy trading and revenue assessment for the utilities. A medium-term forecast is commonly referred to as the monthly load forecast.
- Long-Term: Long-term electricity demand forecasting is a crucial part in the electric power system planning, tariff regulation and energy trading. A long-term forecast is required to be valid from 5 to 25 years. This type of forecast is used to deciding on the system generation and transmission expansion plans. A long term forecast is generally known as an annual peak load. In this Thesis, the main aim is to predict hour ahead load demands which falls under the Short-Term forecasting [2].

Advantages of Load Forecasting

Load Forecasting helps the Utility Companies to Minimize the risks by Understanding the future load Demands which helps the company to plan and make economically viable decisions in regard to future generation and transmission investments like for example, Utility Companies can set up Generation stations near to where the demand is particularly higher and reduce

the transmission costs. It helps the Utility Companies to plan for the scheduled maintenance of the Power systems. This proves the point as to why Load Forecasting is highly essential for Utility Companies, however there are complexities in Load Forecasting as well. Load Forecasting often takes Weather Data as one of the inputs for training the models but given the unpredictable nature of the weather provides a challenge, it sometimes gets tricky while forecasting the Load, given that the weather data set was also predicted by a forecasting model. However it is only tricky for those areas where the weather is highly unpredictable. The cost to supply electricity changes minute by minute. However, most consumers pay rates based on the seasonal cost of electricity. Changes in prices generally reflect variations in electricity demand, availability of generation sources, fuel costs, and power plant availability. Hence Load forecasting helps Utility companies plan out their cost of supply to their consumers well in advance [4, 5, 6, 7, 3].

Challenges in Load Forecasting

There are a number of different challenges faced by utility companies while load forecasting. Some of the most important factors are as follows:

- Highly volatile price and Load values that make the prediction process really hard

- Electricity cannot be transported from one region to another one because of existing bottle-necks or limited transportation capacity
- High percentage of unusual prices (mainly in periods of high demand) due to unexpected or uncontrolled events in the electricity markets.
- Unpredictable Weather in different seasons
- The fundamental problems are the variability and taxing schedule based on the nature of the demand [3].

Problem Statement

It is clear that Electric Energy is an important resource in this day and age. It is expected that by the year of 2040, there will be a rise in demand for Electric Energy up to 85 % in the United States, provided by the US Census. There lies the problem statement as to finding a way for Utility companies efficiently manage the Energy Demand, efficiently schedule their resources and come up with a pricing plan based on the Demand from various real life scenario's.

Objective

It is almost certain that there should be prior knowledge of the Load Patterns and that is where Load forecasting comes in handy and helps forecast the Energy demands using various machine learning predictions algorithms. Objective of this Thesis is to provide Utility Companies with insights as to which Supervised Machine Learning models are best suited for a given real life scenario in terms of Data set that is available to the Utility Companies.

Chapter 2

Literature Survey

Machine Learning

Machine Learning can be defined as the algorithm that has the ability to automatically learn from the Data and Observations and give out a Classification or a Prediction with just the Architecture being designed and not the actual program explicitly being coded. So why Machine Learning now when it was introduced in the late 1950's? There was a lack of Computational and Processing power to deal with Machine Learning Algorithms in the earlier years when they were introduced, also there was a lack of storage resources to run and store such taxing computational tasks. The availability of relevant Data sets were also scarce. Machine learning algorithms learn better as the number of features for that particular data set increased. Hence the Data was not enough for the models to train and efficiently learn to provide good results. The advent of Digitization age paved the way for Machine learning

to gain back its popularity. The wide range of Data that is available today to train the models made Machine Learning attractive in areas where it can be modeled to fit the desired application. The advancement in Computational, Processing power and Storage resources these models were able to run at a much faster pace and store Data with greater ease. Machine Learning also offers us a lot of flexibility to optimally tune the models and make them robust to be backwards compatible with any Data set that is fed into it and get accurate results. In the field of Machine Learning there are three methods as to how to program the models, they are Supervised machine learning, Unsupervised machine learning and Reinforcement Learning.

- Supervised machine learning is carried out with prior knowledge of the output samples or in other words, training samples are labeled as inputs and outputs. For Example, Neural Networks, Logistic Regression, Support Vector Machines are some of the Supervised Machine learning Models
- Unsupervised machine learning does not have labeled outputs, so its goal is to infer the natural structure present within a set of training samples. For Example, Self Organizing Maps, Auto Encoders,

Boltzmann Machines are some of the Unsupervised Machine Learning Models

- Reinforcement Learning or learning with a critic, no desired category label is given; instead, the only teaching feedback is that the tentative category is right or wrong. In other words, it is a type of learning where the weights are learned on the basis of a reward, where it finds the best possible reward. For Example, Markov process is one type of reinforcement Learning Model.

This research focuses only on Supervised machine learning models since the Utility companies have the required data set for training. This Thesis focuses only on Supervised Machine learning techniques not on Unsupervised Learning because in reality for Load forecasting one deals with labelled Data that are readily provided to us by the Utility Companies and the Independent System Operators which are made available online 24/7 [5]. Hence there is no real motivation to pursue towards Unsupervised learning. Neural Networks are a subset of Machine learning algorithms and this research explores three types of neural networks for load forecasting [8, 9, 10].

History of Neural Networks

Neural Networks are a subset of Machine Learning algorithms which are loosely based on the biological neural networks of the human brain. The human brain consists of three major parts namely Temporal lobe, Occipital lobe and the Frontal lobe and the neurons connecting them. Each of these parts along with the neurons served as influence and inspiration for modelling the various types of neural networks available to the present day. The first artificial neural network Preceptron was invented in 1958 by psychologist Frank Rosenblatt [11, 9], it was intended to model how the human brain processed visual data and learned to recognize objects. Preceptrons are also based on the neurons to model to human brain. In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. The idea was to combine a bunch of simple mathematical neurons to do complicated tasks, then came the general feed forward operation combining a bunch of preceptrons as input layer, hidden layers and an output layer. In the modern sense, the preceptron is an algorithm for learning a binary classifier as shown in equation (2.1) [8, 9]. For a simple preceptron the weights are initialized to zero at first and

then passed through the layers, then for each training sample an output was obtained from the unit step activation function. The weights are then updated based on the output values with a goal of minimizing the errors. The weights are all updated simultaneously in a layer [11].

$$X = \begin{cases} 1, & if\ w.x + b > 0 \\ 0, & otherwise \end{cases} \tag{2.1}$$

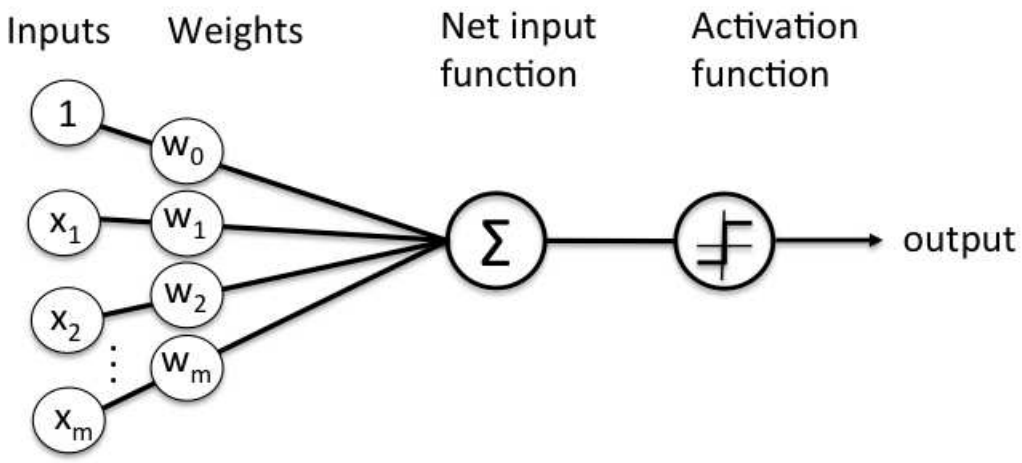

Figure 2.1. Schematic of a Rosenbaltt Preceptron [11]

Backpropagation

Backpropagation also plays a major role in neural networks. Backpropagation is one of the simplest and most general methods for supervised training of multilayer neural networks. Backpropagation was proposed and successfully implemented in neural networks by Paul Werbos in 1974 which slowly paved way for the different type of neural networks like the Artificial Neural Networks based on the Temporal Lobe, Convolutional Neural Networks based on the Occipital Lobe and Recurrent Neural Networks based on Front Lobe. The basic approach in learning is to start with an untrained networks, present a training pattern to the input layers, pass the signals through the net and determine the output at the output layer. The error function is some scalar functions of the weights and is minimized when the networks outputs match the desired outputs. Thus the weights are adjusted to minimize the error. The gradient descent algorithm with the cost functions in (2.2) was used to find the weights [8, 12, 9, 10].

$$J(w) = 0.5 * \sum_{k=1}^{c} (t_k - z_k)^2 = ||t - z||^2 \tag{2.2}$$

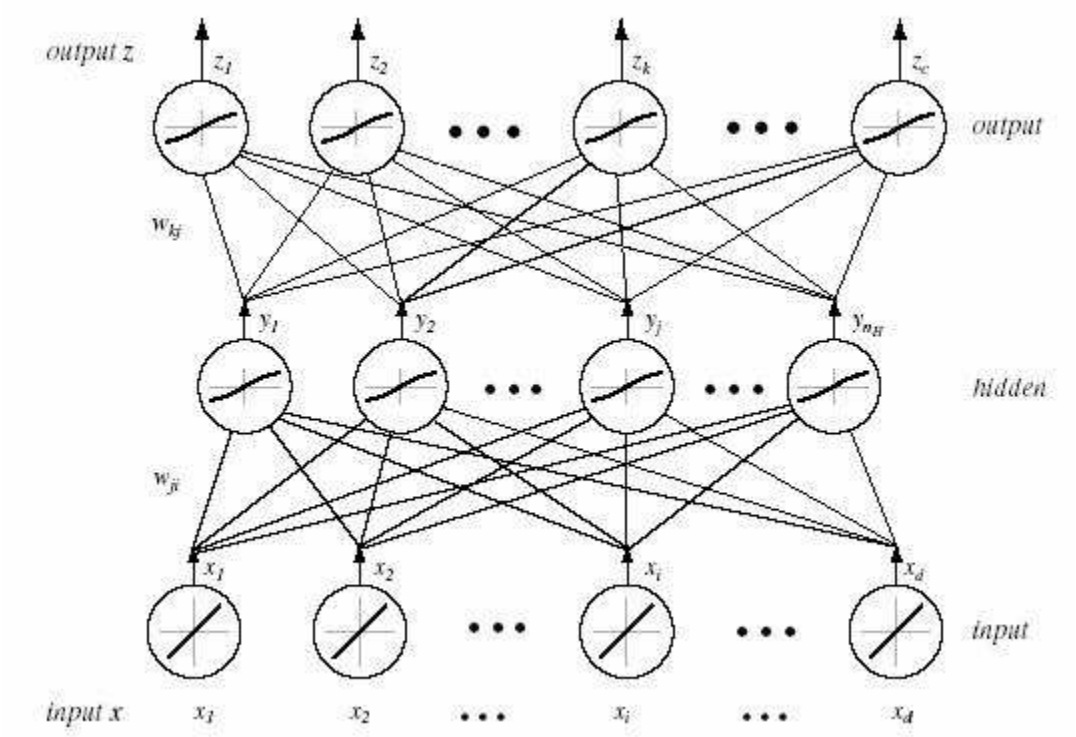

Figure 2.2. Back Propagation [8]

The weights are initialized at the start of training and then they are changed in a way that will reduce the error:

$$\delta w = -\eta \, \partial J / \partial w \tag{2.3}$$

where 'w' corresponds to the weights and 'J' is the cost function which is governed by the rate at which the weights are learned.

Activation Function

A neuron in a neural network computes the weighted sum of input and add a bias before passing into the activation function, which then decides whether the to send that value to the next neuron or not, From (2.1) Say Y is the Output of a neuron in a neural network so X corresponds to the input samples, W is the weight vector and b is the bias.

$$Y = \sum_{t=1}^{n}(W * X) + b \tag{2.4}$$

$$f(Y) = max(0, Y) \tag{2.5}$$

Backpropagation will work with virtually any activation function, given that few simple conditions such as continuity and that it is differentiable. One of the important properties of the activation function is they are non-linear and at the same time ensures continuity and smoothness. There are different activation functions like sigmoid, rectified linear unit (ReLU), tanh and so on. Sigmoid activation is use primarily in classification tasks where the output has to be a probability between 0 and 1. This Research deals with a regression task as it has to predict a continuous range of values. ReLU is best suited

for regression problems. ReLU activation function activates the output if the input is positive else send out a zero (2.5) [8].

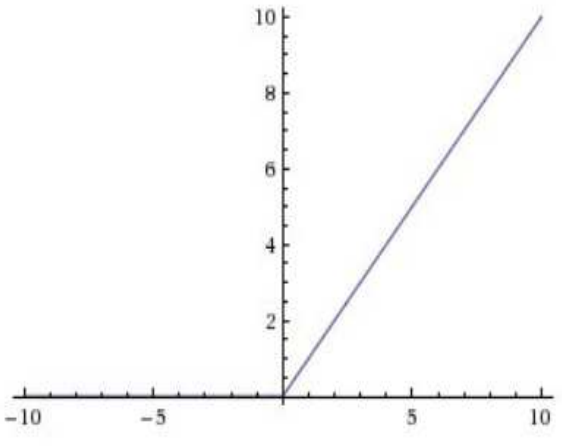

Figure 2.3. Activation Function - ReLU [8]

Loss Function

The basic approach in learning is to start with an untrained network, preset a training pattern to the input layer, pass the signals through the net and determine the output at the output layer. Here these outputs are compared to the target values; any difference corresponds to an error. The weights are then adjusted to reduce the measure of error. Neural networks requires a loss function to be defined to calculate the model error. Some commonly used loss functions are Mean Squared error, Mean Absolute Error, Categorical

cross entropy and so on. First two losses are suited for a regression problem whereas the latter for a classification task.

$$MSE = 1/n \sum_{t=1}^{n} (Actual_t - Predicted_t)^2 \tag{2.6}$$

From Equation (2.6) n refers to the number of samples in the input set, MSE corresponds to the mean of the difference between the actual load value and the predicted load value. MSE was used as the Loss function during the training phase to minimize the errors There is another loss function called Mean Absolute Error (MAE) used in regression analysis for neural networks. MAE is more robust to outliers since it does not make use of square. On the other hand, MSE is more useful if we are concerned about large errors whose consequences are much bigger than equivalent smaller ones.

$$MAE = 1/n \sum_{t=1}^{n} (|Actual_t - Predicted_t|) \tag{2.7}$$

It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. [6, 9].

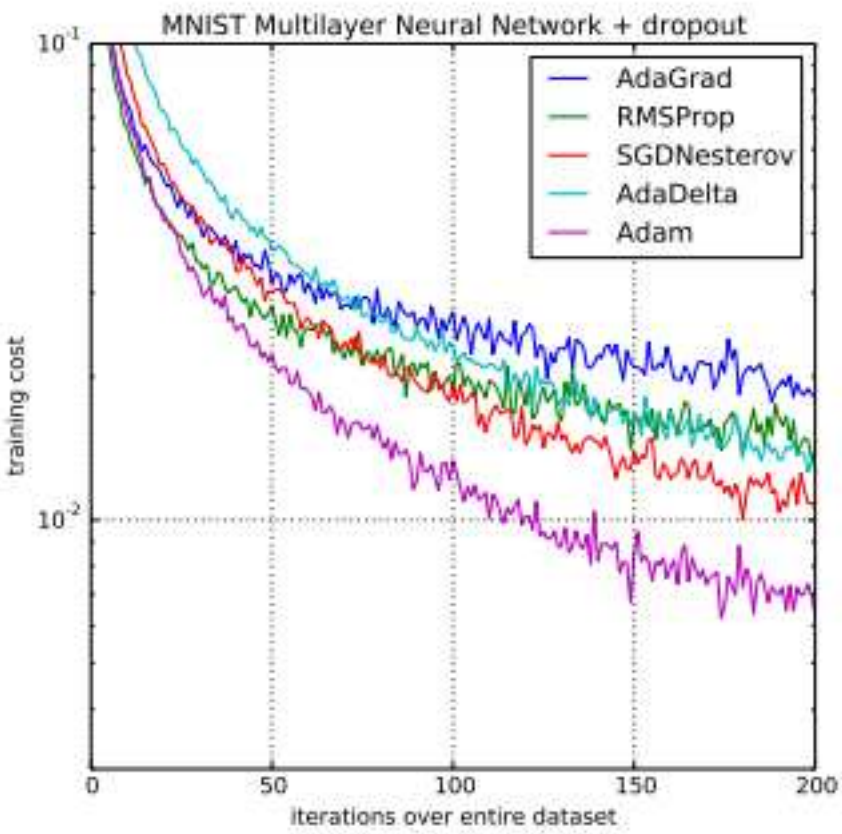

Figure 2.4. Overview of Gradient Descent Algorithms [13]

Optimizer

Optimization algorithms are responsible for minimizing or maximizing the error/loss function stated in the section above and update the system weights. There are different type optimization algorithms like Adam, RMSprop, adelta, adamax, gradient descent, stochastic gradient descent and so on. By experimentation it was found that Adaptive Moment Estimation (Adam) optimizer was well suited for this research problem. Adam was pre-

sented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR Thesis (poster) titled "Adam: A Method for Stochastic Optimization" [14]. Adam combines two optimization algorithms namely the adaGrad and RMSprop. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Adam is regarded as on of the most effective algorithms in Machine Learning as it achieves good results in a short period of time. The compiler for the training set in neural networks requires both the Loss function and the Optimization algorithm to be defined. [13, 9] Sebastin Ruder performed a benchmark on MNIST data set to evaluate the performance of each optimizer reviewed in that paper as shown in Fig (2.4). It is evident from the diagram that adam and RMSprop were the most effective in terms of optimization in regression analysis.

Scoring Criterion

The Machine learning models in this research are subjected to a scoring mechanism to evaluate their performances. Absolute Mean Percentage Error (MAPE) was the primary scoring mechanism employed in the Test set.

$$MAPE = 1/n \sum_{t=1}^{n} |\frac{(Actual_t - Predicted_t)}{Actual_t}| * 100 \tag{2.8}$$

MAPE is a measure of how close the predicted values are to the actual values. MAPE is most commonly used scoring mechanism for regression tasks to evaluate model performance. R2 score otherwise known as coefficient of determination was also used as an evaluation metric for the training and testing accuracies. R2 also gives us a statistical measure of how close the given data is to the actual fitted regression curve. It is one of the most populous evaluation metrics for continuous data set or regression score function. It can also be defined as the proportion of the variance in the dependent variable that is predictable from the independent variables [4, 6, 7].

$$R^2 = 1 - \frac{SSres}{SStot} \tag{2.9}$$

$$y_m = 1/n * \sum_{i=1}^{n} y_i \tag{2.10}$$

$$SStot = \sum_i (y_i - y_m)^2 \tag{2.11}$$

$$SSres = \sum_i (y_i - f_i)^2 \tag{2.12}$$

from the above equations from (2.9 - 2.12), y_i is the true value, f_i is the predicted value and ym is the mean of the samples.

Bias - Variance Trade off

In almost every Machine learning model it is important to understand the prediction errors. Bias and Variance Trade off helps to understand these errors better.

- Bias refers to the accuracy or quality of the match. It can also be defined as the squared difference between the expected value and the true value.
- Variance is the precision or specificity of the match. Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.

The bias-variance trade-off is a general term in Machine learning models where models have procedures with increased flexibility to adapt to the training data. Models with high bias pays very little attention to the training data and oversimplifies the model. Models with high variance pays a lot of attention to training data and does not generalize on the data which it has not seen before. Now comes the problem of overfitting the data and underfitting the data.

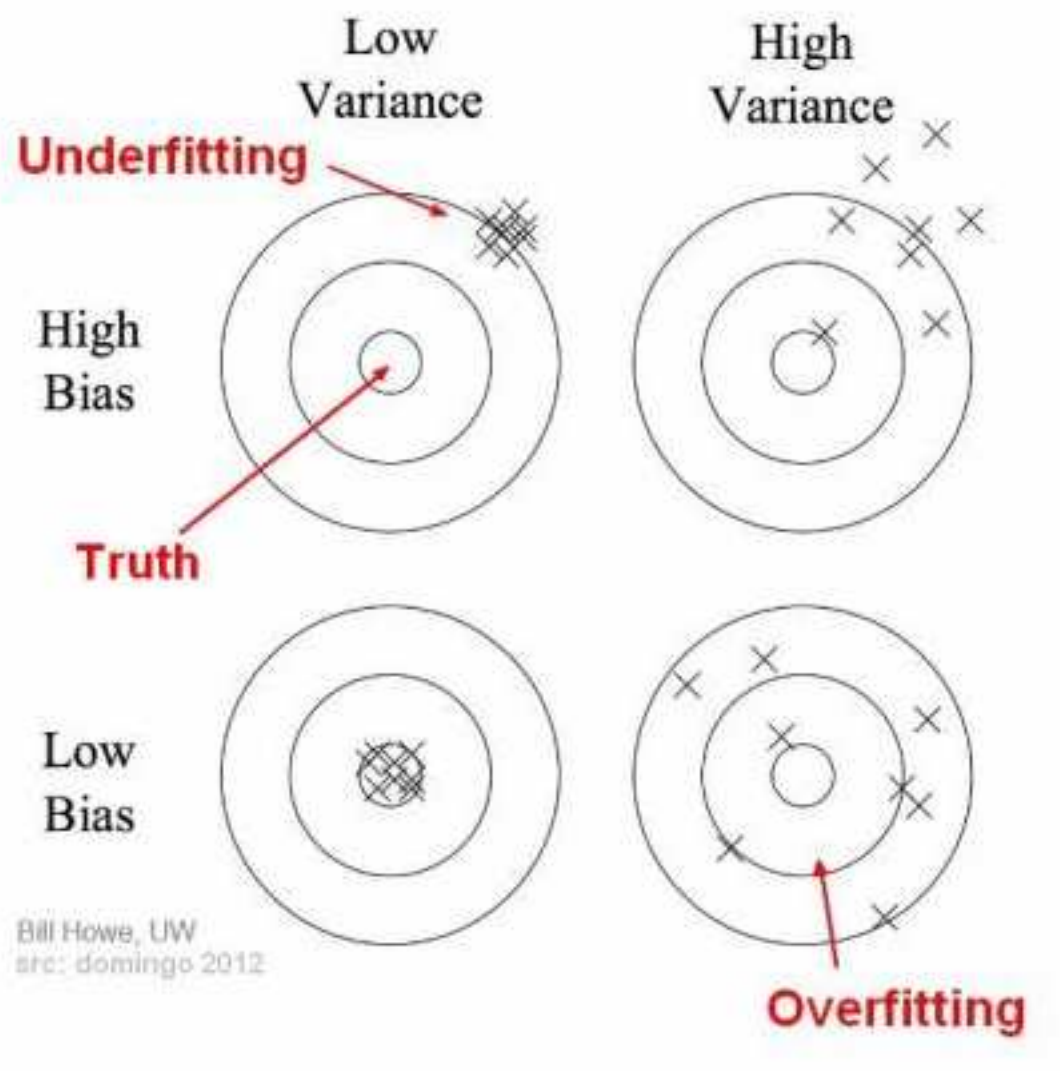

Figure 2.5. Bias - Variance [15]

- Overfitting: Overfitting occurs when a complex system follows a perfect classification or regression on the training data but fails to perform well on the actual test data. It occurs when the model is

trained on a noisy data set and they tend to have low bias and high variance.

- Underfitting: Underfitting happens when a model is unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It usually occurs when the amount of training data available is scarce [16].

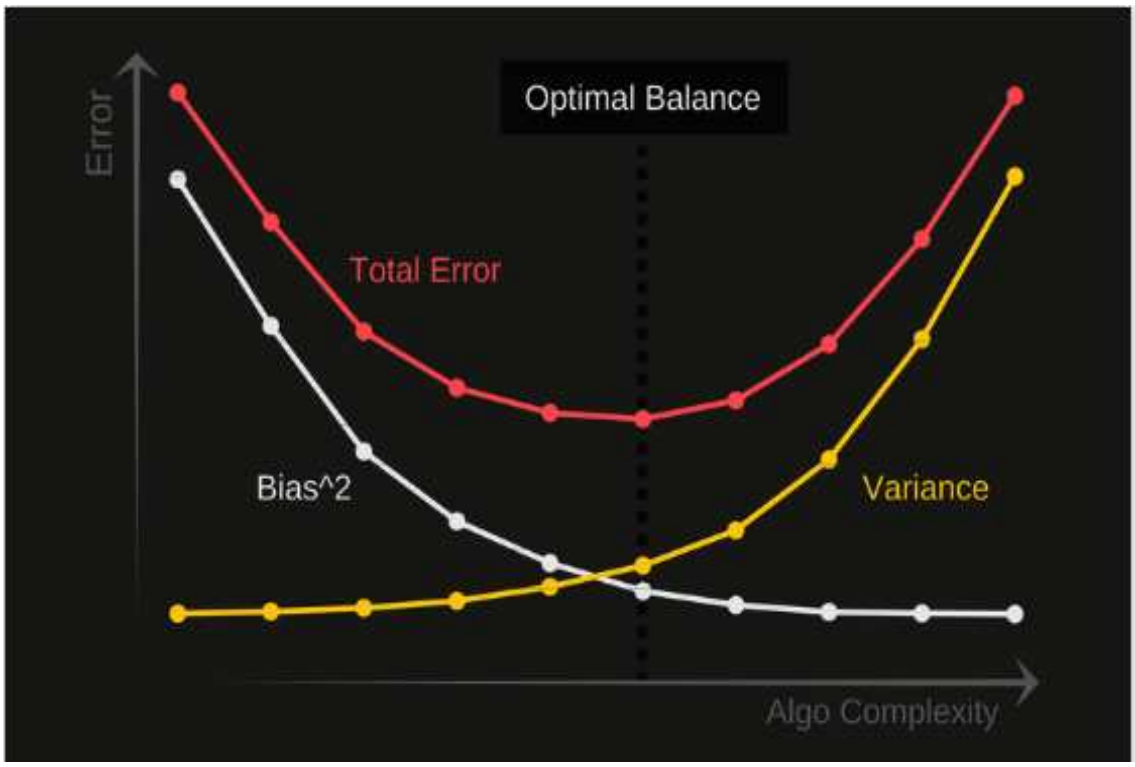

Figure 2.6. Optimal Trade off [15]

The goal is to find the optimal balance in the bias variance trade off, so that it minimizes the error for the case. Thus it is extremely important to understand the trade off for prediction problems.

Methods to overcome - Overfitting/Underfitting

There are different methods that help the machine learning models overcome the problems of Overfitting and Underfitting. This Thesis uses the following methods to help overcome this problem:

- Train-Test-Validation Split
- Cross-Validation
- Lasso and Ridge Regression
- Dropout

Train-Test-Validation Split

It is common practice to split the entire data set into a Training Set and a Test set. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test data set (or subset) in order to test our model's prediction on this subset. The split is usually around 70/30 for the training and the Test sets. Furthermore

the Training set is subjected to K-fold cross validation to split the training set into a part for training and validation across the validation set [8].

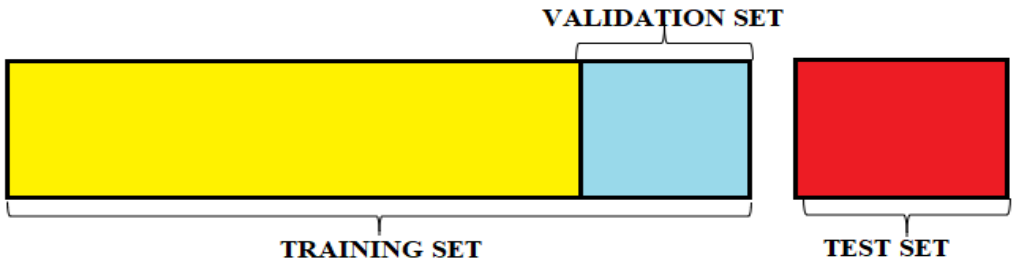

Figure 2.7. Train - Test - Split

Cross-Validation

Cross validation is one of the many techniques to tackle the Overfitting/Underfitting problems that the Machine Learning models encounter in general. It helps measure the stability of the model. In k-fold cross-validation, sometimes called rotation estimation, the data set D is randomly split into k mutually exclusive subsets (the folds) D1,D2,....Dk of approximately equal size. It is split into k folds as given by the user. The model is trained on the split subsets and tested only on the k^{th} subset and the process is repeated for each fold as the test set. It is usually done on the Training Data with Training

and Validation split done beforehand. In the figure below shows a data set with k = 5 and being cross validated over 5 folds [17, 16, 8].

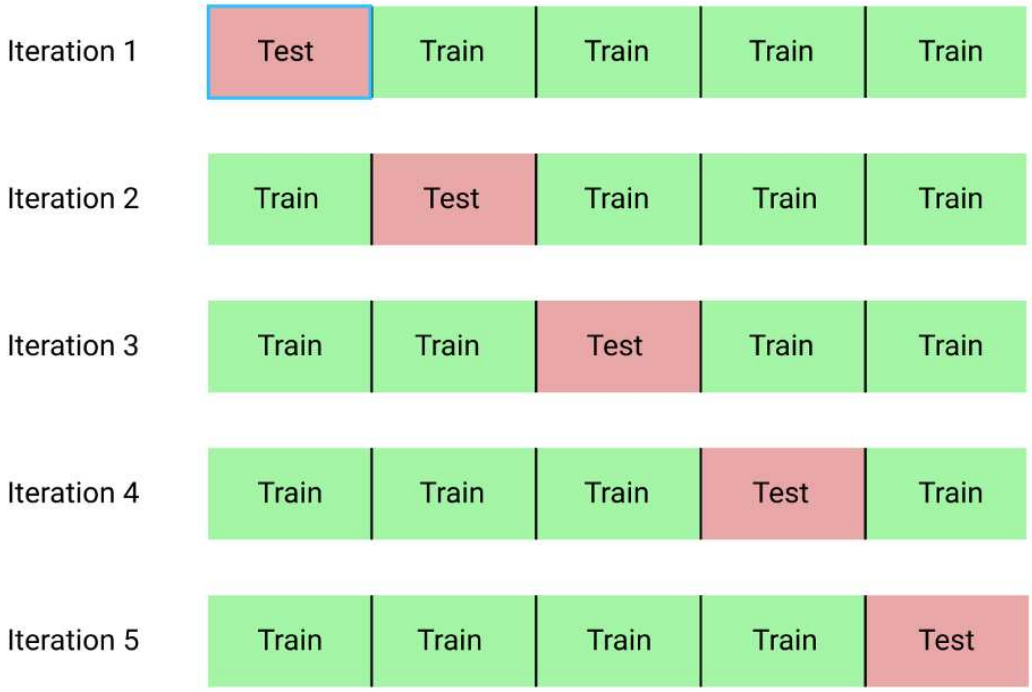

Figure 2.8. K - Fold Cross Validation [18]

Lasso and Ridge Regression

Lasso and Ridge Regression are the most commonly used regularizers for regression analysis when it comes to overfitting and underfitting. They are also referred to as L1 and L2 norms respectively. They are use to make the models Robust. It is used as a part of the regularizing term in neural networks while determining the weights which is given as follows:

$$w^* = argmin(ErrorFunction) + \lambda \sum_{i=1}^{k} (|w_i|) \tag{2.13}$$

$$w^* = argmin(ErrorFunction) + \lambda \sum_{i=1}^{k} (w_i)^2 \tag{2.14}$$

The equation (2.13) corresponds to the L1 norm and equation (2.14) corresponds to the L2 norm. As stated before there is a change only in the regularizing term while learning the weights for both the norms no matter what the error function is.L1 norm is the sum of the normalized values of the weights whereas the L2 norms is the sum of the squared values of the weights. L2 norms are computationally efficient due to analytical solutions on the other hand L1 norms have sparse cases hence it is inefficient. L1 norms have built in feature selection whereas the other does not. Sparsity refers to the non-zero entries that are very scarce in a matrix or a vector. Feature selection refers to the ability of the model to select only useful coefficients that contribute as useful features for the model to train on [19].

Dropout

Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During

training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods.

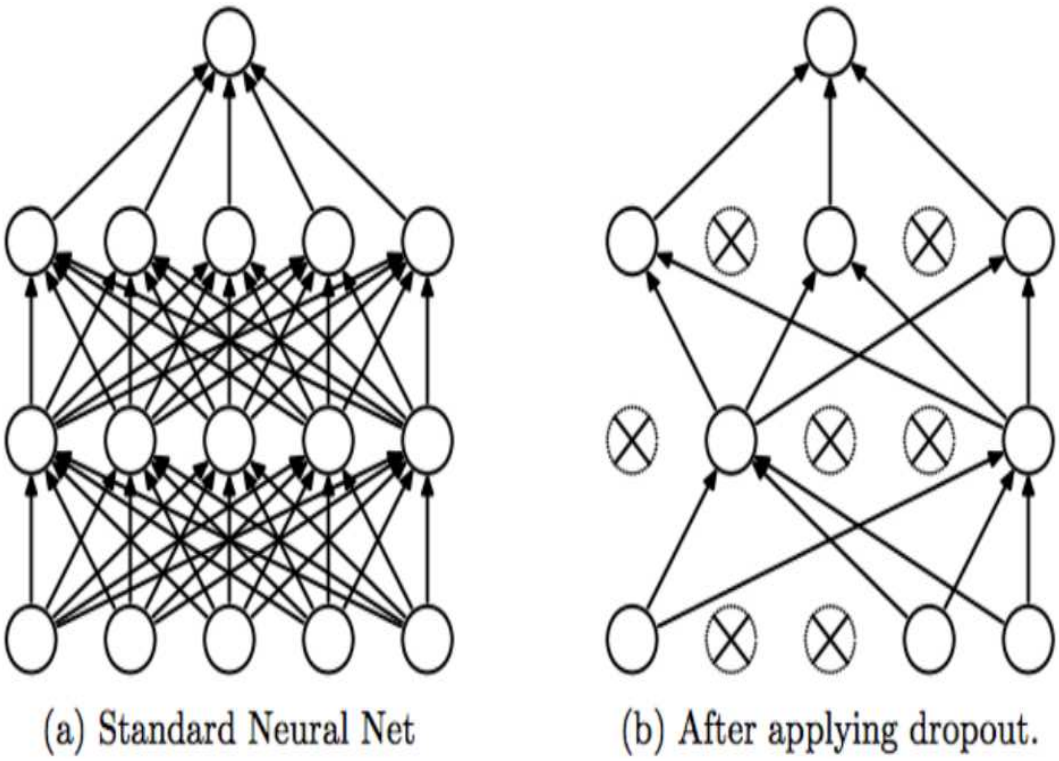

Figure 2.9. Dropout [20]

Tools and Resources Used

The problem statement gives the idea that the problem at hand is Data processing/Statistical analysis problem hence, the reason for choosing python as a standard coding language.

- Programming Language Python version - 3.6.8.
- Integrated Development Environment - Spyder IDE version - 3.3.2.
- Packages Installed - Tensorflow version - 1.12, Keras version - 2.2.4, Numpy version - 1.15.4, scikit-learn version - 0.20.2.

Tensorflow

Tensorflow is a python package available for free licensed by Apache License 2.0. TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It was developed by Google Brain Team [21].

Keras

Keras is also a Python package available for free licensed by MIT. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or

PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible [22].

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays licensed by BSD [23].

scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to inter operate with the Python numerical and scientific libraries NumPy and SciPy licensed by the New BSD [24].

General Architecture

The General Architecture of the models is shown in Fig. 2.10. The first task is to find the right Data set to start analyzing and decide on the right amount required for training. It is always good practice to obtain the Data

and carefully scan them and look for missing values and irregularities. Every model goes through the initial process of preparing and cleaning the Data. Cleaning the data refers to carefully combing through the data in search of missing Data point or irregularities and getting rid of them either by finding the right data or averaging the data that is available. Then it is subjected to Normalization/Standardization in the Data pre-processing phase. Then the Data is split into the Training Set and the Test Set. Validation Set is obtained from the Training Set as a split form different folds of the K-Fold Cross validation. The training set is fed into the model for training and carefully tuned on the validation set based on the scoring criterion for the problem at hand, in this case it will be the best possible MAPE and R2 score. When the training is done, the results are predicted using the Test Set. These predicted Test set Values are then compared with the Actual Load values using MAPE and R2 for performance Evaluation. The real load values and the predicted load values are plotted on a hourly basis to get a visual representation of how good the prediction actually is.

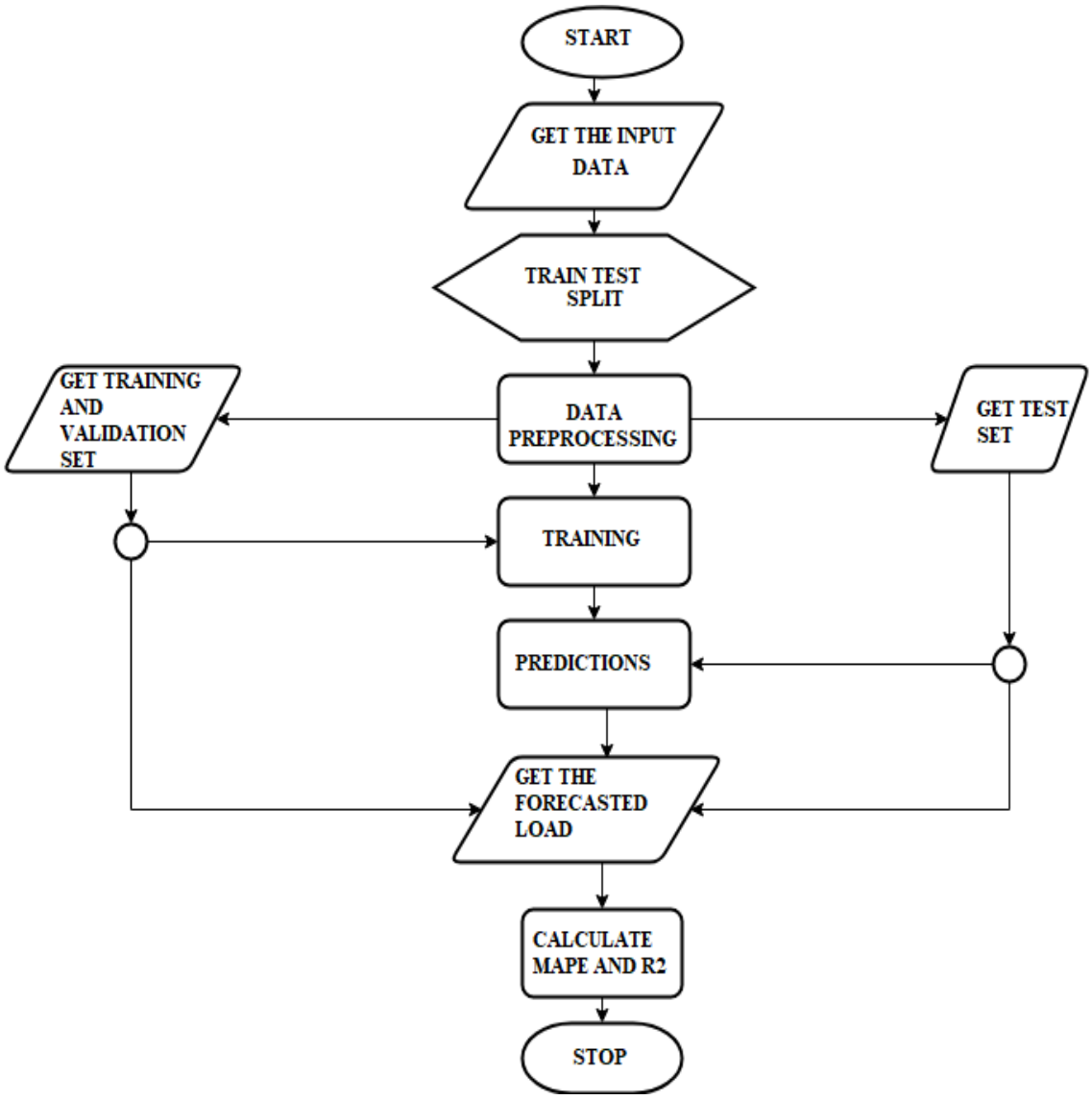

Figure 2.10. General Architecture

Chapter 3

Dataset

Machine Learning algorithms required adequate Data sets for the models to train and make predictions. The Real Load Data set used in this research comes from Electric Reliability council of Texas (ERCOT) [25]. It comprises of Real load values that was supplied to the consumers in the city of Dallas Fort-worth for the entire year for 2018. The load values were obtained for every hour for the year of 2018. The weather plays an important role as to how to load pattern behaves so it was only natural to include the weather data in the input data set. The hourly weather data was obtained from the National Oceanic and Atmospheric Administration (NOAA) which is also centered around the Dallas Fort-Worth Area [26]. These data were properly indexed by the Date and time. There are different types of variables that are encountered in this data set. They are as follows

- Independent Variable: It is a variable that stands alone and is not changed by the other variables that one is trying to measure. It can

also be defined as a variable that is changed or controlled in a scientific experiment to test the effects on the dependent variable. For example, Weekend or Not and Lagged Load have no dependencies on each other but they influence the dependent variables which is the actual Real Load values that is to be predicted.

- Dependent Variable: Variable that depends on other variables from the data set. It can also be defined as a variable being tested and measured in a scientific experiment. For example, the real Load values from the data set that is to be predicted is the dependent variable that depends on rest of the other variables listed in the table (3.1) from 1 through 7.
- Categorical Variable: Categorical variables take on values that are names or labels. From Table (3.1) Hour of the Day, Holiday or Not, Weekend or not are all categorical variables that are just labels and do not have a numerical or quantitative significance.
- Quantitative Variable: Quantitative variables are numerical. They represent a measurable quantity. From Table (3.1) the Lagged Load by 24 Hours, Average Load from 24 Hours ago, Weather data set and the actual Load Demand all belong to this category.

Data set Preparation

The real load values was lagged for 24 hours from the current hour of the target dependent variable [4]. The load values were also averaged for the previous 24 hours for the current hour of the target dependent variable. Load values depend on the type of day as well whether it is a holiday or a weekday hence was included as part of the Data set. The data set was carefully scanned and cleaned of any missing values [27].

Data set Pre-Processing

Table 3.1. Dataset Table

Index	Features
1	Date (MM/DD/YYYY)
2	Hour of the Day (0 – 23)
3	Holiday or not (0 or 1)
4	Weekend or not (0 or 1)
5	Lagged Load by 24 Hours (MWh)
6	Average Load from 24 Hours ago (MWh)
7	Weather Dataset
8	Actual Load Demand from ERCOT

Data Pre-processing was used to get rid of the noise and irregularities that existed in the data set. It may also be defined as a data mining technique where it transforms raw data into a format that is understandable by the Machine learning models. It is clear from Table 2.1 that the Actual Load data

set is the Target Dependent Variable to be predicted, the Index 3 and 4 from the table correspond to the categorical variables of the Data set. It is common practice to center the data set of features around zero and then normalize it. The Load values and the Weather data set were normalized with zero mean and unit variance [28].

From Equation (3.1), x refers to the input samples, u is the mean and s is the standard deviation of the data set [28]. Below is a sample Table of what actually the scaled values of the load for StandardScaler looks like. There is another way in which they Data can be pre-processed which is the MinMaxScaler, where it transforms features by scaling each feature to a given range. This is an alternative to zero mean and unit variance scaling.

$$Standardization = (x - u)/s \tag{3.1}$$

$$X_std = X - X.min(axis = 0))/X.max(axis = 0) - X.min(axis = 0) \tag{3.2}$$

$$X_s = X_std * (max - min) + min \tag{3.3}$$

Table 3.2. Standard Scaler

Load Values	StandardScaler
34670.1	-0.663994
33798	-0.816887
33496.2	-0.869796
33961.7	-0.788185
35922.8	-0.444387
39209.8	0.131844
40373.5	0.335854
40823.9	0.414815
42388.5	0.689112
44054.3	0.981134
45842.8	1.29469

from the equation (3.2) and (3.3) is for the MinMaxScaler where min and max and user input feature range values. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. Table (3.3) shows the scaling for a MinMaxScaler for the load values when the user input feature range is (-2,2) [29].

The Weekend or not and Holiday or not data were encoded using a combination of one One Hot Encoder and Label Encoder to establish the fact that they are only categorical variables [30].

Table 3.3. MinMaxScaler

Load Values	MinMaxScaler
34670.1	-1.15055
33798	-1.2559
33496.2	-1.29236
33961.7	-1.23613
35922.8	-0.999223
39209.8	-0.602155
40373.5	-0.461576
40823.9	-0.407166
42388.5	-0.218154
44054.3	-0.016928
45842.8	0.199135

Label Encoding

Label Encoder's are used to convert categorical data into a numerical format that is understandable by the Machine learning models. It is used to encode variables with values from 0 to N-1, where N corresponds to the number of variables/classes. The problem here is, since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 <1 <2. But this is not the case at all. To overcome this problem, we use One Hot Encoder.

Table 3.4. Label Encoding

Index	Holiday or not	Label Encoding
07/04/2018	Yes	0
07/05/2018	No	1
07/06/2018	No	1
07/07/2018	Yes	0

One Hot Encoder

One hot encoding is a process by which categorical variables are converted into a form that could be provided to Machine Learning algorithms to do a better job in prediction. One hot encoding is used as a Binarization technique for categorical variables to include in the training models

Table 3.5. One Hot Encoding

Index	Holiday or not	One Hot Encoding
07/04/2018	Yes	0 0
07/05/2018	No	0 1
07/06/2018	No	0 1
07/07/2018	Yes	0 0

The entire Data set is split into Training Set and Test Set. The Training Set containing the first full nine months of Data and the Test set having the last three months. The Split is about 67 percent for the training set and 33 percent for the Test Set. Further the Training set is subjected to K-Fold

cross validation where the Model is tuned on the Validation Set for each Fold based on the value of K. By experimentation the models are validated with 5 folds of cross validation to obtain the best possible accuracy for the model to generalize. The main reason for splitting the Data set is to tackle the problem of Over fitting [8]. The Test Set is left undisturbed until the model is properly tuned on the Validation set. The Test set is only used to evaluate and compare the accuracy of the different models explored in this Research. The training set consists of 6552 samples and the Test set with 2208 samples [27]. Some of the Machine Learning models in this Thesis use only part of the data set and the others use the entire data set based on the scenario and desired application in hand. Univariate and Multivariate models present two approaches in statistical analysis.

- Univariate model: Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it
- Multivariate model: Multivariate analysis is the analysis of three or more variables. There are many ways to perform multivariate analysis depending on your goals.

Chapter 4

Supervised Machine Learning Models

This research explores six types of Supervised Machine Learning models for Short-Term Load forecasting. The five types are as follows:

1) Artificial Neural Networks

2) Convolutional Neural Networks - Univariate

3) Convolutional Neural Networks - Multivariate

4) Recurrent Neural Networks - Univariate

5) Recurrent Neural Networks - Multivariate

6) Support Vector Machines - Regression

Artificial Neural Networks

Artificial Neural Networks are based on the Temporal Lobe of the human brain as stated before. It takes it's influence from the temporal lobe's ability to retain long term memories or its ability to learn from past experiences. The ANN model was first designed with the use of only the Load values to see how

they perform. As expected the ANN model performed poorly with only the load data set for training. The model performed poorly in terms of the MAPE as well so it is definitely not the go to model when it comes to univariate or a single feature training. This proves the fact that the ANN models benefits more when the actual data set of features increases. Now the ANN model was

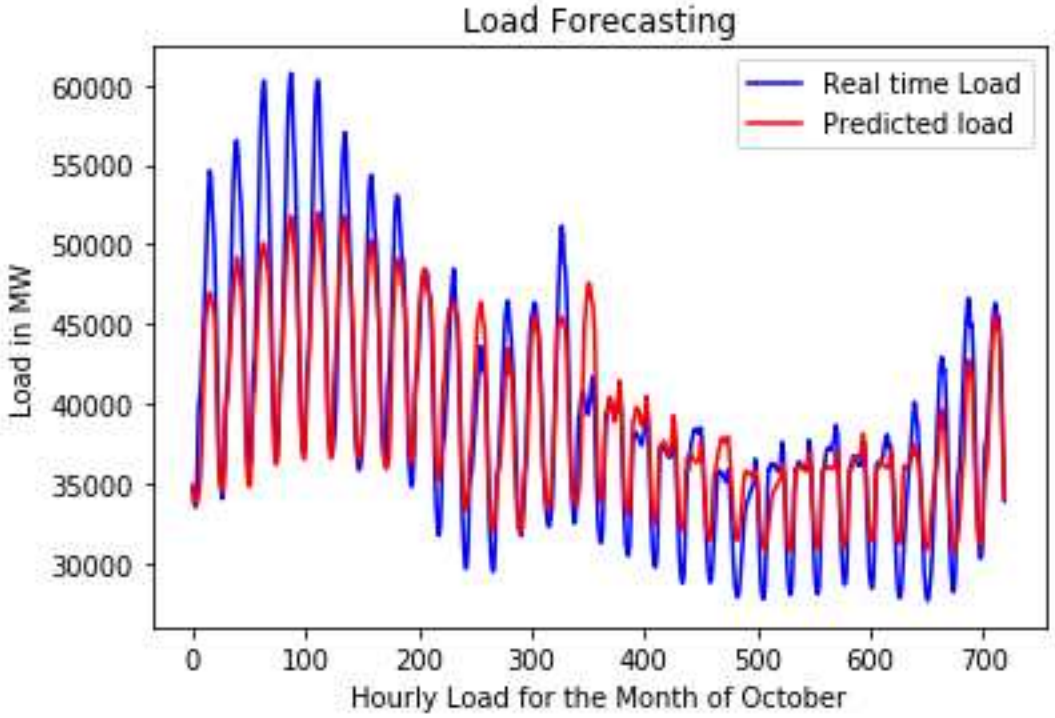

Figure 4.1. ANN - Univariate

designed to fit the scenario where the data set comprises both the Load values and their corresponding Weather Data. Smart City or a Smart Grid is the perfect example where it has both the Load values and the weather data set.

Thus the reason for choosing both the Load and Weather Data set as input to Train, Validate and Test. The ANN model consists of fully connected Dense layers of one input layer, one hidden layer and a single value output layer. The hidden unit had about 20 neurons in the layer [8, 9]. If there are more than

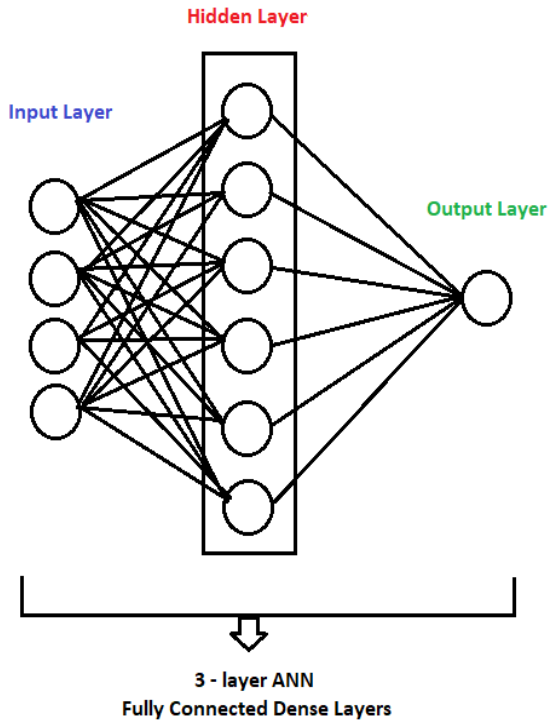

Figure 4.2. ANN

three layers in a Neural Network it corresponds to a Deep Learning which is not required in this case as the number of input samples are small when

compared to the input samples used in Deep Learning models. The ANN is a feed forward neural network where the data travels only in one direction from the input to the output. The weights are updated using the Back Propagation algorithm. These layers are fitted with Dropout Regularization to tackle Over fitting to about 20 Percent and a rectified linear activation function to deal with continuous values and also introduce non-linearity to the model [20]. The layers of this model are subjected to L1 norm or regularization as the kernel regularizer and L2 norm as the activity regularizer which helps tackle over fitting [19].

The model is compiled with an Adam optimizer and mean squared error (MSE) loss with five folds of cross validation and thoroughly optimized with the validation set for optimal result without over fitting. The plot for the Load values of the Actual against the predicted for the month of October from the Test set is shown in Fig (5.1) and the error score of MAPE average equal to 6.45 % was for the entire three months of the Test set [4]. ANN is the type of model that learns well from prior experiences or epochs and updates the weights after each epoch. The more meaningful data set it has as inputs to train, the better the accuracy. Hence, ANN model is best suited for Smart Cities or Smart Grids where they have ample amounts Load data and good Weather data to get short term predictions [31, 32].

Convolutional Neural Networks - Univariate

The architecture of a CNN is analogous to that of a connectivity pattern of Neurons in the Human Brain and was inspired by the organization of visual cortex. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The feed forward operation of the network during recognition is the same as in standard three-layer networks, but because of the weight sharing, the final output does not depend upon the position of the input pattern. Due to lesser parameters, CNN can be trained smoothly and does not suffer over fitting. A general model of CNN consists of four components namely convolution layer, pooling layer, activation function, and fully connected layer as shown in the figure. The weight vector, also known as filter or kernel, slides over the input vector to generate the feature map. This method of sliding the filter horizontally as well as vertically is called convolution operation. This operation extracts N number of features from the input image in a single layer representing distinct features, leading to N filters and N feature map. Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. Next come the Dimensionality reduction where the results are down sampled or pooled feature maps

that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling. This has been found to work better in practice than average pooling for computer vision tasks like image classification. Fully connected layer is similar to the fully connected network in the conventional models. In order to introduce non-linearity, use of Rectified Linear Unit (ReLU) has proved itself better than other activation functions [8, 33].

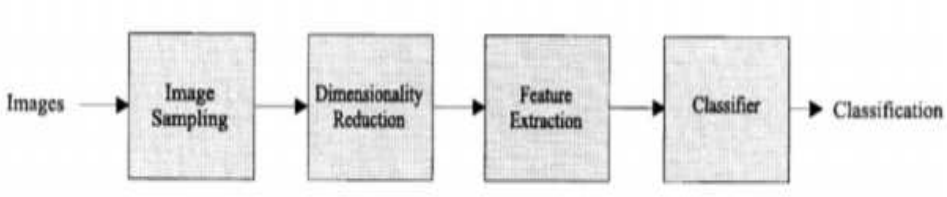

Figure 4.3. General model of CNN [33]

1-Dimensional CNN

1-dimensional CNN's were derived from techniques from LSTM and DSP. 1-D CNN's use the mathematical convolutions from signal processing where two signals are integrated with one being time flipped also known as discrete time convolutions. In 1-D CNN's it uses vectors as inputs and outputs rather than matrices that are involved with 2-D CNN's. Convolution Neural Networks is mainly used for Classification Applications where most of the

Data are images which are converted to a 2-dimensional array from which the model learns. In this case, the inputs are just 1-dimensional continuous values which is the Load data set (Univariate), hence, it is enough to design a 1-dimensional convolutional layers to provide time series predictions. The input layer had 8 filters with a kernel size of 3 and the hidden layer had 16 filters with a kernel size of 3. Input fed into a CNN are to be processed according to the batch size, the number of steps and the number of channels to fit a 3D tensor shape of the layers as required by convolutional layers. The input were the previous 24 hours from the current real load and trains on the 25th hour as output. The CNN layers are also subjected to L1 and L2 norms and Dropout classes to tackle over fitting [19]. CNN model just before the output layer must flatten the 3D tensor to get the appropriate value for the output layer. The CNN model was fed with the input which is the training set and validated on different folds of the cross validation to optimally tune the model. The CNN model after being fitted on the training set was used to predict the Load values using the inputs from the Test set. The MAPE average score for the entire Test set was about 2.98 % and the plot for the Load values of the actual to the predicted for the month of October from the Test set is shown in Fig. 3. CNN is well suited only when we have the Load data set available for manipulation. CNN models of this kind will be best suited for scenario's

where the load data set is the only data available for training, like a remote village or a hilltop where weather data set is not recorded or when the weather data set is highly unpredictable due to the region it is located. [34, 35, 5].

Convolutional Neural Networks - Multivariate

Multivariate can be defined as two are more variable quantities, where here it refers to all the features from the Data set. The input layer had 8 filters with a kernel size of 3 and the hidden layer had 16 filters with a kernel size of 3. The output layer had a single neuron for the predictions. Multivariate CNN is similar to Univariate in Architecture with minor changes in they hyper parameters. Multivariate CNN after being trained on all the features from the Dataset returned an average MAPE score of 4.10 percent which is almost close to that of the Univariate model but not an improvement over the previous in terms of MAPE score and the R2 score.

This model is designed based on an assumption that it has all the Data set for Load Forecasting, which is a type of scenario that only exists in Smart Cities and Smart Grids or other booming cities like New York where the weather and Load data set are readily available. [34, 35, 5].

Recurrent Neural Networks - Univariate

RNN is based on the Frontal lobe of our brain where it learns from short term impulses or memories for example, hyphenated phone numbers which a human remembers for a short period of time. RNN is not a feed forward neural network as it has in it's architecture Long Short Term Memory (LSTM's) which allows the models to learn from recent experiences. LSTM's are capable of learning long-term dependencies. RNN's introduce feedback into their networks. It is one of the most populous time series prediction models. The recurrent "unfolded" architecture as shown in Fig (4.4), has output unit values fed back and duplicated as auxiliary inputs, augmenting the traditional feature values. Recurrent networks have proven effective in learning time-dependent signals whose structure varies over fairly short periods, thus the error gets diluted when passed back through the layers many times [33, 8, 36].

LSTM

LSTM's were designed to mitigate the vanishing gradient and exploding gradient problem. Each LSTM cell has a Cell state vector C_t so that the next LSTM can choose to read, write or reset the cell using an explicit gating mechanism. There are three gates in each LSTM cell as binary gates. The three gates are the input gate i_t which decides whether the memory cell is

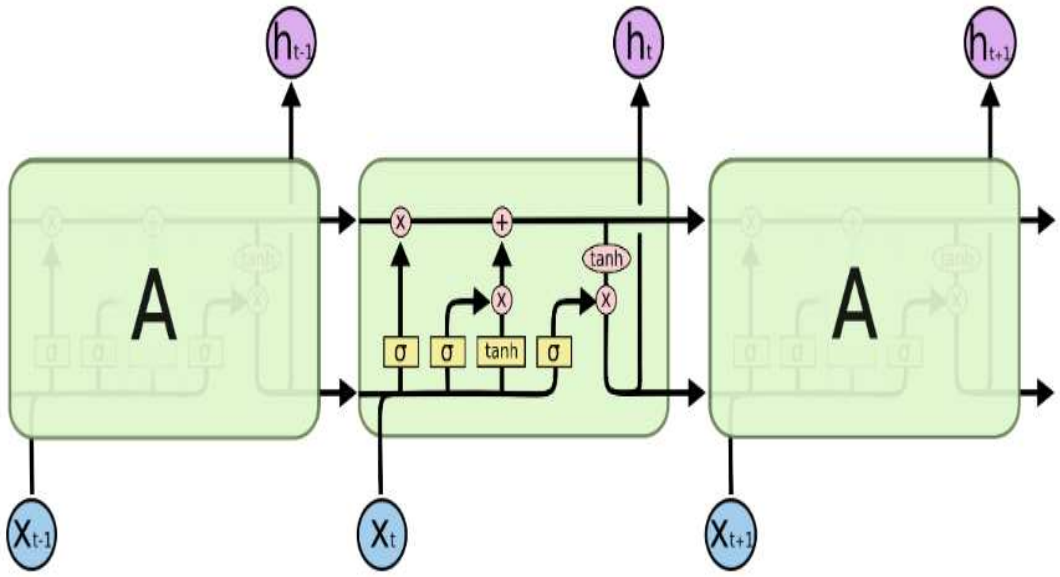

Figure 4.4. LSTM [37]

updated, the forget gate f_t controls whether the memory cell is reset to zero and the output get h_t controls whether the information of the current cell state is made visible or not. The three gates are based on a sigmoid activation function because the constitute a smooth curve from zero to one and the model variable is differentiable. Apart from these three gates there is one other vector $\bar{C}_t$ that modifies the cell state with a tanh activation function because with a zero centered range a long sum operation will distribute the gradients well which in turn prevents the vanishing/exploding gradient problem. Each of the state takes the hidden state and the current input x as the inputs. h_t state is applied to the output gate to get the hidden vector. [38]

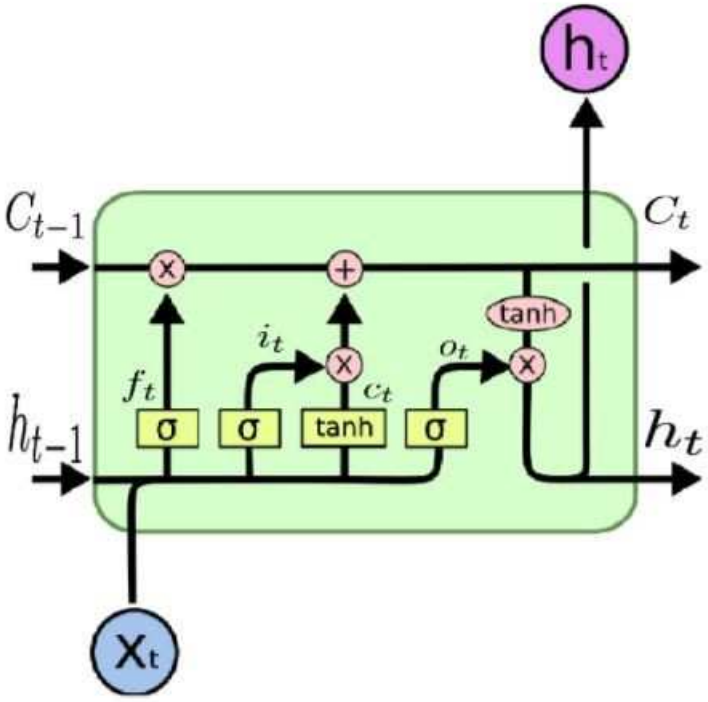

Figure 4.5. LSTM - Cell [37]

RNN basically updates the weights and backpropagates from the short timestamps of its previous inputs. The long short-term Memory (LSTM) paved the foundation for RNN [36]. RNN's input must be processed in a way so it corresponds to a 3D tensor input with the Samples, time stamps and features. The difference between CNN and RNN is that RNN can handle data with unknown lengths or in other words it can handle dynamic lengths for both inputs and outputs. The RNN had three layers the input, hidden and

the output layer. The input layer had 50 units the hidden had 100 units and a single neuron in the output layer for the load predictions.

RNN can handle sequential data whereas CNN cannot handle it. Input here is the same as it was in CNN using the previous 24 hours Load data as the dependent variable and the 25th hour acts as the Independent variable to be trained on and this recurs for the entire data set. It has one input LSTM layer with two hidden layers with Dropout classes to deal with Over fitting and one Dense output layer for the output [20]. These layers are also subjected to the L1 and L2 norms like for the ones that were employed earlier in ANN to tackle Over fitting [19]. The RNN model was fed with the input which is the training set and validated on different folds of the cross validation to optimally tune the model. The RNN model after being fitted on the training set was used to predict the Load values using the inputs from the Test set. The resulting average MAPE was about 2.44 percent for the Test set. RNN takes a longest time to train when compared to the other models. RNN's accuracy improves when working on bigger Data sets. Taking in account that the RNN only had the Load data set to train on it provided with the best MAPE score for a Univariate model. This also suits a scenario where a particular region has information only on the Load Data set [6, 5, 39].

Recurrent Neural Networks - Multivariate

RNN being the most sought out Time Series prediction model gave out the best MAPE in both Univariate and Multivariate models. Multivariate RNN has eighteen features which are time lagged for 24 time stamps. The multivariate RNN had three layers the input, hidden and the output layer. The input layer had 50 units the hidden had 100 units and a single neuron in the output layer for the load predictions. In Multivariate RNN it returned a score of 3.06 percent for the Test set, which is the best among the Neural Networks and it only improves as the data set gets bigger. This model also suffers from the assumption that all of the data set are readily available for training, Hence falls under the Smart Grid and Smart City scenario. Multivariate RNN takes the longest to run in terms of computational time which should also be taken into consideration [6, 5, 39].

Support Vector Machines

Support Vector Machines model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. With an appropriate non linear mapping to a sufficiently high dimension, data from different categories can always be separated by a hyperplane. They are mainly used for classifica-

tion. Support Vector Machines are regarded in short as Large margin Linear Classifiers. Consider non-separable cases where the hyperplane is modelled by the equation (4.1)

$$y_i[w^T x + w_0] >= 1 - \xi \tag{4.1}$$

where in (4.1) are known as the slack variables. The goal is to make the margin as large as possible. The cost function becomes as follows:

$$J(w, w_0, \xi) = \frac{1}{2} * ||w||^2 + C \sum_{i=1}^{N} (I(\xi_i)) \tag{4.2}$$

where in (4.2) the parameter C is a positive constant that controls the relative influence of the two competing terms, it is also known as the Cost function which has to minimized.

Support Vector Regression

The problem statement in this research is a regression task, that is where Support Vector Regression comes into play. They rely on pre-processing the data into a higher dimension from the original feature space into a hyperplane for either regression or classification. Support Vector Machines are machine learning algorithms, hence it does not require as much of time to run as neural networks. It uses the kernel trick to map the lower dimensional

data to a higher dimension data. Hyperplanes are the lines that are used the separate/classify different data points in the Data set. SVR tries to maximize the boundary margin between the hyperplanes. Support Vectors are the data points or vectors that either lie on the hyperplanes or inside the boundary that constitute the weights for the boundary lines. In the same way as with classification approach there is motivation to seek and optimize the generalization bounds given for regression. They relied on defining the loss function that ignores errors, which are situated within the certain distance of the true value. This type of function is often called – epsilon intensive – loss function. The figure below shows an example of one-dimensional linear regression function with – epsilon intensive – band. The variables measure the cost of the errors on the training points. These are zero for all points that are inside the band [40, 41, 42, 8]. SVM regression performs linear regression in the high-dimension feature space using -insensitive loss and, at the same time, tries to reduce model complexity by minimizing the margin . This can be described by introducing (non-negative) slack variables, to measure the deviation of training samples outside -insensitive zone. Thus SVM regression is formulated as minimization

of the following functional:

$$minimize - J(w, w_0, \xi, \xi^*) = \frac{1}{2} * ||w||^2 + C \sum_{i=1}^{N} (\xi_i + \xi^*) \tag{4.3}$$

such that the following conditions (4.4 - 4.6) are met

$$y_i - w * x_i - b <= \epsilon + \xi, \tag{4.4}$$

$$w * x_i + b - y_i <= \epsilon + \xi^*, \tag{4.5}$$

$$\xi, \xi^* >= 0 \tag{4.6}$$

The SVR was initially designed with only the load values as a input to the

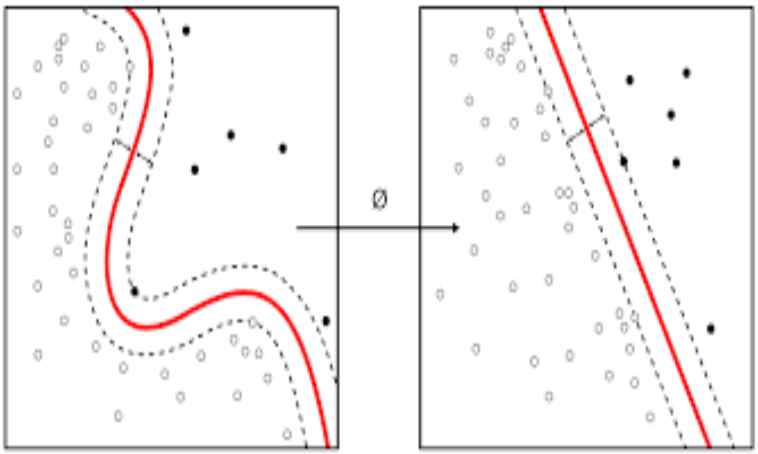

Figure 4.6. Support Vector Regression [43]

training set to see how to perform with lesser features. With limited data set

and feature availability it provided with predictions results which were fairly okay. They provided with a 5.61 % MAPE and almost a 70 % R2 score given the limited availability of the data and features. Below is the actual plot for a univariate SVR model.

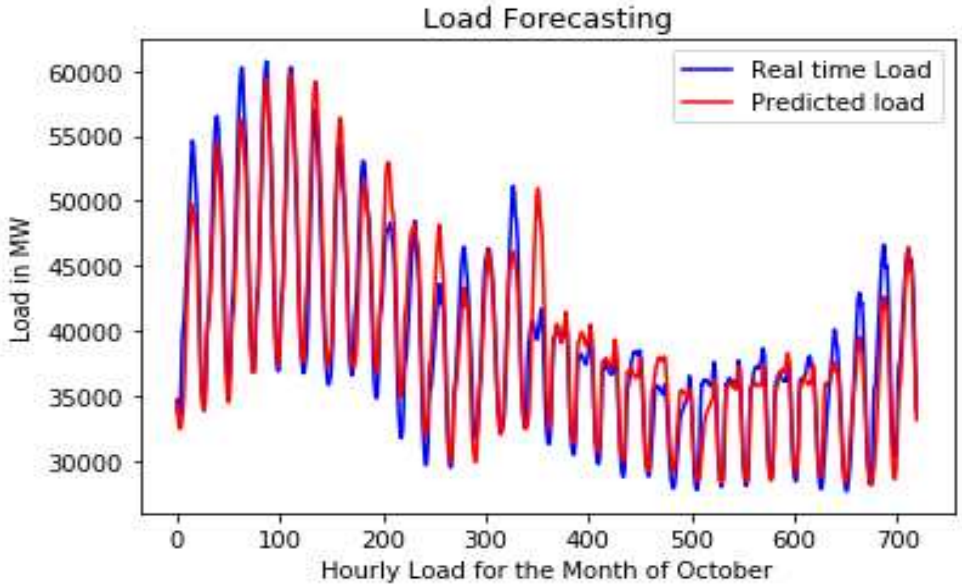

Figure 4.7. SVR - Univariate

Epsilon is the tolerance for the margin. SVR was trained on the training set which comprises of both the Load values and their corresponding Weather data. Eplison was set to 0.01 after tuning on the validation set. The Cost of tolerance was set to 0.1. This SVR model used the radial basis function as the kernel. The training time was comparatively faster than the other models. SVR's usually give better results with smaller data sets as seen from Fig. (5.6)

plotted from the Test set with the actual Load values and the predicted Load values. The kernel used in this model is the radial basis function and after some trails the value of epsilon was optimized to yield the best results. SVR gave about 1.46 percent MAPE on the Test set and had the best result in terms of computational time. SVR generally over fits when there is a larger data set hence, they are well suited for a scenario where the model has access to both the Load values and their corresponding Weather data given that the data set is smaller. SVR's are particularly good in Load forecasting when it deals with a smaller controlled environment also with a much smaller data set. They usually under perform with larger data sets, they run a bunch of complex computations which takes longer in terms of the actual computational time required during the training phase. [40, 41].

Chapter 5

Results

Scenarios

The Machine Learning models in this Thesis were designed for real life scenarios that the utility companies face. Some of the factors that affect these scenarios are Data set availability, geographical location, weather and an ideal situation.

Case - I

Machine learning models rarely have an ideal situation where the Data set is free of noises and just enough for training the model to get the best possible outcome. This includes that the Data set has all the necessary and relevant features like weather and load values for every hour. This type of situation only occurs in hypothetical places where everything is readily available like a Smart Grid or a Smart City. This case serves as a benchmark for

the other cases that are to follow. This scenario can also be regarded as the benchmark case for the other models that follows.

Case - II

There are cases where the geographical location plays a very important role for the data set availability. The Data set is only fully available in big booming cities, not exactly is the same case for remote cities which still need load forecasting. The weather and the load data set may not be fully available for these places where it ends up to substituting values which are not exactly in coherence with the actual pattern, this in-turn leads to irregularities of the predicted values.

Case - III

There case depends on the integrity of the actual data set. Places like the High Plains/Rockies in Colorado have highly unpredictable weather data set which will end up harming the desired outcome of our load values. In these cases the weather data set will be dropped due to their insignificance in their features. Hence at the end there is only the load values as input to the training models to learn from.

Case - IV

In the last case occurs a situation where the required data set is not actually enough to do proper predictions. In this case the models are designed to learn from smaller data sets at the worst case with only one feature. This type of situation occurs quite often in developing cities where the data set available to start with is very scarce.

ANN - Multivariate

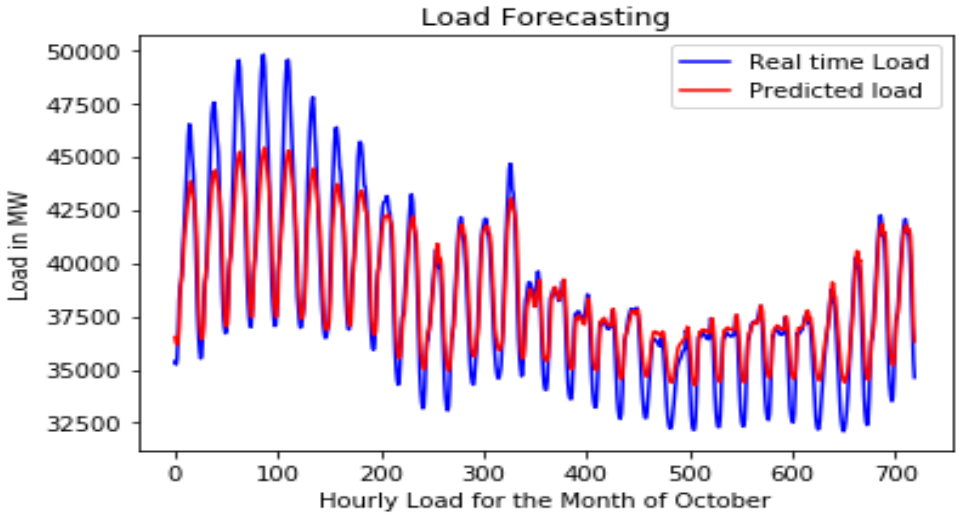

Figure 5.1. ANN - Load in MW vs Hours

Table 5.1. Performance of ANN - Multivariate Model

Model	MAPE %	R2 Accuracy Score %
ANN	6.45	68.81

The ANN Multivariate model was the least performing model in terms of both the MAPE and R2 score, yet it provided acceptable scores around 4 % errors. This will improve if fed with a larger data set hence it falls under the Case I as it is best suited with larger data set like the one's available in Smart Cities and Smart Grids.

CNN - Univariate

Table 5.2. Performance of CNN - Univariate Model

Model	MAPE %	R2 Accuracy Score %
CNN - Univariate	2.98	84.88

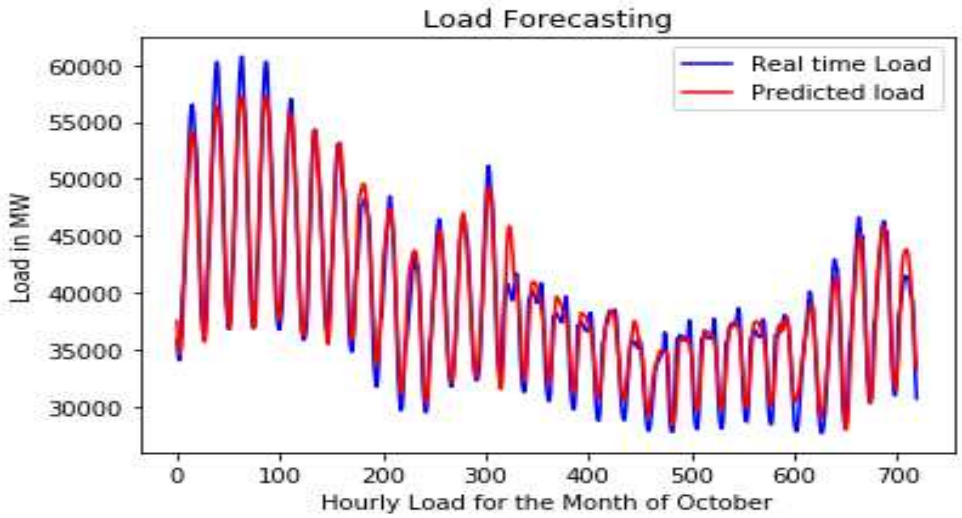

Figure 5.2. CNN - Load in MW vs Hours - Univariate

The one dimensional CNN univariate model performed better than expected as traditionally it is not regarded as the go to time series prediction tool. It comes under both Cases II and III because of its ability to perform even with limited data set. The important thing to take note here is their computational time which is much lower when compared to traditional time series prediction models.

CNN - Multivariate

Table 5.3. Performance of CNN - Multivariate Model

Model	MAPE %	R2 Accuracy Score %
CNN - Multivariate	4.10	76.40

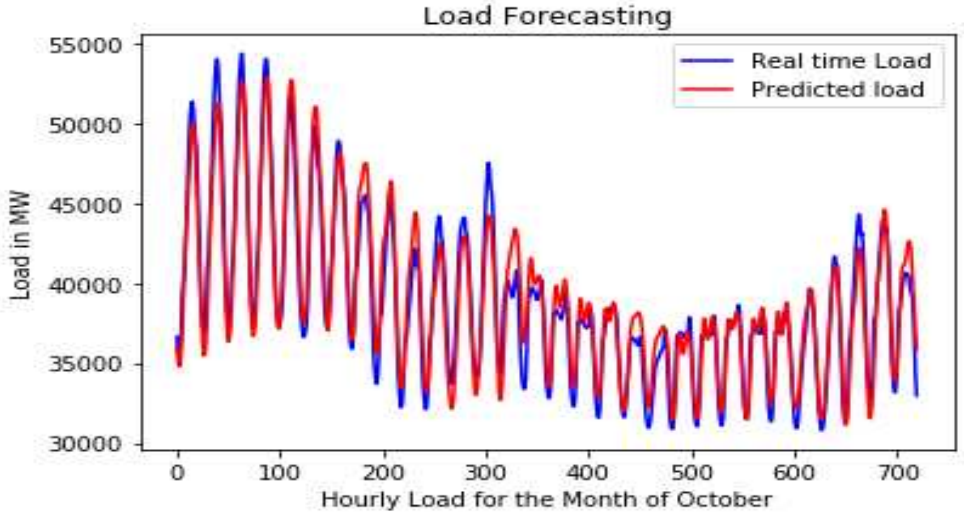

Figure 5.3. CNN - Load in MW vs Hours - Multivariate

The CNN Multivariate model only gave slight improvements over it's univariate counter part. This model can fall under Case I if there is a time constraint where the predictions are to be give out in a short period of time. These model train faster when compared to other multivariate models in this research which is an important resource to be taken account of.

RNN - Univariate

Table 5.4. Performance of RNN - Univariate Model

Model	MAPE %	R2 Accuracy Score %
RNN - Univariate	2.44	88.80

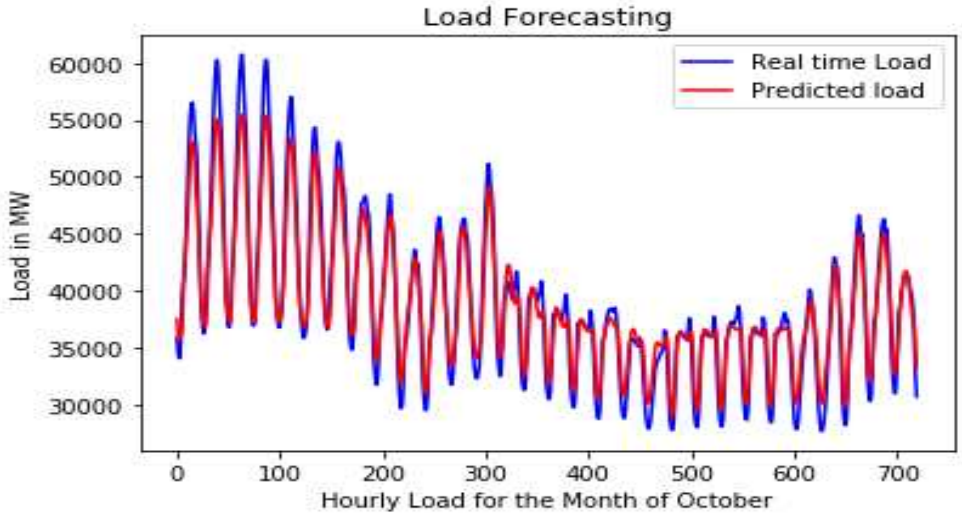

Figure 5.4. RNN - Load in MW vs Hours - Univariate

The RNN univariate model in this research gives out one of the stable results. These are traditional time series prediction models but perform much slower when compared to the other models. These fall under cases II and III if there are no time constraints for training the data.

RNN - Multivariate

Table 5.5. Performance of RNN - Multivariate Model

Model	MAPE %	R2 Accuracy Score %
RNN - Multivariate	3.06	83.72

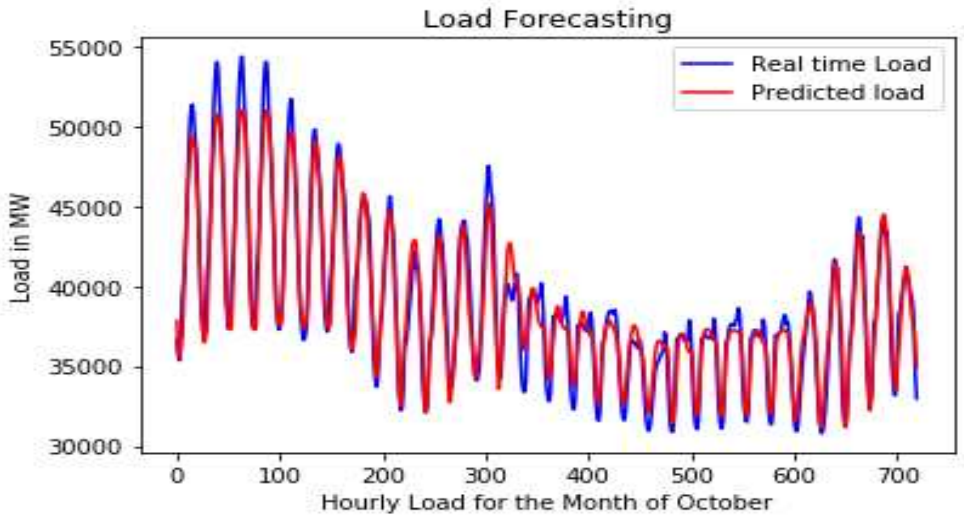

Figure 5.5. RNN - Load in MW vs Hours - Multivariate

The RNN multivariate model is the best among the time series prediction models. It performs better than it's Univariate counter part as expected

and slowest among the other models. This model also fits Cases II and III if there are no time constraints for training the data.

Support Vector Machines - Regression

Table 5.6. Performance of SVR - Multivariate Model

Model	MAPE %	R2 Accuracy Score %
SVR	1.46	92.52

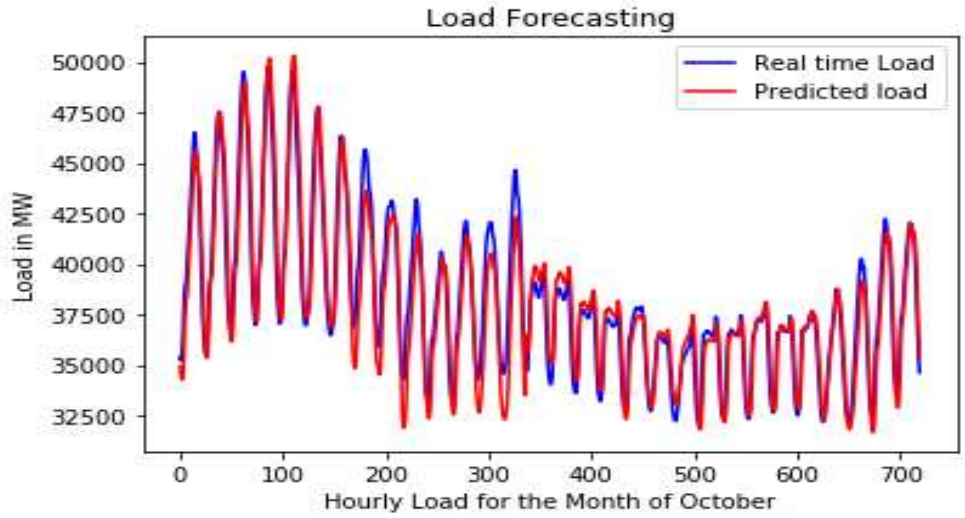

Figure 5.6. Support Vector Regression - Multivariate

The SVR model performs the best when compared to the other models in terms of MAPE, R2 and the computational time as well. This belongs to Case IV where there is a limitation on the data set. SVM's perform well with smaller data sets.

Performance Comparison

The Models in this Thesis are scored on MAPE and R-Squared accuracy score which are regarded as the conventional measure of accuracy for regression analysis. Time being an important resource, the models were also evaluated on the computational time for the training over the training data set. The time observed from table 5.7 are only in seconds which is meagre compared to the actual data because we are only working with a year's worth of data for performance evaluation but the actual training time would take a lot longer than what is observed from Table 5.7 also shows SVR as the most successful model and next comes the RNN model. RNN takes the most time to train the model as they deal with complex and taxing computations in their recurrent layers. The accuracies of the three neural network models will improve with a larger training data set.

Table 5.7. Performance Comparison Table

Model	MAPE %	R2 Accuracy Score %
ANN	6.45	68.81
CNN - Univariate	2.98	84.88
CNN - Multivariate	4.10	76.40
RNN - Univariate	2.44	88.80
RNN - Multivariate	3.06	83.73
SVR	1.46	92.52

Chapter 6

Conclusion

It is hard to differentiate between these models to pick a clear winner as each have their own benefits and shortcomings. The Recurrent Neural Networks Multivariate model and Support Vector Machines stands out with their results from the average value of MAPE, R-Squared score and computational time. SVR works better with a smaller data set given that they have all the required features for forecasting hence, they are used in a scenario where the environment is controlled, like a building where all the features for Load forecasting are readily available but only in smaller data sets. RNN's accuracy improves with a bigger data set that is provided to the model for training but like SVR they require all the features in the data set for it to perform better. RNN's Multivariate model are well suited for a scenario where the data set has all the features and they are also larger, such an environment exists in Smart Cities or Smart Grids where they have all the data they need to train their models and hence provide good results on the load forecasts [31]. The

Recurrent Neural Network and the Convolutional Neural Network model also performs even better in terms of their current MAPE and R-Squared score if they are fed with a larger data set for training. The RNN's and CNN's Univariate models were trained only with the load values. They did not use the weather data set for training even then they provided with a MAPE values which were almost similar. The MAPE for these models even though on the higher side than the other models, they are better results as they were trained only on the actual Load values as the primary feature. RNN's and CNN's Univariate fit a scenario where there may not be a data set with all the features, like a remote city with only the actual load values as the data set available for training. Though the results of CNN are quite like RNN but in practice RNN's are much better Time-Series Prediction models [6]. RNN's and CNN's Univariate model also works well for this scenario, where the forecasting must be done only with the load values which is what happens in most cases today where the weather data set is available for only a certain period of time and missing for the other, this is mainly due to the fact that the weather is highly unpredictable in certain areas and cannot really rely on the weather for forecasting in that area. Hence, it really depends on the data set availability to determine which model to be used for load forecasting for the given scenario.

Future Work

This Research aimed to explore the difficulties faced by Utility companies in the real world on different scenario's based on the availability of data and provided quick results, but they all focused on Supervised Learning Methods. It is a possibility to work on Unsupervised Learning Methods like K-Means Clustering, Self-Organizing Maps, Boltzmann Machines, Auto encoders and so on to provide a much better insight on clustering of the data and then feed into the actual networks, which gives rise to hybrid models for Load forecasting. These Hybrid models are time consuming compared to other Supervised Learning models but will provide new insights from the data and then used for training.

Appendix A

Definitions

- Bias: The bias is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

- BigData: extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.

- Biomass: organic matter used as a fuel, especially in a power station for the generation of electricity.

- Classification: In machine learning and statistics, classification is the problem of identifying to which of a set of categories (subpopulations) a new observation belongs, on the basis of a training

set of data containing observations (or instances) whose category membership is known.

- Convolution: It is defined as the integral of the product of the two functions after one is reversed and shifted.
- Deep Learning: is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled.
- Encoder: An encoder is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another, for the purpose of standardization, speed or compression.
- Entropy: in machine learning, is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random
- Geothermal: relating to or produced by the internal heat of the earth.

- Gradient Descent: is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model.
- Hydroelectric: relating to or denoting the generation of electricity using flowing water (typically from a reservoir held behind a dam or other barrier) to drive a turbine that powers a generator.
- Hyperplane: In geometry, a hyperplane is a subspace whose dimension is one less than that of its ambient space.
- Independent System Operator: is an organization formed at the recommendation of the FERC.
- Labelled Data: data that typically takes a set of unlabeled data and augments each piece of that unlabeled data with some sort of meaningful "tag," "label," or "class" that is somehow informative or desirable to know.
- Neural Networks: Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.

- Regression: a measure of the relation between the mean value of one variable (e.g. output) and corresponding values of other variables (e.g. time and cost)
- Solar: relating to or denoting energy derived from the sun's rays.
- Sigmoid: It is used in neural networks to give logistic neurons real-valued output that is a smooth and bounded function of their total input
- Smart City: A smart city is a designation given to a city that incorporates information and communication technologies (ICT) to enhance the quality and performance of urban services such as energy, transportation and utilities in order to reduce resource consumption, wastage and overall costs.
- Smart Grid: is an electrical grid which includes a variety of operation and energy measures including smart meters, smart appliances, renewable energy resources, and energy efficient resources
- Time-Series Analysis: A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time.
- Unlabelled Data: data that consists of samples of natural or human-created artifacts that you can obtain relatively easily from the world.

- Utility Company: An electric utility is a company in the electric power industry (often a public utility) that engages in electricity generation and distribution of electricity for sale generally in a regulated market.
- Validation: In machine learning, model validation is referred to as the process where a trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived.
- Variance: is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting). Variance is the difference between many model's predictions.

Appendix B

Acronyms and Abbreviations

- Adam: Adaptive Moment Estimation
- ANN: Artificial Neural Network
- AI: Artificial Intelligence
- CNN: Convolutional Neural Network
- DBSCAN: Density-based spatial clustering of applications with noise
- DSP: Discrete Time Signal Processing
- ERCOT: Electric Reliability council of Texas
- FERC: Federal Energy Regulatory Commission
- ICLR: The International Conference on Learning Representations
- ICT: Information and Communication Techonologies
- ISO: Independent System Operator
- LSTM: Long Short Term Memory
- MAPE: Mean Absolute Percentage Error
- ML: Machine Learning

- MNIST: Mixed National Institute of Standards and Technology
- MSE: Mean Squared Error
- NOAA: National Oceanic and Atmospheric Administration
- R2: R Squared score
- ReLU: Rectified Linear Unit
- RMSE: Root Mean Squared Error
- RNN: Recurrent Neural Network
- SVM: Support Vector Machines
- SVR: Support Vector Regression

Bibliography

[1] US Energy Information Administration.
US Electricity Generation by major energy source, 1950 - 2018. 2019.
URL: `https://www.eia.gov/energyexplained/index.php?page=electricity_in_the_united_statesm` (visited on 01/04/2019).

[2] N Phuangpornpitak and W Prommee.
"A Study of Load Demand Forecasting Models in Electric Power System Operation and Planning".
In: *GMSARN International Journal* 10 (2016), pp. 19–24.

[3] Sumit Saroha. "Forecasting issues in present day power systems".
In: (2013).

[4] Kishan Bhushan Sahay and MM Tripathi.
"Day ahead hourly load forecast of PJM electricity market and ISO New England market by using artificial neural network".
In: *ISGT 2014*. IEEE. 2014, pp. 1–5.

[5] Abdulaziz Almalaq and George Edwards.
"A review of deep learning methods applied on load forecasting".
In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, pp. 511–516.

[6] Xueheng Qiu et al.
"Ensemble deep learning for regression and time series forecasting".
In: *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*. IEEE. 2014, pp. 1–6.

[7] Daniel L Marino, Kasun Amarasinghe, and Milos Manic.
"Building energy load forecasting using deep neural networks".
In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 7046–7051.

[8] EDH Stork et al. "Pattern classification".
In: *New York [ua]: Academic Internet Publishers* (2006).

[9] Jacek M Zurada. *Introduction to artificial neural systems*. Vol. 8. West publishing company St. Paul, 1992.

[10] Yann LeCun and M Ranzato. "Deep learning tutorial". In: *Tutorials in International Conference on Machine Learning (ICML'13)*. Citeseer. 2013, p. 35.

[11] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[12] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[13] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[15] Towards Data Science. *Overfitting/Underfitting*. 2018. URL: `https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229` (visited on 01/10/2019).

[16] Anders Krogh and Jesper Vedelsby. "Neural network ensembles, cross validation, and active learning". In: *Advances in neural information processing systems*. 1995, pp. 231–238.

[17] Ron Kohavi et al. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[18] Raheel Shaikh. *Cross Validation Explained: Evaluating estimator performance*. 2018. URL: `https://towardsdatascience.com/`

cross-validation-explained-evaluating-esimator-performance
(visited on 12/10/2018).

[19] Feiping Nie et al. "Efficient and robust feature selection via joint 2, 1-norms minimization".
In: *Advances in neural information processing systems*. 2010, pp. 1813–1821.

[20] Nitish Srivastava et al.
"Dropout: a simple way to prevent neural networks from overfitting".
In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[21] Google Brain Team.
Neural Network Library Python Package - Tensorflow. 2015.
URL: https://www.tensorflow.org/ (visited on 10/12/2018).

[22] Francois Chollet. *Neural Network Library Python Package - Keras*. 2015. URL: https://keras.io/ (visited on 10/12/2018).

[23] Travis Oliphant. *Scientific Library - Python Package*. 1995.
URL: https://numpy.org/html (visited on 10/12/2018).

[24] David Cournapeau. *Scientific Library - Python Package*. 2018.
URL: https://scikit-learn.org/html (visited on 10/12/2018).

[25] ERCOT (Electric Reliability Council of Texas). *2018 Load Data set*. 2018. URL: http://www.ercot.gov (visited on 01/25/2019).

[26] NOAA (National Oceanic and Atmospheric Administration).
2018 Weather Data set. 2018.
URL: http://www.noaa.gov (visited on 02/11/2019).

[27] Rich Caruana, Steve Lawrence, and C Lee Giles.
"Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping".
In: *Advances in neural information processing systems*. 2001, pp. 402–408.

[28] David Cournapeau. *Standard Scaler*. 2018. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (visited on 10/12/2018).

[29] Scikit learn. *Data Pre-Processing - MinMaxScaler*. 2015. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (visited on 12/12/2018).

[30] David Cournapeau. *Data Pre-Processing*. 2018. URL: https://scikit-learn.org/stable/modules/preprocessing.html (visited on 10/12/2018).

[31] Aymen Chaouachi et al. "Multiobjective intelligent energy management for a microgrid". In: *IEEE transactions on Industrial Electronics* 60.4 (2012), pp. 1688–1699.

[32] Dong C Park et al. "Electric load forecasting using an artificial neural network". In: *IEEE transactions on Power Systems* 6.2 (1991), pp. 442–449.

[33] Quoc V Le et al. "A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks". In: *Google Brain* (2015), pp. 1–20.

[34] Kasun Amarasinghe, Daniel L Marino, and Milos Manic. "Deep neural networks for energy load forecasting". In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. IEEE. 2017, pp. 1483–1488.

[35] Xishuang Dong, Lijun Qian, and Lei Huang. "Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach". In: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE. 2017, pp. 119–125.

[36] Jürgen Schmidhuber and Sepp Hochreiter. "Long short-term memory". In: *Neural Comput* 9.8 (1997), pp. 1735–1780.

[37] colah's blog. *Understanding the LSTM Networks*. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs (visited on 10/10/2018).

[38] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks". In: *Journal of machine learning research* 3.Aug (2002), pp. 115–143.

[39] Weicong Kong et al. "Short-term residential load forecasting based on LSTM recurrent neural network". In: *IEEE Transactions on Smart Grid* (2017).

[40] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[41] Bo-Juen Chen, Ming-Wei Chang, et al. "Load forecasting using support vector machines: A study on EUNITE competition 2001". In: *IEEE transactions on power systems* 19.4 (2004), pp. 1821–1830.

[42] Christopher JC Burges. "A tutorial on support vector machines for pattern recognition". In: *Data mining and knowledge discovery* 2.2 (1998), pp. 121–167.

[43] Wikipedia. *Support Vector Machines*. 2000. URL: https://en.wikipedia.org/wiki/Support-vector_machine (visited on 10/13/2018).