**Comparison of Machine Learning Algorithms for Short Term Load Forecasting Based on**

**Internet of Things (IoT)**



Thesis submitted in partial fulfilment of the requirement for the degree of Bachelor of Computer Science and Engineering

Under Supervision of

**Amitabha Chakrabarty, Ph.D.**

By

Mahdi Uz Zaman (14101263)

Anisul Islam Fahad (130000)

Nahid Sultana (13301133)

**School of Engineering and Computer Science**

22th March 2018

BRAC University, Dhaka, Bangladesh

# Declaration

We hereby declare that this thesis has been done based on the results we obtained from our work. Due acknowledgement has been made on text. This thesis neither in parts nor as a whole have been submitted previously by anyone of any institute or university for the award of any degree.

**Signature of Supervisor** **Signature of the Authors**

|  |  |
| --- | --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Amitabha Chakrabarty, Ph.D.**  Assistant Professor  Department of Computer Science and Engineering  BRAC University | Mahdi Uz Zaman  (14101263) |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Anisul Islam Fahad  (13000000) |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Nahid Sultana  (130000000) |

# ABSTRACT

In the era of internet every device is getting connected to the internet. If the device upload information about their power usages

# Acknowledgement

First and foremost, we would like to thank Almighty Allah for enabling us to initiate the research, to put our best efforts and successfully conclude it.

Secondly, we would like to thank our supervisor Dr. Jia Uddin. Without his support, guidance and contribution we could not have done it. He motivated us to do this research thoroughly and always was there to help. This paper have been completed due to his great supervision and inspiration. We are truly grateful to him.

We revere the patronage and moral support extended with love, by our parents as well as our friends. They helped us with their direct or indirect suggestions which aided in achieving our goal. We would also like to acknowledge the assistance we received from numerous resources over the Internet especially from fellow researchers’ work.

Last but not the least, we thank BRAC University for providing us the opportunity of conducting this research and for giving us the chance to complete our Bachelor degree.

Contents

[Declaration ii](#_Toc508829920)

[ABSTRACT iii](#_Toc508829921)

[Acknowledgement iv](#_Toc508829922)

[List of Figure vii](#_Toc508829923)

[List of Tables viii](#_Toc508829924)

[List of Abbreviations ix](#_Toc508829925)

[CHAPTER 01 1](#_Toc508829926)

[1.1 Introduction 1](#_Toc508829927)

[1.2 Motivation 1](#_Toc508829928)

[1.3 Contribution Summary 2](#_Toc508829929)

[1.4 Thesis Outline 2](#_Toc508829930)

[CHAPTER 02 3](#_Toc508829931)

[2.1 Literature Review 3](#_Toc508829932)

[2.2 Load Forecasting with Machine Learning 6](#_Toc508829933)

[2.3 Time Series Analysis with Machine Learning 6](#_Toc508829934)

[2.4 Long Short-term Memory Network (LSTM) 7](#_Toc508829935)

[2.4.1 Recurrent Neural Network 7](#_Toc508829936)

[2.4.2 RNN Cell 8](#_Toc508829937)

[2.4.3 Vanishing Gradient of RNN 8](#_Toc508829938)

[2.4.4 LSTM Network 9](#_Toc508829939)

[2.4.5 LSTM Cell 9](#_Toc508829940)

[CHAPTER 03 13](#_Toc508829941)

[Proposed Model 13](#_Toc508829942)

[3.1 Process 13](#_Toc508829943)

[3.2 Data Collection 14](#_Toc508829944)

[3.2 Preprocessing and Filtering Data 16](#_Toc508829945)

[3.7 Training Machine Learning Model 21](#_Toc508829946)

[3.7.7 Programming Language and Tools 21](#_Toc508829947)

[3.7.7 Long Short-term Memory Network 22](#_Toc508829948)

[3.8 Nearest neighbour 23](#_Toc508829949)

[3.9 Load Forecasting 24](#_Toc508829950)

[CHAPTER 04 25](#_Toc508829951)

[Experimental Setup & Result Analysis 25](#_Toc508829952)

[CHAPTER 05 37](#_Toc508829953)

[Conclusion & Future Work 37](#_Toc508829954)

[Reference 38](#_Toc508829955)

# List of Figure

Figure 1: Naïve bayes classifier workflow……………………………………………………7

Figure 2: Naïve Bayes Algorithm.…………………………………………………………….8

Figure 3: Proposed Model’s workflow……………………………………………………….10

Figure 4: Twitter advanced search…………………………………………………………...12

Figure 5: Data before processing in JSON format…………………………………………...13

Figure 6: Data after processing …………….……………………………………………......14

Figure 7 Sentence polarity …………….…………………………………………………......16

Figure 8: Initial polarity graph ………………………………………………………………17

Figure 9: Processor sentiment graph………………………………………………………....19

Figure 10: Camera sentiment graph ….……………………………………………………...20

Figure 11: Battery sentiment graph ……….…………………………………………………21

Figure 12: Price sentiment graph ……….……………………………………………………22

Figure 13: Display sentiment graph……….…………………………………………………23

Figure 14: UI sentiment graph ………….……………………………………………………24

Figure 15: Time vs Polarity graph of Samsung Galaxy S8………….……………………….25

Figure 16: Time vs Polarity graph of Apple IPhone X ……………………………………...26

Figure 17: Time vs Polarity graph of Google Pixel 2 ……………………………………….27

# List of Tables

|  |  |  |
| --- | --- | --- |
| Table 1: Processor attribute based sentiment count table. | …………………… | 19 |
| Table 4: Camera attribute based sentiment count table. | …………………… | 20 |
| Table 4: Battery attribute based sentiment count table. | …………………… | 21 |
| Table 4: Price attribute based sentiment count table. | …………………… | 22 |
| Table 4: Display attribute based sentiment count table. | …………………… | 23 |
| Table 4: UI attribute based sentiment count table. | …………………… | 24 |

# List of Abbreviations

API – Application programming interface

CSV – Comma separated value

IDE – Integrated development environment\

JSON – Java script object notation

MatPlotLib – Mathematical plotting library

NLP – Natural Language Processing

NLTK - Natural language tool kit

# CHAPTER 01

## 1.1 Introduction

Electrical energy generation and distribution is a complex and costly process. Efficient grid management plays a big role to reduce the cost of energy production. Grid management comprises of planning for load demand, maintenance of generation units, supply lines and efficient load distribution across the supply line. Therefore an accurate load forecast will increase the efficiency of planning process of a power generation company. Power generation companies do their plan based on data collected manually. Therefore real time prediction is not possible. If data can be collected in real time, forecasting in real time will be possible. Strong and reliable Internet infrastructure are already present. Every device we use in our daily life are gradually getting connected to the internet to facilitate smart home technologies like Google Home, Amazon Alexa. A device connected to internet usually treated as IoT. In general a device with sensors, microprocessor or microcontroller which can connect to the internet, send and receive information through internet is called Internet of Thing (IoT).  If the devices are configured to send energy uses data to the internet, these data can be used to give real time forecasting. In this paper we have shown that real time load forecasting is possible with the help of IoT and state of the art artificial intelligence algorithm LSTM Network and machine learning algorithms Nearest Neighbors Regression, Support Vector Regression and Decision Forest Regression.

## 1.2 Motivation

First of all, increased demand of electricity is creating pressure on production companies as well as natural resources. We know natural resources which are used to produce to create electricity are limited in nature. Secondly, the byproduct of electricity generation is pollution. Again, cost for producers and consumers are increasing day by day. Therefore to ensure sustainable development research communities have shown great interest on how to reduce electricity demand by efficient use of electricity. One of the important of methods that is used to facilitate efficient use of electricity is load forecasting. With the help of load forecasting producers can tune their production plan and consumer can optimized their electricity consumption. Existing forecasting system relies on data collected from production and distribution unit. We have shown that with the help of IoT load forecasting can be done in more easy, convenient and reliable way.

## 1.3 Contribution Summary

The main contribution of this project is to show that IoT can be used as a reliable backbone of a load forecasting system. To support our claim we have tested our system with real world datasets. Based on this dataset we have done empirical comparison and performance evaluation of four machine learning algorithm. This system will help home user to reduce their power consumption by early warning of future power use. This will also help the power generation company to meet their demand efficiently by planning ahead of time.

## 1.4 Thesis Outline

Over all in this paper, Chapter 2 provides the Background study in details including the algorithms and techniques used in the system, Chapter 3 discusses the proposed model including the algorithms and techniques, Chapter 4 presents the results and analysis and lastly Chapter 5 gives the conclusion and future work. Need to change

# CHAPTER 02

## 2.1 Literature Review

Load forecasting plays an important role to the efficient use of electricity as well as efficient production and distribution. Power load forecasting is categorized in three categories. They are short term load forecasting (STLF), medium term load forecasting (MTLF) and long term load forecasting (LTLF). These categorization is depend on the range of future time taken in to consideration to be predicted. Prediction process which give prediction day or week ahead is called short term load forecasting (STLF). Medium term load forecasting (MTLF) system are built for month ahead prediction and when years ahead predictions are needed long term load forecasting (LTLF) system are incorporated. In this paper we have built and tested a system which can predict a day ahead forecasting. Researchers have proposed different types of methods for load forecasting. Some of them are discussed below.

Kong, W. contributed in deep learning based method with appliance behavior learning for meter level load forecasting which demonstrated an advantageous performance through extensive comparison with other predictors. According to this paper, if we can learn the lifestyle pattern of certain resident can help us achieve better metering forecasting. His work showed that the using appliance measurements in training data can improve the forecasting accuracy. Contextual variables like temperature, humidity, day of the week and special events are taken into consideration in this method for better forecasting performance. In this paper individual load forecasting is done using LSTM. Long short-term memory (LSTM) is one of the RNN structure, the specialty of this RNN is sequence learning. It maintains a memory cell in its structure to remember important state in past to reset the memory cell it has a forget gate. As mentioned above the learning lifestyle pattern can be done if appliance level consumption are directly measured which assists in interpreting in the forecasting. So instead of serving aggregated data to the LSTM the inputs are all available major appliance energy sequence to train the predictor. Here the dataset is taken from a Canadian household and its 19 appliance for a year. Then the current reading is converted into Ampere hour for every 30 minutes to imitate the smart meter data. The appliance chosen for appliance learning are clothes dyer, clothes washer, dishwasher, heat pump, television and wall oven which are manually operated. For resident behavior learning this approach used the measurement of both the whole house-hold consumption and the selected appliance from the past several time interval until the current time as inputs. The consumption forecast of subsequent time interval is the output. To compare the performance of this proposed method feed forward neural network (FFNN) and K-nearest neighbor (KNN) is used. The lowest benchmark is set by empirical mean which is the forecasting value of the statistical mean given by the time of the day and day type. Here “look-back” input scheme, a system level load forecasting which uses measurement of same time interval of the past few days also compared referred with suffix “D” suffix “WA” is used to label test cases that use extra appliance measurement in training data and suffix “W” for whole house measurement only. This paper concluded showing that LSTM-WA outperformed all other methods. And LSTM-WA with two look back interval achieved the best overall MAPE scores and the second best LSTM-W predictor with a 4.24% MAPE margin. KNN and FFNN the version of extra appliance data gave better result compared to whole house consumption. In conclusion, the LSTM based forecasting framework gives better accuracy when consumption sequence of major appliance is available.

Ghulam and Angelos, worked on the applicability and compared the performance of Feed-forward deep neural network (FF-DNN) and recurrent deep neural network (R-DNN) models on the basis of accuracy and computational performance in the context of time wise short term forecast of electricity. Analyzing the data on the time and frequency domain independently and subsequently frequency domain components are transformed back to the time domain. The parameter which are taken into consideration are weather, time, holidays, working days, and lagged load and data distribution effects. This paper collected the dataset from ISO New England for duration 2007-2012. The load consumption is recorded at the end of every hour of a day and the whole dataset consisted of 52600 records that represented data of 6 states of New England, USA. The changing behavior of the dataset is captured during analysis in time domain and effects that were captured are temperature effect, working and non-working day’s effect, time effect, lagged load effect and data distribution effect. After time domain extraction the data is further analyzed in frequency domain. The random signals of time domain are converted to different frequencies that are stable and easily predictable which improves accuracy. Fast Fourier is performed to determine the dominant frequencies and the one with higher magnitude represent the dominant frequencies. Here evaluating the proposed models 43824 records from 2007-2011 are used in training dataset and 24 and 168 records for days and weeks for test dataset. The RMSE, MAE and MAPE are calculated for four seasons of 2012 and the 5 features extracted from the original features are taken into consideration for better accuracy. The result of considering only time domain in both FF-DNN and R-DNN varied due to temperature variation in different seasons. The MAPE is 1.30% for R-DNN and 1.42% for FF-DNN in a year. On the other hand the error are much lower and the accuracy is improved in frequency domain analysis where MAPE is 0.067% and 0.057% for FF-DNN and R-DNN respectively. So based on the analysis it was shown that weather, time, holidays, lagged load and data distribution have most dominant factors and the TF features can be utilized for load forecasting.

Saber and Khandelwal, study showed a combination of IoT devices and Load forecasting (LF). IoT based LF collects the data from the internet and performs statistical and optimization methods for efficient forecasting. For training to map the dynamic of load is done by Neural Network (NN) and then optimization on the NN weights to improve overall forecasting error is done by Particle swarm optimization (PSO). Along with NN and PSO weather data of the geographical location including temperature, humidity, wind speed, wind direction, heat, sunlight, solar radiation and rainfall are taken into consideration for better accuracy. NN is chosen for an initial forecasting of load pattern and multi-objective PSO is used to improve the accuracy of results. Dynamic real-time load data were collected from smart or semi smart meter in interval of 15 minutes to 1 hour. Weather data is taken from the internet using weather API. The training data set is divided into two subsets, training set and validation set. Validation set is used to avoid over fitting problem. Data from 01 April to February are used for training and data of March is used as test set. Daily load of previous day, temperature and humidity of forecast day are most prominent input variables. After 10 runs MAPE of NN-PSO is better than NN. Average MAPE is 2.0433% for NN and 2.0129% for NN-PSO. Which shows that multi-objectives PSO has improved the results because it consists of two methods and covers more search space. This paper showed that optimization of NN weights can give more accurate load forecasting. But again the issue of cyber security in IoT should be considered.

Papia Ray, Santanu Sen and A.K. Barisal presented two hybrid methodologies based on discrete wavelet transform (DWT) in combination with ANN or SVR for Short Term Load Forecasting (STLF) using feature selection. This method was done with the data taken from a particular area of New Delhi for a particular month. The data is taken from December 1 to February 28. Temperature, humidity, dew point and load consumed for a particular day at a particular hour are also taken into consideration. Here data from December 5 to January 31 are taken as training data, from February 1 to 28 are used as validation set and the 4days data are taken as test set data. Here the feature selection is done through Forward Feature Selection (FFS). The analysis was done in two ways one using FFS and other without using FFS and it showed that the one done with FSS gave a better result. These analysis was done on monthly based, weekly based and daily based and among two hybrid methods DWT-SVM showed an error of 0.1% and DWT-ANN showed error of 0.6% which concluded that DWT-SVM showed better result than DWT-ANN.

Taking in consideration of the above mentioned work we have implemented an IoT based load forecasting system. The core algorithm of the forecasting system is a machine learning algorithm. To select best performing algorithm we have tested performance of several machine learning algorithm with a new dataset called “UK-DALE dataset”.

## 2.2 Load Forecasting with Machine Learning

Machine learning is the ability of a machine to do certain task without being explicitly programed. One of the branches of machine learning is supervised learning. Supervised learning is process of turning experience into expertise. In another way when machine learn from previous incidents and take decision on unknown incidents is called supervised learning. Load forecasting is a complex process due to its nonlinear and always changing nature. If we can teach a machine the user pattern of a certain device which consume electric power a good machine learning algorithm can tell the probability of future occurrence. In conventional statistical methods hypothesizes are formalized before seeing the data. Which tend to give faulty results. On the other hand machine learning algorithm produce hypothesis after seeing the data. As hypothesizes are constructed based on datasets machine learning hypothesis are adaptable with different scenario and circumstances. Therefore we have choose machine learning approach for load forecasting.

## 2.3 Time Series Analysis with Machine Learning

In time series data points are collected sequentially with respect to time. Three are mainly two type of time series. One is discrete time series, where data point are collected with a fixed interval of time. One the other hand in continuous time series data points are taken continuously without any interval. In the definition of discrete time series it is said that difference between two data points must be maintained throughout the whole datasets. The datasets we have used in this project is discrete time series. In figure [] a screen of a time series datasets has ben shown.

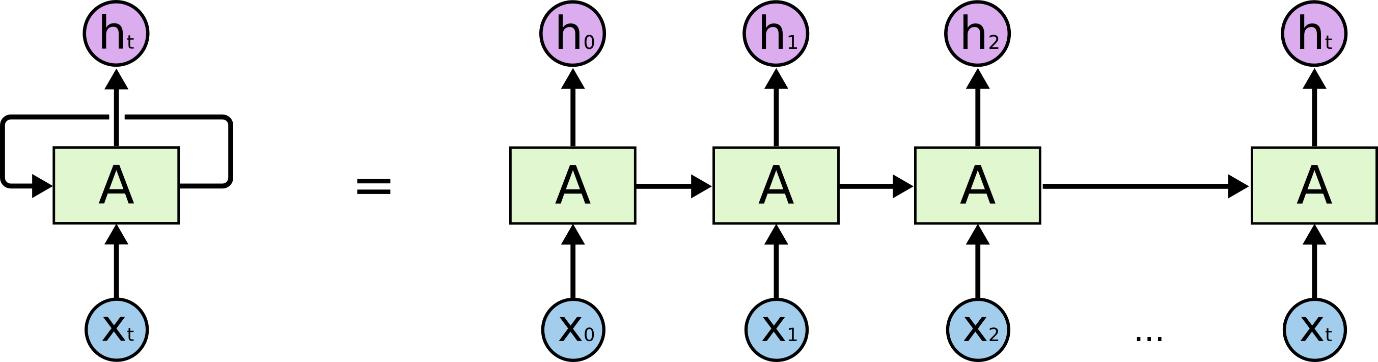
Time series helps to analyze the user pattern of electrical devices with respect to time. Time series analysis is a process where time series datasets are analyzed to extract features. In this paper we have used machine learning approach to analyze time series datasets. We cannot use a time series dataset directly for a machine learning algorithm. Different algorithm requires special kind of data preparation. In general events on (t-1) is compared with t.

## 2.4 Long Short-term Memory Network (LSTM)

LSTM Network is built based on the basic concept of Recurrent Neural Network (RNN). Before going to the details description of LSTM a short summary on RNN is given below.

### 2.4.1 Recurrent Neural Network

A Recurrent Neural Network is a combination of numbers of identical feed forward neural network. This combination is made by copying the same network sequentially with respect to time steps. Each copy of the network can pass information to the newer copy of the network. In figure[] how a RNN increase by coping itself with respect to time has shown.



### 2.4.2 RNN Cell

Unlike neuron of a neural network, RNN has cell. A RNN cell itself contain a neural network. In figure[] a RNN cell has been shown. In figure[] box denoted by “A” is a RNN cell. In each time step RNN cell has a cell state.



|  |  |
| --- | --- |
|  | (1) |

Output of RNN cell is computed by the equation (1). Here is the output of the current state. Previous state and current input is multiplied by a weight and added to bias *b.* Then the result is passed through a sigmoid activation function. Vector dimension of is same.

### 2.4.3 Vanishing Gradient of RNN

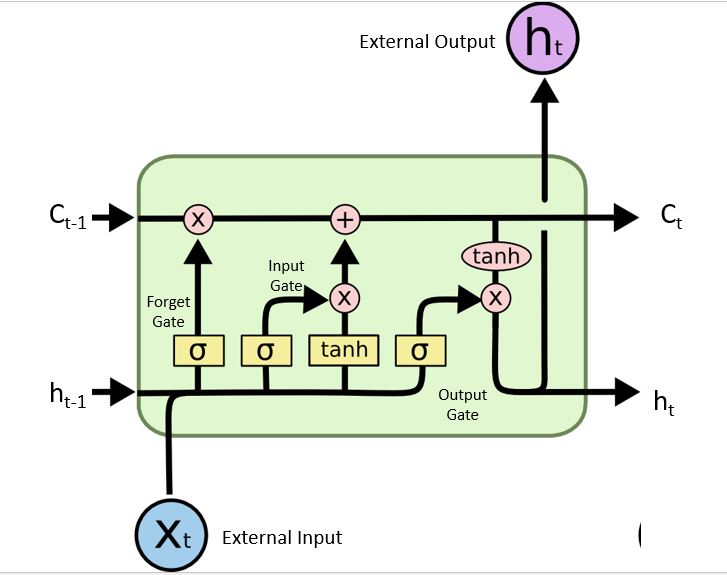
In figure[] a sequence of RNN Network has been shown. If the sequence is too long then the network faces vanishing gradient problem due to backpropagation through time. Due to this problem RNN Network cannot remember an occurrence which happened long time ago. This is the major drawback of RNN Network. This is also called long term dependency problem.

### 2.4.4 LSTM Network

Sepp Hochreiter and Jürgen Schmidhuber published a paper in 1997 called “Long Short-term Memory” to address the drawbacks of basic RNN Network. They proposed a new architecture of RNN Cell. They named the new architecture as LSTM.

### 2.4.5 LSTM Cell

Specialty of a LSTM cell is its memory and three gates. LSTM uses the memory to remember from very deep down the sequence. It uses the gates to control the flow of information from memory and to the memory. In figure[] a schematic of LSTM cell is given.



Input of LSTM Cell:

* Previous cell State (Ct*-1*): Memory of previous state is forwarded to the current state (Ct). Which will then added to the current state by increment. Process of increment is given in the description of forget gate.
* Previous cell output (*ht-1*): Cell output of previous time step is taken as input in current time step.
* External input (*xt*): Neural network output of current time step. This output is optional. Depend of the need it sometimes give output an sometimes not.

Output of LSTM Cell:

* Current cell state (Ct): Current cell state is forwarded to future time step through cell state output.
* Cell output (*ht*): After squashing cell state through tanh activation function, cell output is forwarded to future time step.
* External output (*ht*): Optional Neural Net output.

Equations for LSTM is given below

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |
|  | (4) |
|  | (5) |
|  | (6) |
|  | (7) |

* : Weight for forget gate layer
* : Bias for forget gate layer
* : Output of input gate layer
* : Weight for input gate layer
* : Bias for input gate layer
* : Candidate value
* : Weight of tanh layer
* : Bias of tanh layer
* : Weight of output gate layer
* : Bias of output gate layer

Gates of LSTM Cell:

* Forget Gate (*ft*): Overwhelming unnecessary information causes information morphing to the cell state, as known as memory. Forget gate shorts out most relevant information by sigmoid layer. In figure[] a sigmoid function is given. It squashes the output between (0 to 1). 1 means completely remember and 0 means completely forget. Equation 2 shows that with respect to previous cell output and current input it calculate which part of the previous information need to remember in the current state.
* Input Gate (): It is also known as write to the memory. When LSTM cell gets new external input it needs to decide which part of the memory it will overwrite with new value. First with the help of sigmoid layer it decides which values it needs to update. Weighted external input, previous cell output and bias is passed to sigmoid layer according to equation (3). Then a tanh layer select a vector of candidate values by equation (4) for the selected values by equation (3). In figure[] a tanh function is given. A tanh function squashes input between (-1 to 1). After multiplying output of equation (3) and (4) element wise we get a new cell state or memory state (*Ct*) [*Eqn: 5*]
* Output Gate (*ot*): Unlike RNN cell output, LSTM cell does not output the exact copy of cell state. Instead it outputs a filtered version of the cell state. A sigmoid gate[eqn: 6] decide which part of the memory will be forwarded to the next state. Present cell is squashed by a tanh layer to scale it between (-1 to 1) multiplied elementwise with output of equation (6) to produce cell output [eqn:7].

The ability of selectively read, write and remember the events happened in numbers of previous time steps makes LSTM a robust algorithm for sequence learning.



**2.4. K-Nearest Neighbors Regression**

K-NN algorithm saves all convenient cases to predict the targeted value based on the similar values. K-NN is not a new technique, it has been used in 1970’s for statistical estimation and pattern recognition as non-parametric technique.

Algorithm

KNN regression is used to calculate average of targeted value of k nearest neighbors. Using inverse distance weighted average of the k nearest neighbors can also be calculated. KNN classification and KNN regression uses the same distance functions.

With the help of the following functions distance between neighbors are measured:

Euclidean

Manhattan

Minkowski

These equations can only be used for continuous variables. For categorical variables Hamming distance must be used. This measures the number of instances where different corresponding symbols are in two strings of equal length.

Hamming Distance

Inspecting the data the ideal value for K is chosen. With a large K value the noise is reduced but it becomes harder to detect the distinct features. To determine good K value using independent data set to validate K values, cross- validation is an ideal way. The ideal K for most datasets is 10 or more which produces better results than 1-NN.

Standardized Distance

Standardizing the training set can overcome the difficulty to calculate distance measures directly from the training set where there is a mixture of numerical and categorical variables.

Using the standardized distance on the same training set, the unknown case returned a different neighbors which is not a good sign of robustness.

**Support Vector Machine – Regression**:

Support Vector Machine can also be used as a method of regression, keeping all the fundamental elements intact that designate the algorithm (maximal margin). With scarcely trivial distinction, the Support Vector Regression utilizes the same postulates as the SVM for categorization. Firstly, because it has indefinite possibilities, the prediction of the resulted real number becomes perplexing. In case of regression, a margin of tolerance (epsilon) is allocated in approximation to the SVM which would have already arise from the problem. Aside from this, there is another complication. Thereby the algorithm is further more problematic to be accepted. Nevertheless, the gist stays the unchanged: diminishing error, singularizing the hyperplane which maximizes the margin, realizing the part of the error is tolerated.

Linear SVR:

Non-Linear SVR:

Polynomial:

Gaussian Radial Basis function:

**Decision Tree – Regression with AdaBoost:**

Decision tree establishes regression models in the form of a tree structure. It simultaneously jots down dataset onto smaller and smaller subsets and aligned decision is incrementally established. The end result is a tree with decision nodes and leaf nodes. A decision node has two or more sectors, each indicating values for the feature tested. A numerical target is illustrated by leaf node. The highest decision node in a tree which becomes equivalent to the leading predictor is called root node. Both categorical and numerical data can be conducted by decision trees. The key advantages of trees are that they can be rapidly instructed (say, as distinguished between neural networks) and are non-parametric. The major disadvantages are that the space has limits that are parallel to the characteristic axes and representation based on powers or products of the features are disapproved. Making decision surfaces that are oblique to the axes are possible and using so called oblique decision trees [Bradley and Utgoff (1995), Ittner and Schlosser (1996), Murthy et.al. (1993). Mouth et.al. (1994)] and as a matter of fact CART has that option. Moreover, in lieu of the input to the tree being the features, we could have maintained both the products of features and features raised to some powers. All these alternatives make the constructing of trees prolonged.

# CHAPTER 03

# Proposed Model

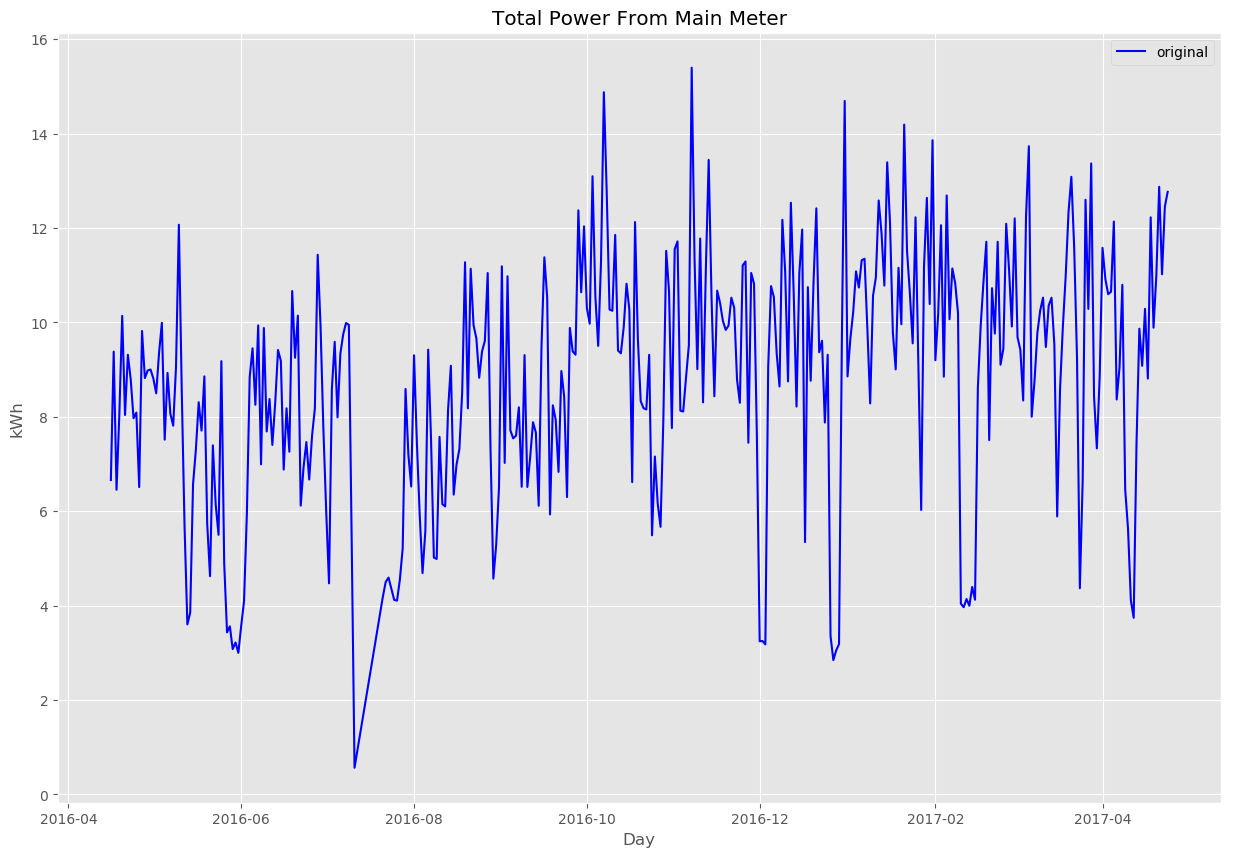
The IoT based Short Term Load forecasting system is consist of IoT devices and a central processing unit. IoT devices are configured to upload power usage data to the server. The central processing unit is configured to do calculation based on a selected machine learning algorithm. Based on the calculation and learning process the unit will give prediction.

## 3.1 Process

There are five steps to give Short Term Load Forecasting:

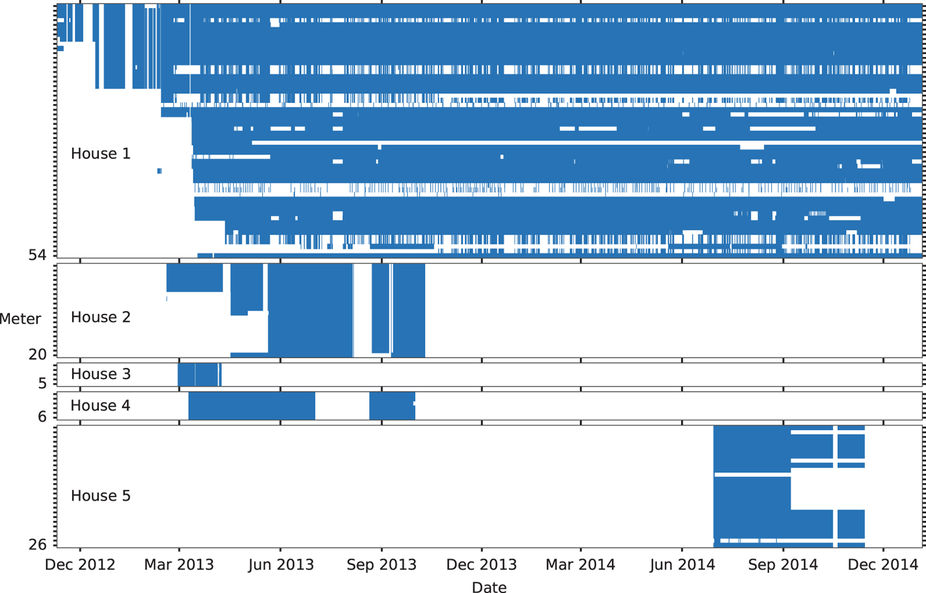
* Data collection from IoT devices
* Preprocessing and filtering
* Training machine learning model
* Give prediction

## 3.2 Data Collection



Main idea of this project is forecast total power consumption based on the data collected from the IoT devices. Due to limitations of time and resources we could not configure devices to upload power uses data to the internet. In this project we have used “The UK-DALE dataset” created by Jack Kelly & William Knottenbelt. This datasets contains appliance level disaggregated power consumption record as well as aggregated whole house power consumption record. In figure[] whole house power consumption of last one year is given. We assumed that all the appliances in this datasets are IoT devices. In table[] a short overview of the whole dataset is given. For training and testing we have used data of house\_1 because it contains maximum number of appliances. Also they have given more emphasis on recording house\_1 data. In figure[] shows that house\_1 data are more consistent than other houses.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| House | 1 | 2 | 3 | 4 | 5 |
| Number of occupants | 4 | 2 | 2 | 2 | 2 |
| Description of occupants | 2 adults and 1 dog started living in the house in 2006. One child born in 2011. Second child born in 2014. | 2 adults. 1 at work all day; the other sometimes home | 1 adult and 1 pensioner |  | 2 adults |
| Total number of meters | 54 | 20 | 5 | 6 | 26 |
| Date of first measurement | 2012-11-09 | 2013-02-17 | 2013-02-27 | 2013-03-09 | 2014-06-29 |
| Date of last measurement | 2017-04-26 | 2013-10-10 | 2013-04-08 | 2013-10-01 | 2014-11-13 |

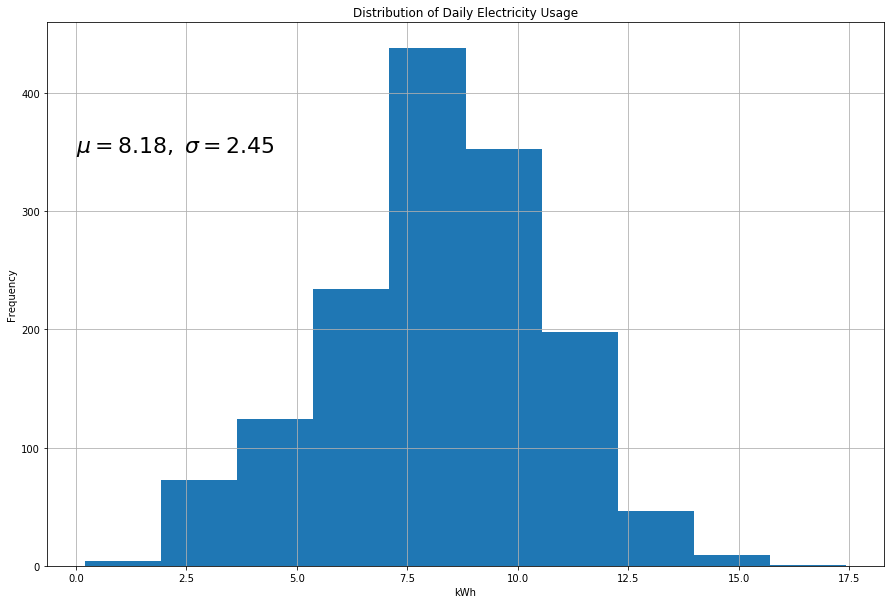


## 3.2 Preprocessing and Filtering Data

Noise and misleading data in any dataset are bad for any model to train on. A misleading dataset will eventually produce a hypothesis which will not do well in unseen data. Therefore noise cancellation has done with great care. The dataset contains UML configuration file for every houses. The file has details description of meter devices and appliances. Each appliance has upper bound and lower bound of power consumption. Any power consumption beyond that limit is considered as noise or bad reading. In table[] upper and lower bound of every device recorded in house\_1 is given.

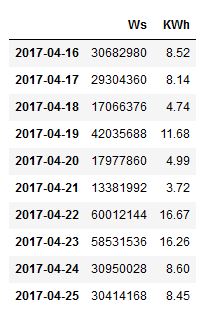
The main dataset contains five folder. Each folder correspond to each house. Under each house numbers of CSV files according to number of devices are given. Each CSV file contains records of power consumption with time. In each CSV file time is give in format of UNIX time epoch. Interval data recording is six seconds. In figure[] a screenshot of first 10 rows of channel\_6 which is the records of power consumption of dishwasher is given. In figure[] we can see CSV file contains two column and column\_0 contains time and column\_1 contains power consumption record.

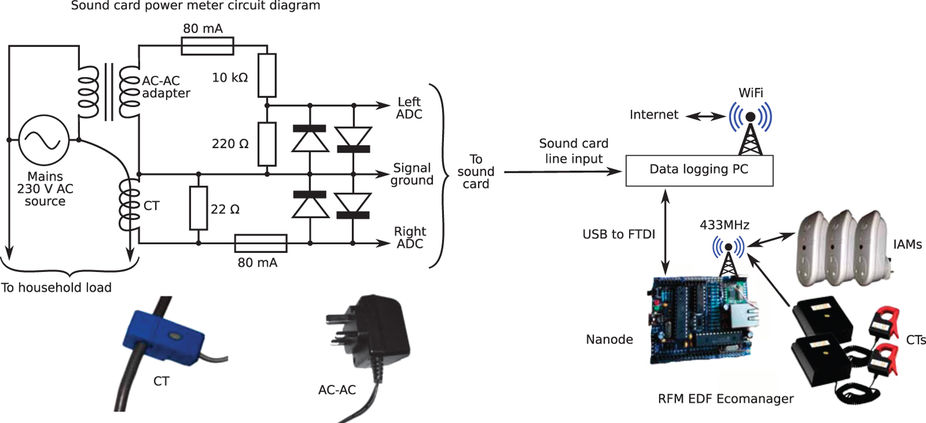
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |



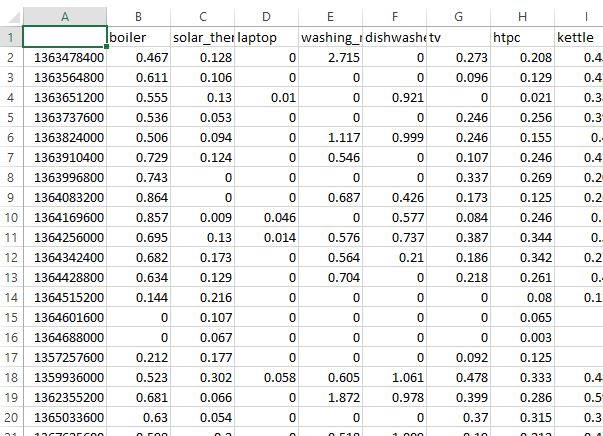
We have used pandas, a powerful library written in python for data analysis and manipulation. Steps for filtering every CSV file is given below.

* With the help of pandas we converted UNIX time epoch to human readable date and time.
* We dropped all the data record that do not comply with the bound given in table[]. In figure[] a screen shot of noise free reading is given. Now we can say that the dataset does not contain any noise according to table[]. Power consumption records are taken via external meter fig[]. The meters require some power to operate and not factory standard. Therefore error is obvious in their reading. In IoT devices power consumption reading capability will be integrated in their internal circuit and will not face this kind of error.
* Goal of this project is to give a day ahead forecast. Therefore we do not need six second interval datasets. Six second datasets are then resampled by day. Then total power consumption of the day is converted into kWh. Figure[].
* House power consumption has good relation with weather condition. Therefore we have added average temperature, average humidity and average wind speed of every day.
* In figure[] distribution graph of total power consumption is given. From this distribution we have dropped lower than (mean - 2 \*σ) and greater than (mean + 2 \*σ)
* After completing aforementioned steps all the CSV files are concatenated into a single CSV file. Then the date and time was again converted into UNIX time epoch. Reason of converting is, date time format changes in different computer due to different version of software. UNIX time epoch is an integer number. As a result date and time remain intact in every computer that want to work on the dataset. In figure[] a partial screen shot of final CSV file is given.





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Channel ID | Name | Min Power (Watt) | Max Power (Watt) | Type |
| 2 | boiler | 70 | 4000 | Apparent |
| 3 | solar\_thermal\_pump | 43 | 4000 | Apparent |
| 4 | laptop | 70 | 4000 | Active |
| 5 | washing\_machine | 20 | 4000 | Active |
| 6 | dishwasher | 10 | 4000 | Active |
| 7 | tv | 10 | 4000 | Active |
| 8 | kitchen\_lights | 50 | 4000 | Apparent Sub meter of channel 25 |
| 9 | htpc | 20 | 4000 | Active |
| 10 | kettle | 2000 | 4000 | Active |
| 11 | toaster | 1000 | 4000 | Active |
| 12 | fridge | 50 | 4000 | Active |
| 13 | microwave | 200 | 4000 | Active |
| 14 | lcd\_office | 40 | 4000 | Active |
| 15 | hifi\_office | 9 | 4000 | Active |
| 16 | breadmaker | 500 | 4000 | Active |
| 17 | amp\_livingroom | 25 | 4000 | Active |
| 18 | adsl\_router | 6 | 4000 | Active |
| 19 | livingroom\_s\_lamp | 16 | 4000 | Active |
| 20 | soldering\_iron | 50 | 4000 | Active |
| 21 | gigE\_&\_USBhub | 5 | 4000 | Active |
| 22 | hoover | 1200 | 4000 | Active |
| 23 | kitchen\_dt\_lamp | 13 | 4000 | Active |
| 24 | bedroom\_ds\_lamp | 26 | 4000 | Active |
| 25 | lighting\_circuit | 40 | 4000 | Apparent |
| 26 | livingroom\_s\_lamp2 | 86 | 4000 | Active |
| 27 | iPad\_charger | 7 | 4000 | Active |
| 28 | subwoofer\_livingroom | 15 | 4000 | Active |
| 29 | livingroom\_lamp\_tv | 13 | 4000 | Active |
| 30 | DAB\_radio\_livingroom | 300 | 4000 | Active |
| 31 | kitchen\_lamp2 | 20 | 4000 | Active |
| 32 | kitchen\_phone&stereo | 5 | 4000 | Active |
| 33 | utilityrm\_lamp | 35 | 4000 | Active |
| 34 | samsung\_charger | 4 | 4000 | Active |
| 35 | bedroom\_d\_lamp | 45 | 4000 | Active |
| 36 | coffee\_machine | 1000 | 4000 | Active |
| 37 | kitchen\_radio | 2 | 4000 | Active |
| 38 | bedroom\_chargers | 2 | 4000 | Active |
| 39 | hair\_dryer | 1600 | 4000 | Active |
| 40 | straighteners | 170 | 4000 | Active |
| 41 | iron | 1700 | 4000 | Active |
| 42 | gas\_oven | 11 | 4000 | Active |
| 43 | data\_logger\_pc | 12 | 4000 | Active |
| 44 | childs\_table\_lamp | 14 | 4000 | Active |
| 45 | childs\_ds\_lamp | 10 | 4000 | Active |
| 46 | baby\_monitor\_tx | 15 | 4000 | Active |
| 47 | battery\_charger | 20 | 4000 | Active |
| 48 | office\_lamp1 | 14 | 4000 | Active |
| 49 | office\_lamp2 | 10 | 4000 | Active |
| 50 | office\_lamp3 | 7 | 4000 | Active |
| 51 | office\_pc | 100 | 4000 | Active |
| 52 | office\_fan | 20 | 4000 | Active |
| 53 | LED\_printer | 400 | 4000 | Active |



## 3.7 Training Machine Learning Model

### 3.7.7 Programming Language and Tools

Programming Languages

* Python 3.5

List of Frame Work and Libraries based on Python

* ScikitLearn
* Tensorflow
* Keras
* Pandas
* numPy
* MatplotLib

### 3.7.7 Long Short-term Memory Network

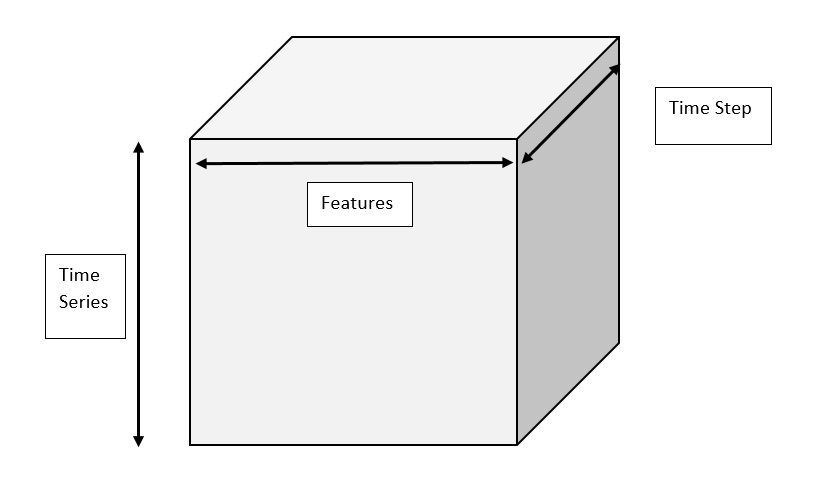
LSTM networks are renowned for their ability to remember pattern and sequence. Human behavior tends to be repetitive. From this intuition we used LSTM Network to learn the behavior pattern of power usages. Also, unlike normal datasets time series has a complexity of order dependence between items and sequence. Long Short-term Memory Network were made to deal with these kind of sequence dependencies. To build LSTM Network, programming was done in Python 3.5. Python framework especially Pandas was used to read dataset from a CSV file. To make the dataset suitable for LSTM, numPy was used to reshape the dataset. We used TensorFlow with a rapper called Keras.

Input to a LSTM Network is 3D matrix. Datasets we have used in this project is 2D matrix which consist of columns and rows. Number of columns corresponds to dimension of feature vector and number of rows corresponds to number of data points. Input of LSTM Network consist of another dimension of the matrix, which corresponds to time steps. LSTM network use this time steps to keep track of the previous occurrence. In figure [] graphical view of the input of a LSTM Network is given.

Structure of LSTM Model

* Number of LSTM Cell: One.
* Batch Input Shape: (Batch Size \* Time Step \* Input Vector Dimension)
* Batch Size: 10 sequence of samples
* Time Step: 10. Every sequence consist of today and previous 10 days of power consumption reading
* Input Vector Dimension: 55
* Activation: tanh
* Recurrent activation: Hard sigmoid
* Return Sequence: True

To prevent over fitting of the model a threshold was given. A program always keep track on difference between loss of training set and loss of test set. When loss of training set is decreasing and loss of test set is increasing and difference is bigger than threshold, it stops training the model.



To prevent over fitting of the model a threshold was given. A program always keep track on difference between loss of training set and loss of test set. When loss of training set is decreasing and loss of test set is increasing and difference is bigger than threshold, it stops training the model.

**3.8 K-Nearest Neighbors Regression**

Neighbors-based regression uses continuous data labels rather than discrete variables. The query point is appointed a label which is the mean of the labels of nearest neighbors.

Two different neighbors regressor are implemented by scikit-learn those are where the implementation is done learning the *k* nearest neighbors of each query and *k* is user specified integer. And is implemented learning about the neighbors in a fixed radius r of the query point and r is user floating-point value.

Uniform weights are used by the basic nearest neighbors that is the classification of query point is contributed uniformly by each point in local neighborhood. In some conditions, to weight points in a way that the nearby points accord more to regression than faraway points can give us advantage. This can be done by using the keyword. is the default value assigned to all points with equal weights and from the query points the weights proportional to the inverse of distance is allocated by. In other way, to compute the weights user can define a function of the distance.

Brute Force

In machine learning the committed area of research is fast computation of nearest neighbors. The most raw neighbor search implementation is the brute-force computation of distances among all pairs of points in the dataset. is scaled where is the number of samples and is dimensions. For small data samples brute-force neighbor search can be very moderate. However with the increase in the number of samples the brute-force approach becomes unworkable. The search by brute-force neighbors in the class is limited to the keyword and computation is done using the routines available in.

K-D Tree

Tree based data structure have been invented to meet the computational ineffectual of brute-force approach. By encoding aggregate distance information from the sample these data structure reduces the number of distance calculation. For example if point A is very far from point B and point B is very near to point C then the points A and C are very far, this can be found without separately calculating their distance. This way the cost of nearest neighbors can be minimized to. This is a notable upgrade over brute-force for large.

KD tree data structure derives two-dimensional Quad-trees and 3-dimensional Oct-trees to an arbitrary number of dimensions. In KD tree the parameter space is recursively partitioned along the data axes and data points are filled dividing the space into nested orthotropic regions. Since the partitioning is done only along data axes the construction of KD is very fast and D-dimension is not computed. After construction the nearest neighbor of query point can be determined with. But with the growth of D this approach becomes inefficient. In scikit-learn KD tree neighbors uses class for computation and searches are specific to keyword.

Ball Tree

The ball tree data structure was developed to meet the inefficiencies of KD tree in higher dimensions. Ball tree data structure partitions data in a series of nesting hyper-spheres whereas KD tree did along Cartesian axes. This gives very effective result in very high dimensions but also increases the cost.

In ball tree data is divided recursively into nodes interpreted by centroid and radius r in a way that each point lies in the hyper-sphere defined by and. By triangle inequalitythe number of points for neighbor search is reduced. With this the single distance between a tests point and the centroid can be calculated which is sufficient to establish a lower and upper bound on the distance to all points within the nodes. The spherical geometry of ball tree nodes surpasses the KD tree in high dimensions even though the performance is mainly dependent on the structure of training data. In scikit-learn the neighbors searches are specific to keyword and are computed using class.

Choice of Nearest Neighbors Algorithm

The ideal algorithm for a given dataset depends on certain factors.

* Number of samples and dimensionality.

- Brute force query time grows as

- KD tree query time depends on. For small the cost is and is very efficient but with the growth of the cost increase to.

- Ball tree query time grows approximately.

For small data set less than 30 or so then brute force is more efficient than tree based approach.

* Data structure
* Brute force query time is unchanged by data structure.
* Data structure influences both Ball tree and KD tree. The data with small dimensionality leads to faster query time. It will not show much improvement as ball tree for arbitrarily structured data because the KD tree internal representation is aligned with parameter axes.
* Requested number of neighbors k for a query point.
* The value of does not affect brute force query time.
* With the increase of Ball tree and KD tree query time becomes slower. This is due to necessity of searching a large portion of parameter space and as the tree is traversed internal queueing is required.
* Number of query points

In Ball tree and KD tree the cost of construction in construction phase becomes negligible for many queries. The construction cost take a significant fraction of the total cost for small number of queries. So for small queries brute force is better than tree based method.

If the selects ‘’ and the ‘’ is in the ‘’ list of ‘’.

If it selects ‘’ and the ‘’ is in the ‘’ list of ‘’.

If it selects ‘’ and the ‘’ is not in the ‘’ list of or.

If it selects ‘’. This is done on the assumption of the number of query points same as the number of training points and is close to default value of 30.

Effects of

As stated before, brute force search is more efficient than a tree based query for small sample sizes. This can be resolved in ball tree and KD tree by switching to brute force searches internally within leaf node. This can be defined with the parameter which has many effects.

Construction time- In large fewer nodes needs to be created which leads to fast tree construction time.

Query time- Large or small can lead to minimal query cost. Iftends to 1 then traversing nodes requires slow query times. Ifis closer to the size of training set the queries becomes brute force.

Memory- With the increase of the memory to store a tree structure decreases. This is important in case of ball tree as it stores D-dimensional centroid for each node. is the size of the training set space is needed to store.

is not used in brute force queries.

We used 15 neighbors () i.e. depending on the values 15 neighbors it gives prediction.

**Support Vector Regression**

The function of Support Vector Classification can be extended further to solve regression problems. This method is called Support Vector Regression.

The cost function of building a Support Vector Classification model does not count the training points which are beyond specific margin. As a result, the model depends on a subset of the training data. Similarly the cost function of building a Support Vector Regression leaves out any training data close to the model prediction. For which, the model depends on a subset of the training data.

#NEED to paraphrase

Support Vector Machines are powerful tools, but their compute and storage requirements increase rapidly with the number of training vectors. The core of an SVM is a quadratic programming problem (QP), separating support vectors from the rest of the training data. The QP solver used by this libsvm-based implementation scales between O(n\_{features} \times n\_{samples}^2) and O(n\_{features} \times n\_{samples}^3) depending on how efficiently the libsvm cache is used in practice (dataset dependent). If the data is very sparse n\_{features} should be replaced by the average number of non-zero features in a sample vector.

#need to paraphrase

In scikit-learn the support vector machines support both dense and sparse sample vectors as input. For the efficient performance,-ordered (dense) or (sparse) with data type float64 is preferred.

In the library of scikit-learn three different implementation of SVR is given. They are:

A faster implementation is possible in than as it only considers linear kernels. We have implement in our dataset. Mathematical formulation of is given below:

Assuming a set of training points, where is a feature vector and and is the target output. If the given parameters and then according to Vapnik Support Vector regression is

The twofold problem is

After solving the above equation, the approximate function is

In the above equation notations are,

In scikit-learn these parameters can be accessed through  which contains the difference between, which holds the support vectors, and which holds the independent term.

The fit method trains the model which takes as argument vectors where is expected to have floating point values. We used default parameters of library for implementation of SVR where Penalty parameter =1.0, =0.2 (specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value), (only for Gaussian Radial Basis function and Polynomial kernel).

Decision Tree Regression with AdaBoost:

is a class capable of performing regression on a dataset. Like with other regressions, takes as input two arrays. An array of size of the training points, and an array of integer values having size [n\_samples] of the class labels for the training points.

An efficient implementation for the construction of decision tree is offered by scikit-learn. By presorting the features before training and keeping the label count the total cost of the algorithm becomes. This is optional for all tree based algorithms.

Tree algorithms:

In 1986 Ross Quinlan developed ID3 (Iterative Dichotomiser 3). This algorithm creates a multiway tree in which finds the categorical feature for each node that yields the largest information gain for categorical target. These trees are grown as big as possible and then pruned to improve the ability of the tree to generalize to unseen data.

C4.5 descendant of ID3. The trained trees are converted to sets of if-then rules by C4.5. The accuracy of each rule assessed to determine the order in which they should be applied. Removing a rule’s precondition if the accuracy of the rule improves without it is done by pruning.

C5.0 is the latest version of Quinlan released under a proprietary license. It is more accurate than C4.5 using less memory and builds smaller rulesets.

CART (Classification and Regression Trees) is same as C4.5 but it does not compute rule sets and it supports numerical target variables. Binary trees are constructed using the feature and threshold that yield the largest information gain at each node.

Optimized version of CART algorithm is used by scikit-learn.

Mathematical Formulation:

Assuming training vectors and targeted values containing vector, partition of the space is made recursively such that the samples with the same labels are in a group.

Suppose the data at node be represented by. Splitting is done using having of an attribute and threshold. After partitioning the data is kept into and subsets and. Noise at is calculated using a function which calculates impurity. The choice of function depends on the method of solving (classification or regression).

To minimize impurity parameters are selected using

It is continued for subsets and until the maximum depth is reached where.

#need to paraphrase

The module [**sklearn.ensemble**](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble) includes the popular boosting algorithm AdaBoost, introduced in 1995 by Freund and Schapire [[FS1995]](http://scikit-learn.org/stable/modules/ensemble.html#fs1995).

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights w_1, w_2, …, w_N to each of the training samples. Initially, those weights are all set to w_i = 1/N, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence

The number of weak learners is controlled by the parameter n\_estimators. The learning\_rate parameter controls the contribution of the weak learners in the final combination. By default, weak learners are decision stumps. Different weak learners can be specified through the base\_estimator parameter. The main parameters to tune to obtain good results are n\_estimators and the complexity of the base estimators (e.g., its depth max\_depth or minimum required number of samples at a leaf min\_samples\_leaf in case of decision trees).

#need to paraphrase

Decision Tree Regression with AdaBoost is a powerful model. For parameters we used for Decision Tree Regression. That is, the depth of tree can be maximum 16. For boosting parameter is used. That is, 299 decision tree is compared with a single tree regressor. If we increase this number, the regressor can fit more details.

Root Mean Squared Error:

#need to paraphrase

The root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The RMSD represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed out-of-sample. The RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSD is a measure of accuracy, to compare forecasting errors of different models for a particular data and not between datasets, as it is scale-dependent.

#need to paraphrase

The RMSE of an approximation with respect to a real value is defined as the square root of the mean square error:

The RMSE of predicted data for survey of , for variables is calculated for numbers of cases using the following formula:

Then RMSE is normalized using:

Scikit-learn has a class to measure mean squared error of a model which is. Its takes input of array containing original values and containing predicted values of a model then gives a floating number value as output which is MSE of the model. Then we used function to calculate RMSE as RMSE is square root of MSE.

# CHAPTER 04

## Experimental Setup & Result Analysis

The algorithms Nearest Neighbors Regression, Support Vector Machine Regression, Decision Forest with AdaBoost is performed in Intel(R) Core™ i5-4200 [CPU@ 1.60GHz](mailto:CPU@1.60GHz) 2.30GHz, 8,192MB RAM using environment PyCharm IDE. Python v3.6.2 was used as coding language. The python library we used for building the above mentioned algorithms scikit-learn v0.19.1. It is an open source, commercially usable –BSD license. It has simple and efficient tools for data mining and data analysis.

|  |  |
| --- | --- |
| Algorithms | Error |
| Root Mean Squared Error |
| Nearest Neighbors Regression | 1.93 |
| Support Vector Machines  Regression | 1.83 |
| Decision Forest Regression  With AdaBoost | 1.86 |
| Long Short Term Memory | 1.65 |

As our goal is to predict the power consumption of the house the next day depending on the power usage of the appliances of the present day. Model is evaluated on test set. The error result is obtained from test set. For preprocessing we scaled the data using the library function. We used splits for splitting the dataset as it’s a time series dataset. Later we trained out model on train set and the error rate in the above table is acquired by evaluating the models against our test set. We used all the default settings of the library and few changes in parameters of few algorithms. We used an API out of 3 APIs of the library to measure our trained model’s performance. The API scoring parameter contains model-evaluating tools using cross-validation depends on an internal scoring strategy. We used this API to find the RMSE of our models and we compared the algorithms based on the result of the API.

Nearest Neighbors Regression:

Nearest Neighbors has RMSE of 1.9331727. As our goal is to predict the total power consumption of a house of the next day depending on the usage of the today’s power consumption of the appliances, for which we are getting this much higher RMSE value. In fig[] the predicted values of the model and true values are compared.

Support Vector Regression:

The kernel RBF performs better than other kernels. RBF kernel performs better in this context because of the data. The higher the degree, the performance of other kernels are worse than RBF. SVR using kernel polynomial and Gaussian Radial Basis function with degree of 3 has RMSE error of 2.1229618 and 1.8341087 respectively. But kernel linear performs slightly lesser than RBF but better than the polynomial as it has a degree of 1. It has RMSE of 1.8474361. For having the lowest RMSE among the kernel function, we preferred the kernel function RBF for SVR.

Decision Tree Regression with AdaBoost:

The RMS error of this model is 1.9202281.

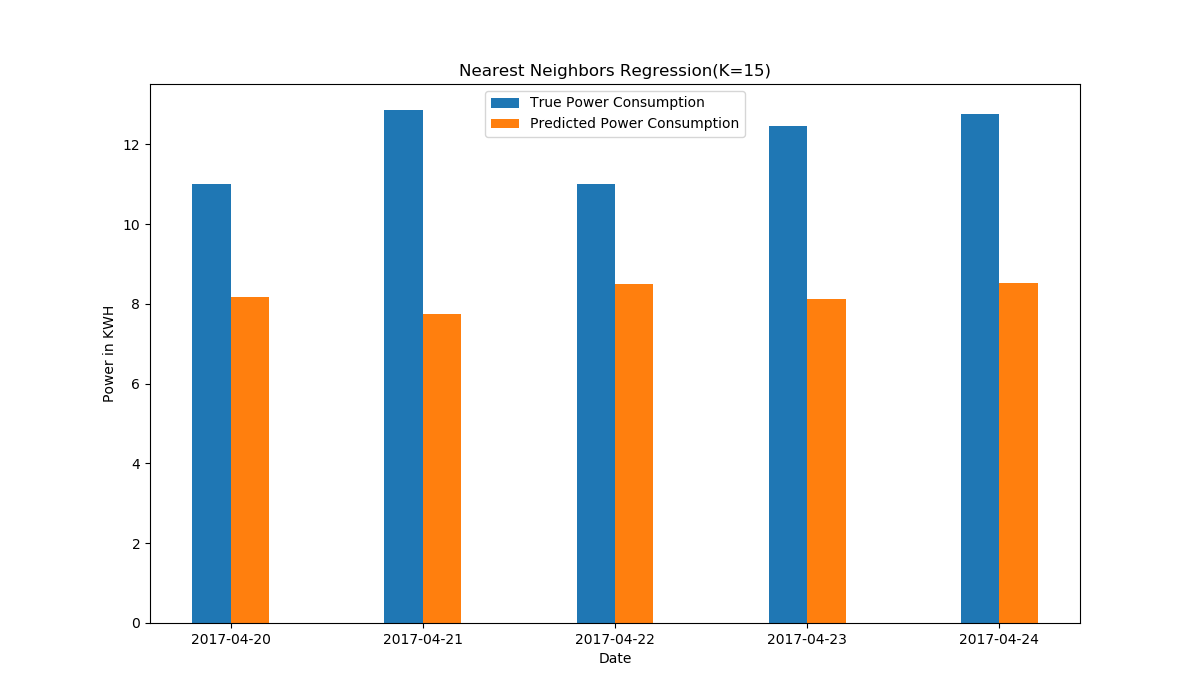
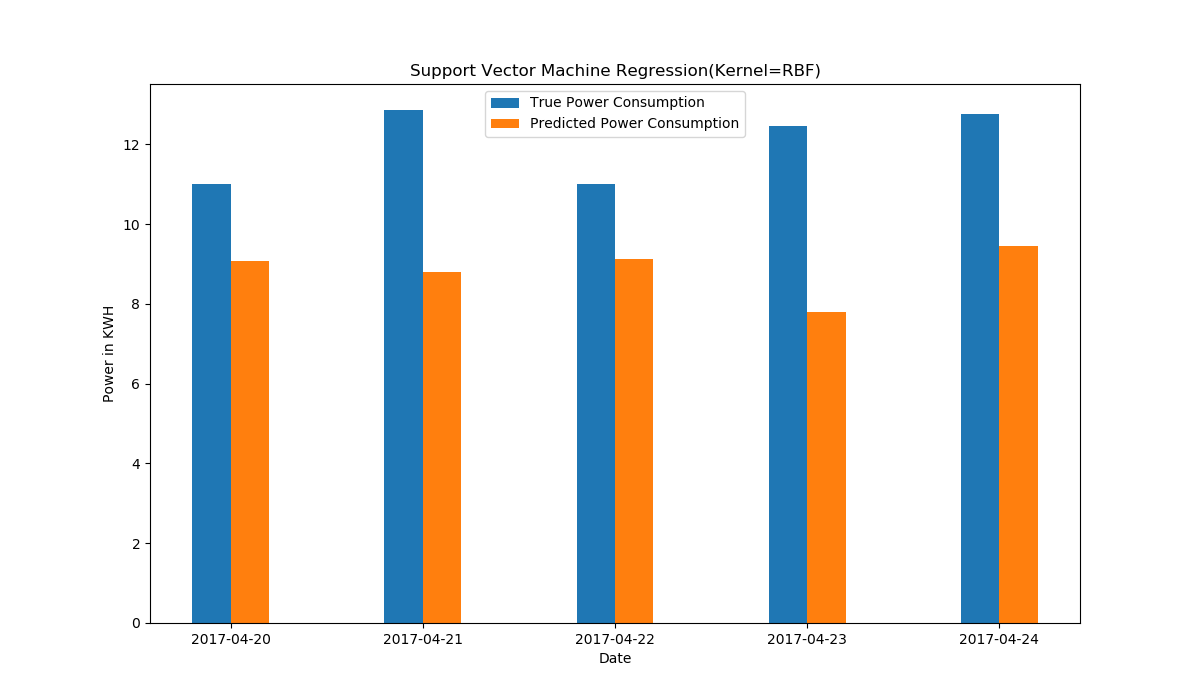


Fig-1: Real power consumption vs Predicted power consumption (Nearest Neighbors Regression)

In fig-1 the comparison of the real value and the predicted value of the trained model Nearest Neighbors Regression is given.

 Fig-2: Real power consumption vs Predicted power consumption (Support Vector Machine Regression)

In fig-2 the performance of Support Vector regression (kernel RBF) is shown by comparing the true value with predicted value. The predicted power consumption is the output of the model.

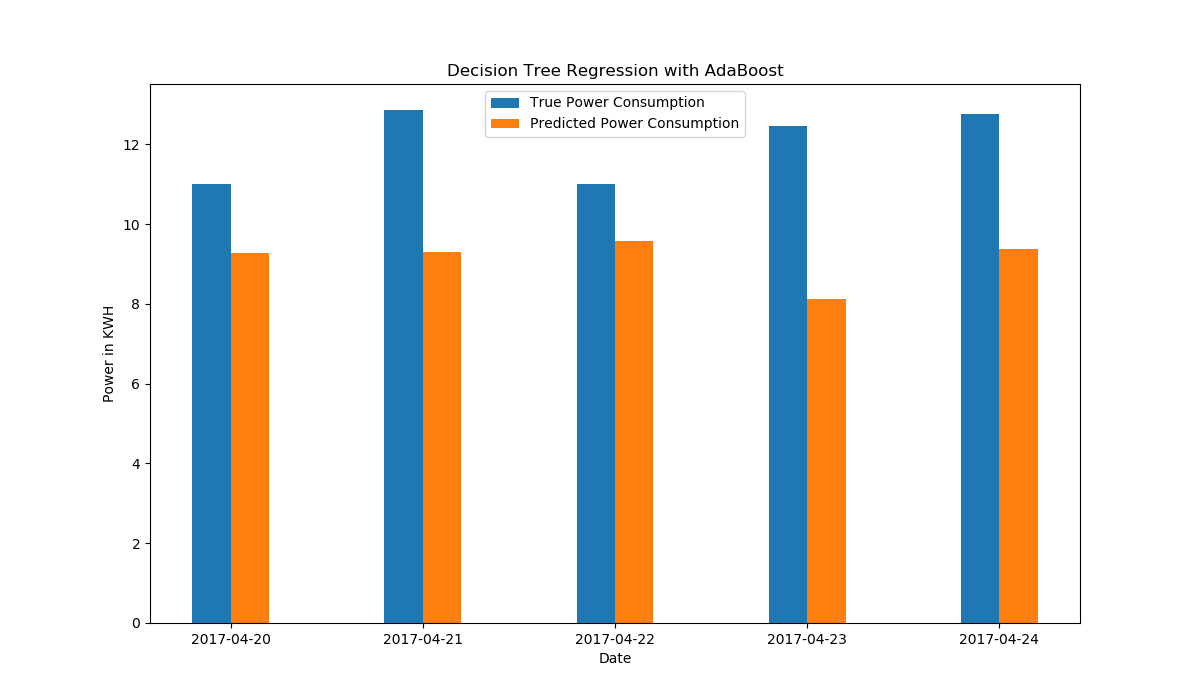


Fig-3: Real power consumption vs Predicted power consumption (Decision Tree Regression with AdaBoost)

In fig-3 Evaluation of the model Decision Forest Regression with AdaBoost is shown by comparing predicted and real value.

Done Till This….!!

The experiment of this algorithm in this thesis is performed in the AMD FXtm-8300 Eight-Core Processor, ~3.3GHz, 16384MB RAM environment using Spyder IDE. The codes were done in python v3.6.2. We used TextBlob library for building the Naïve Bayes classifier. We used pythons NLTK (natural language toolkit) for natural language processing basics. Our training dataset for naïve Bayes classifier were manual datasets we collected from twitter and later we used a dataset of amazon unlocked mobile [15].

In the classifier we have calculated the polarity. The polarity range is (-1.0 to 1.0) and if the polarity is less than 0 then the sentence is negative. If the polarity of the sentence is 0.0 then the sentence is neutral. Thus if the polarity greater than 0.0 then the sentence is positive.

* Positive ( >0.0)
* Neutral (0.0)
* Negative (<0.0)

Figure 9 shows the graph for the review of processors of different phone models. The graph shows the positive and negative polarity of the sentiment.

|  |
| --- |
|  |

**Figure 9: Processor sentiment graph**

In Figure 9, the bars represent the number of positive, negative and neutral sentiment of Samsung Galaxy S8, IPhone X and Google Pixel 2’s processor. Blue bar illustrates Samsung Galaxy S8, Orange bar illustrates Apple IPhone X and Gray bar illustrates Google pixel2.

**Table 1: Processor attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 826 | 240 | 606 |
| Positive | 231 | 59 | 214 |
| Negative | 162 | 361 | 66 |
| Neutral | 433 | 690 | 326 |

|  |
| --- |
|  |

**Figure 10: Camera sentiment graph**

In Figure 10, the bars represent the number of positive, negative and neutral sentiment of Samsung Galaxy S8, IPhone X and Google Pixel 2’s camera.

**Table 2: Camera attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 24747 | 9289 | 6989 |
| Positive | 15182 | 5669 | 4537 |
| Negative | 1236 | 881 | 355 |
| Neutral | 8329 | 2739 | 2097 |

|  |
| --- |
|  |

**Figure 11: Battery sentiment graph**

In Figure 11, the bars represent the number of positive, negative and neutral sentiment of Samsung Galaxy S8, IPhone X and Google Pixel 2’s battery.

**Table 3: Battery attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 938 | 1198 | 739 |
| Positive | 235 | 309 | 146 |
| Negative | 211 | 262 | 206 |
| Neutral | 492 | 627 | 387 |

|  |
| --- |
|  |

**Figure 12: Price sentiment graph**

In Figure 12, the bars represent the number of positive, negative and neutral reviews of Samsung Galaxy S8, IPhone X and Google Pixel 2’s price.

**Table 4: Price attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 3198 | 1573 | 1129 |
| Positive | 1369 | 359 | 422 |
| Negative | 257 | 681 | 163 |
| Neutral | 1572 | 533 | 544 |

|  |
| --- |
|  |

**Figure 13: Display sentiment graph**

In Figure 13, the bars represent the number of positive, negative and neutral reviews of Samsung Galaxy S8, IPhone X and Google Pixel 2’s display.

**Table 5: Display attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 2076 | 1844 | 919 |
| Positive | 922 | 826 | 451 |
| Negative | 158 | 186 | 102 |
| Neutral | 996 | 832 | 366 |

|  |
| --- |
|  |

**Figure 14: UI sentiment graph**

In Figure 14, the bars represent the number of positive, negative and neutral reviews of Samsung Galaxy S8, IPhone X and Google Pixel 2’s UI.

**Table 6: User Interface attribute based sentiment count table.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samsung Galaxy S8 | Apple iPhone X | Google Pixel 2 |
| Total | 1238 | 885 | 479 |
| Positive | 568 | 364 | 206 |
| Negative | 45 | 96 | 52 |
| Neutral | 625 | 423 | 221 |



|  |
| --- |
|  |

**Figure 15: Time vs. Polarity graph of Samsung Galaxy S8**

The above graph is a time vs. polarity graph of Samsung Galaxy S8. In this graph, x-axis is the time and y-axis is the polarity. As this model was released before several months, we averaged the polarities of data from one month and input the results with respect to the month. This graph shows the users preference shifting with respect to time in a complex method. We did not include the attribute price on this graph as the cost of cellphones do not change much resulting in the sentiment of people being constant.

|  |
| --- |
|  |



**Figure 16: Time vs. Polarity graph of Apple IPhone X**

The above graph is a time vs. polarity graph of Apple IPhone X. In this graph, x-axis is the time and y-axis is the polarity. As this model was released before few months, we averaged the polarities of data of one day and input the results with respect to the month. This graph shows the users preference shifting with respect to time in a complex method. We did not include the attribute price on this graph as the cost of cellphones do not change much resulting in the sentiment of people being constant.

|  |
| --- |
|  |

**Figure 17: Time vs. Polarity graph of Google Pixel 2**

The above graph is a time vs. polarity graph of Google Pixel 2. In this graph, x-axis is the time and y-axis is the polarity. As this model was released before few months, we averaged the polarities of data of one day and input the results with respect to the month. This graph shows the users preference shifting with respect to time in a complex method. We did not include the attribute price on this graph as the cost of cellphones do not change much resulting in the sentiment of people being constant.

# CHAPTER 05

# Conclusion & Future Work

**5.1 Conclusion**

In this paper, we have proposed a general product rating system based on public opinion. This can be widely used in the future to get proper reviews of any product to get the best review possible for a product. This system will allow user to briefly analyze bunch of products based on their rating and choose the best one depending on the budget and requirement of the user. This system is reliable as the rating is basically generated based on public opinion and this rating will reflect the actual state of the product in the market of buyers with limited information on the product.

**5.2 Future Work**

We have built this system to work on cell phone’s rating. Our basic plan was to implement the system for any product. In future we wish to work to integrate the system so that it can generate rating for any product. That would not require much effort as we can always collect data from twitter on any product and we can train more datasets in our created classifier. We wish to make this system user friendly more.

We look forward to ease the process of decision making about buying any product for people. This system would be just one more step towards digitalization. We belief it will make people’s lives easier as this system will allow users to see an accurate rating of any product and save them from the misery of finding a good review from various review sites.

# Reference

1. Vishal A. ,S.S. Sonawane, “Sentiment Analysis of Twitter Data:A Survey of Techniques” 2016 volume 139 No.11
2. Prerna Chikersal, Soujanya Poria, and Erik Cambria “SeNTU: Sentiment Analysis of Tweets by Combining a Rule-based Classifier with Supervised Learning”
3. M. Vadivukarassi, N. Puviarasan and P. Aruna “Sentimental Analysis of Tweets Using Naive Bayes Algorithm”. World Applied Sciences Journal 35 (1): 54-59, 2017 ISSN 1818-4952
4. Xin Chen, Mihaela Vorvoreanu, and Krishna Madhavan “Mining Social Media Data for Understanding Students’ Experiences”, IEEE transactions on learning technologies, vol. 7, no. 3, july-september 2014
5. Luciano Barbosa Junlan Feng “Robust Sentiment Detection on Twitter from Biased and Noisy Data” Coling 2010: Poster Volume pages 36–44, Beijing, August 2010
6. Vimalkumar B. VaghelaBhumika M. Jadav Analysis of Various Sentiment Classification Techniques
7. Bingwei Liu ; Erik Blasch ; Yu Chen ; Dan Shen ; Genshe Chen “Scalable sentiment classification for Big Data analysis using Naïve Bayes”.
8. Liu, B., 2012. Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), pp.1-167.
9. A. Popescu and O. Etzioni, “Extracting Product Features and Opinions from Reviews,” Proc. Human Language Technology Conf. and Conf. Empirical Methods in Natural Language Processing, pp. 339- 346, 2005.
10. Chaovalit, P. and Zhou, L., 2005, January. Movie review mining: A comparison between supervised and unsupervised classification approaches. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on (pp. 112c-112c). IEEE.
11. Gobinda G. Chowdhury (2006). Natural language processing.
12. Bing Liu list of words, http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar
13. Bill McDonald list of words, http://www3.nd.edu/~mcdonald/Word\_Lists.html,
14. Montejo-Ráez, A., Martínez-Cámara, E.,Martín-Valdivia, M., Ureña-López, L.: Random walk weighting over sentiwordnet for sentiment polarity detection on Twitter. WASSA '12 Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis
15. Promtcloud, Amazone unlocked phone review, https://www.kaggle.com/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones
16. [R. Creţulescu](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.R.%20Cre%C5%A3ulescu.QT.&newsearch=true) ; [A. David](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.A.%20David.QT.&newsearch=true) ; [D. Morariu](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.D.%20Morariu.QT.&newsearch=true) ; L. Vinţan “Part of speech tagging with Naïve Bayes methods” System Theory, Control and Computing (ICSTCC), 2014 18th International Conference 15 December 2014
17. Abinash Tripathy Ankit Agrawal, Santanu Kumar Rath Classification of Sentimental Reviews Using Machine Learning Techniques
18. Tripathy, A., Agrawal, A. and Rath, S.K., 2015. Classification of Sentimental Reviews Using Machine Learning Techniques.Procedia Computer Science,57, pp.821-829
19. Chaovalit, P. and Zhou, L., 2005, January. Movie review mining: A comparison between supervised and unsupervised classification approaches. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on (pp. 112c-112c). IEEE.
20. Pang, B., Lee, L. and Vaithyanathan, S., 2002, July. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10(pp. 79-86). Association for Computational Linguistics.
21. Y. Jo and A.H. Oh, “Aspect and Sentiment Unification Model for Online Review Analysis,” Proc. Fourth ACM Int’l Conf. Web Search and Data Mining, pp. 815-824, 2011.
22. Hu Minqing, Bing Liu, "Mining opinion features in customer reviews", Proceedings of the national conference on artificial intelligence, vol. 4, no. 4, pp. 755-760, 2004.
23. Mouthami, K., Devi, K.N. and Bhaskaran, V.M., 2013, February. Sentiment analysis and classification based on textual reviews. In Information Communication and Embedded Systems (ICICES), 2013 International Conference on (pp. 271-276). IEEE.
24. Gautam, G. and Yadav, D., 2014, August. Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In Contemporary Computing (IC3), 2014 Seventh International Conference on (pp. 437-442).IEEE.