Smart Predictive Analysis of Energy Consumption Using Cloud

-Team Number: 7, TA: Yue Shi

Aayush Pathak *aayushpa@usc.edu*
Nikita Ahuja *nikitaah@usc.edu*
Soundarya Singh *soundars@usc.edu*
Yesha Shah *yeshasha@usc.edu*
Ishan Patiyat (Team Leader) *patiyat@usc.edu*

***Abstract*- In this report, an application to analyze and predict the energy consumption for household customers is presented. The predictive user-interactive model has been created by using Linear Regression technique in R and ShinyApp package. Further, the entire application is deployed on AWS cloud platform. AWS EMR is used to reduce the large data in reduced meaningful data from which the data model can be extracted. AWS S3 is used to store the data and the code for the application. The results generated from the application give useful insights for the customers in terms of predicting the future bills and making smart decisions on controlling their power needs.**

Keywords- Big data, Smart Grid, Predictive Analytics, Cloud computing, Energy Prediction.

I. INTRODUCTION

Evolving technologies in the energy and utilities industry, including smart meters and smart grids, can provide companies and the users with unprecedented capabilities for forecasting demand, shaping energy usage patterns, preventing outages, optimizing unit commitment and more. At the same time, these advances also generate unprecedented data volume, speed and complexity [34]. To manage and use this information to gain insight, utility companies must be capable of high-volume data management and advanced analytics designed to transform data into actionable insights for the household users and customers. It will benefit the users in predicting their energy consumption and making smart decision to save energy and money.

Companies applying predictive analytics to smart energy initiatives can maximize their benefits by identifying the most cost-effective technologies to use based on their energy consumption patterns and energy management goals [35]. Further, the electrical grid is gaining intelligence. So-called smart grid initiatives seek to better handle demand spikes and more efficiently use electrical power. To make all this work, it requires data modeling, big data analytics, fore casting and simulation software.

This proposed project extends predictive analytics in the utility sector. With data management platform and massively parallel processing (MPP), utilities can integrate all their data streams to form a complete picture of every home and every business in their service territories. With the help of predictive analysis using big data platform, they can predict high bills weeks before they hit, and help customers get on track for lower ones. This prediction can help in developing core infrastructural technologies like Smart Grid, help consumer regulate and control their power needs and thus controlling their utilities bills without compromising on the quality of life by making trivial changes.

A. Objective

The objective of this project is to design an application to analyse and predict the energy consumption for household customers and provide meaningful insights to the users based on predictive analytics. It includes data modelling to design a user interactive model to extract meaningful analysis from the large database collected from users.

B. Motivation

The energy and utility sector is rich in data but the need of the hour is to quickly and accurately gain insights from that data. A customer no longer needs just a monthly bill in his mailbox but the accurate prediction of future bills and the recommendation to reduce his bill which indeed requires data modelling and predictive analytics.

C. Use of AWS Cloud

The user interface code of web-application and the analytic model written in R resides in a cloud (S3) in the form of an analytics program that uses the user given data to make a usage analysis and prediction. The entire data resides on the Amazon S3 cloud so that the data can be stored and retrieved from anywhere on the web. The large data is processed dynamically into meaningful data by using AWS EMR which is easy, fast and cost-effective. The model is an adaptive one, i.e. the computation and analysis on the data is done by the model created dynamically so that it includes all latest user-fed data. The use of cloud computing enables lower cost, easier deployment and higher scalability.

D. Overview and Workflow

The application is based on Big Data Analytics. The model is designed in R using R-Studio and further extended to user-interactive model using ShinyApp package. The large database is compressed to meaningful data using EMR. Finally, the entire application is deployed on AWS cloud.
The project was divided in 5 parts. First part involved model creation on R. Second was the design of user-interactive model on ShinyApp. Third was the data set selection and mapreduce code in python for Mapper and Reducer to reduce the big data into meaningful data. Fourth was reduction of large data set into model driven data using AWS EMR. Lastly part was running entire application on Cloud using AWS services.

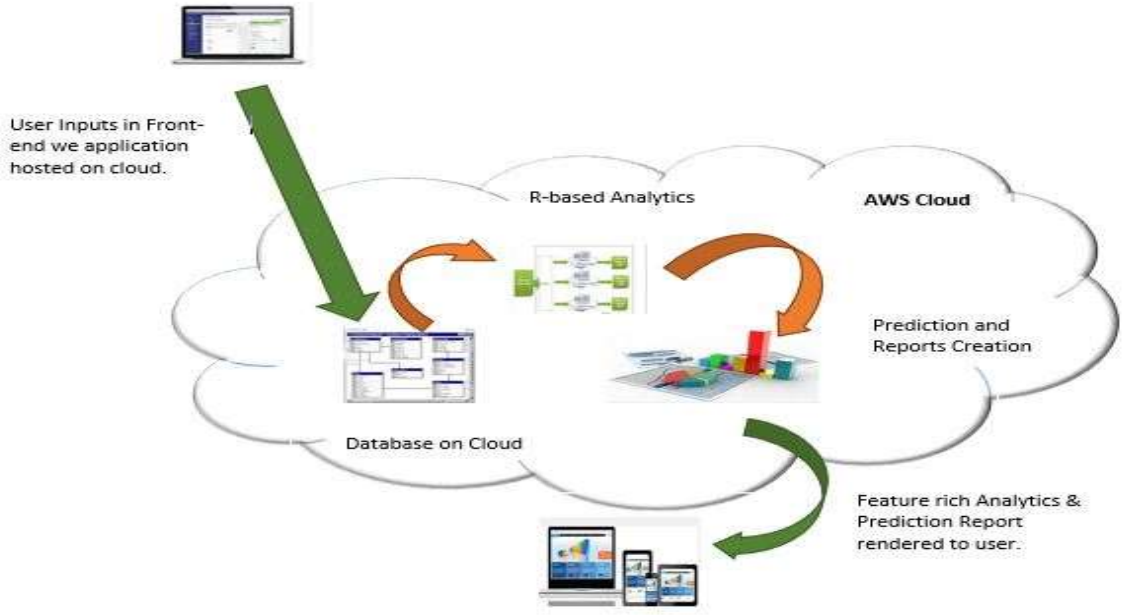

Figure 1: Basic Flow of the project

This report is divided into 8 sections. Section II and III describe data model design and data set selection. Section IV and Section V deal with Data Reduction by EMR and Final Web Application. Section VI presents Discussion and Analysis. Lastly, section VII and VIII concludes the report and deals with future work respectively.

II. DATA MODEL DESIGN

Before starting the model design, it is very important to understand the aim of the model and various statistical models and to choose the model best suited for the application. It is very clear that the model for the application should be able to tackle both- prediction and inference/analysis. The model has many variables and hence, the approach to be used will be parametric instead of non-parametric so that individual variables will contribute to the final result. Moreover, the application has responses which is the monthly bill. Hence, the model which can be used should fall in

supervised category. Also, the problem here is not the classification but the regression, i.e. to predict and infer the monthly summer and winter electricity bill.

To sum up the features of the model to predict and analysis summer and winter electricity bill:

1. Parametric
2. Supervised
3. Regression

There is a requirement for another model which falls under Classification category to divide the electricity bill of customers into three categories – High, Low, Medium. For this Bayesian Classifier, K-Nearest Neighbour or K-means clustering can be used.

A. Model Selection

For the prediction and analysis of household electricity bill, the variables are both categorical and numeral.

1. K-Nearest Neighbour: The variables are numeral too, hence it cannot be used. [2]
2. Linear Regression: Linear Regression can be used and since it is a supervised parametric method, it can be best suited for the application.
3. Bayesian Classifier: It can be used when the prediction of electricity bill falls into one of these categories – High, Low or Medium. As of now, the application deals with the prediction of electricity bill and can further extended to have this feature.
4. K-means Clustering: It is also a classification method and can be used for classifying the bill as High, Low or Medium.

Hence, for the application in the project, Linear Regression will suit the best. Also, since there are many variables for the model, the Linear regression is called as Multiple Linear Regression as the model is created by running multiple linear regression technique.

B. Modelling with Linear Regression in R

R is an integrated suite of software facilities for data manipulation, calculation and graphical display [1]. Among other things it has

1. an effective data handling and storage facility,
2. a suite of operators for calculations on arrays, in particular matrices,
3. a large, coherent, integrated collection of intermediate tools for data analysis,
4. graphical facilities for data analysis and display either directly at the computer or on hardcopy, and
5. a well developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.

R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly, and has been extended by a large collection of packages.

Here is a small part of code which applies Linear Regression model in R.

A small part of code for the model written in R

```
#Read the data
cloud_data=read.csv("cloud_data.csv")        #reads the data file

cloud_data$Street = as.factor(cloud_data$Street)        #data type change
cloud_data$Type = as.factor(cloud_data$Type)            #data type change
str(cloud_data)

### Split
set.seed(100)
```

```
#Selecting entire model as training data
train = sample(1:nrow(cloud_data), nrow(cloud_data))
training_data = cloud_data[train,]

### Apply Linear Regression
model_summer = lm(Summer ~.-Winter , data=training_data)
summary(model_summer)
model_winter = lm(Winter ~.-Summer, data=training_data)
summary(model_winter)
```

Function 'lm' is used to create Linear Regression Model.

C. User interactive Model Design

For the web application, the model should be user interactive in order to take the data as inputs from user and output the predictive results dynamically at run time. Hence, a user interactive model was design using ShinyApp. ShinyApp is available in RStudio as a package called Shiny. There are two code files are needed written in R: server.R and ui.R.

Server.R code is for the server, i.e. it captures the back end of the application including mathematical description of model, adaptation of the model using new user data, generating results and prediction. The following code reads the data file and create a scoll bar menu with list of all streets. This is the dynamic creation of input list.

A small part of code from **<server.R>**

```
Below function creates a scroll bar menu for selecting street
shinyServer(function(input, output) {
  output$streetcontrols <- renderUI({
            selectInput('street', 'Street:',
                        choices=levels(read.csv("cloud_data.csv")$Street),selectize = FALSE,
                       selected="Ellendale"
                      )              })
}
```

Ui.R is for user interface. It is similar to the HTML code. It comprises the design of user interface environment where user can input his data and can see and analysis the results generated. User interface was created to allow maximum convenience to users to understand and feed-in the inputs. The code creates sidebar panel for inputs and mainpanel to show outputs

A small part of code from **<ui.R>**

```
Shiny Library needs to be called
library(shiny)
Creating an input for selecting street in sidebar panel
shinyUI(
   fluidPage(                  #creates a page
    titlePanel("Real-time Prediction and Analysis of Energy Consumption"),
    sidebarLayout(            # for layout
      sidebarPanel(           # for sidebar buttons
        uiOutput("streetcontrols"),
       textInput("newstreet", "Enter the new street", value = "Enter text..."),
```

```
    .....
  ),  #Sidebar panel ends
  mainPanel(     #mainpanel for outputs
  .....
)))
```

III. DATA SET COLLECTION

Since the implemented application belongs to the family of predictive data analytics, the heart of all the modelling, training and testing set is 'Data'. The Big-Data that has been used in this application for modelling and training has been collected from the **California Government's Report on Energy Consumption by End Use** [3][4]
The predictive power analytics as implemented in this report takes into account several factors that impact the final billing for electricity and that include the following:

a. **Temperature Prediction**: This control variable decides a lot of the energy demand in the house as most of the electricity is used to condition the ambient temperature of the house and operating the refrigerator to store the perishable food. We incorporate this data from [3][4]
b. **Street Name** – This input buckets the data into discreet locations and can be used to create to groups based on the locations.
c. **Type of House**: This input is used to model the impact of the type on the final electricity bill. For example, a 5 bedroom bungalow will have higher average consumption over a two bedroom apartment.
d. **Number of Rooms**: This parameter is directly proportional to relative electricity bill.
e. **Number of Residents:** This control parameter is necessary to reveal the real active users of the electricity.
f. **Floor in the building:** Is the house top floor or not, this binary information controls the ambient temperature of the house which in turn impact how much temperature conditioning appliances are active.
g. **Weekly Occupancy of the house:** This control value characterizes the activity at the house and its live electricity consumption time.
h. **Number of high energy consumption appliances:** Increasing number of such devices raises their probability of operation and higher their number, more is the electricity consumption.

These parameters are input by the user to create and/or test our prediction model using Linear Regressions.
We collected our training data from an online survey from residents around the USC campus. This data was used to develop and train the initial model. Further average data was taken up from the **California Government's Energy Consumption report from [3][4]** For doing an **extensive testing** of our data we developed a PERL based data generator that generated data in the similar ballpark as our training data with a few outliers to make our model robust.

The pseudo-code for the PERL based data generator for extensive testing is:

Pseudo-code

```
for (i = 0 → i == 1000000)
{
  Random_sel_street(Ellendale, Menlo, 23, Jefferson….)
  sel_type(Bungalow, Townhouse, Apartment)
  sel_room_cnt (1-8)
    {
      Number_of_people = room_cnt*(rand(2))# Assuming max 2 people share a room
      Number of appliance = 2*room_cnt+rand(3) #Assuming Central AC or 1 AC/room and Refrigerator +
Electric Stove and a Heater
  }
  Is_top(true or false)
  Generate_bill (type, room_cnt, number_of_people, number of appliance, is_top)
```

```
{
   If(is floor == top)
    {
       Summer_Bill_factor = Type_factor*room_cnt*number_of_people*1.5 #Top floor is hotter
       Winter_Bill_factor =  Type_factor*room_cnt*number_of_people*0.9
    }
   If(is_floor == not_top)
    {
      Summer_Bill_factor = Type_factor*room_cnt*number_of_people*1
       Winter_Bill_factor =  Type_factor*room_cnt*number_of_people*1.2 #Lower floors are cool
    }
}
```

This data set generator generates a data set of 1000000 samples which are later fed to Amazon Web Server Elastic Map Reduce for reduction.

IV. DATA REDUCTION BY EMR

The dataset, which is 1000000 entries big with a tie complexity of O(n) takes a long time to process and thus there was a need of reducing this data using AWS EMR. The mapper and reducer functions were both written in Python and executed on Hadoop using the data_set.csv as the input and data_cloud.csv as the output.

A. Mapper

The mapping function maps the key-value pair for the original data-set. The keys are the control values for the house location and its value is the electricity bill the house gets.
The pseudo-code for the Mapper function is:

Mapper Pseudo Code

```
Block_id = 0;
For line in stdin
{
        Line = line.strip
        Inter_val, inter_key = line.split (' ' , 1)
  }
Sort.line(key);
For line in stdin
{
        If(inter_key == inter_key -1)
        {Next;}
        Else
        {Block_id ++;}
}
```

The efficiency measurement of sorting function used here takes its understanding from the [23].

B. Reducer

Once the sorted key-value pairs are available, the data is sent to the reducer which s again implemented in python to reduce the data set. The function of the reducer is to collect the sorted data and generate an average value for every distinct key. The reduced file is

data_cloud.csv and this is further used by the prediction analytics application to train and test the prediction models. The pseudo-code for reducer is:

Reducer Pseudo-code

```
Block_id = 0;
Num = 0; Avg_val = 0; Sum_bill =0;
for line in sys.stdin:
{
  line=line.strip(); #remove leading and trailing whitespaces
  input_key, input_value = line.split('\t',1)
  for I in range Block_id
        {
              Sum_bill += input_value;
              Num++
        }
Avg_val = Sum_bill/Num
```

After Reducer for the data-set the 1000000 entry data-set reduces to a smaller 10000 key-value pairs where values are the average values for the given key. This data is now given to the predictive model for testing. The performance of the Map-Reduce was tracked using AWS EMR monitoring using AWS Cloudwatch. Each of the graphs are further analyzed and described in the following section

C. EMR Performance Analysis

This EMR function utilizes the sorting algorithm and the averaging. The performance of Sorting on EMR clusters has been discussed extensively in [23]. For this project the EMR cluster that has been implemented is m3.xlarge with 1 master and 2 core nodes.

The overall performance with the used data set is plotted and explained as follows:

1. Execution Performance

Figure 2: Execution Performance for Application Completion (HDFS Utilization and Exec Time)

This plot indicates the execution time for the whole application. The steps in the 'Submitted' plot could be reasoned because of more blocks than total reducers implemented in 1 master 2-core architecture. The number of blocks were 1178. Some of the blocks worked longer to due to large collection of value that needed to average out.

2. Memory Consumption

Figure 3: Plot of Memory Consumption in the application execution

The memory consumption for the complete execution of the MapReduce function is as expected. The sudden rise indicates the extensive set of data that is processed in the MapReduce.

D. Relevance of Individual HiBench Programs

The five HiBench program that were run for this project are TeraSort, Bayesian Classifier, K-Means Clustering, join and aggregation. The current application needs sorting of data based on street for which TeraSort benchmark tests the performance of the cluster. The application can further extended to classify electricity bill into 3 categories, High, Low and Medium for which Bayesian Classifier and K-Means Clustering will be useful to data modelling. Also, the application in future can support database querying to search certain houses below a threshold bill level. Hence, for testing database querying, join and aggregation programs with Bayesian Classifier and K-means Clustering Benchmarks were run to evaluate cloud framework.

V. Final Web Application

A. Running R based web applications on AWS

R is an open source programming language and software environment designed for statistical computing, visualization and data. RStudio is an IDE for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. The server version of RStudio allows us to access R via a web browser hence avoiding the need to connect to the server via SSH.

Once the results have been analyzed, we can visualize them using Shiny, which is a great R package, that can be used to create interactive dashboards. Shiny provides a web application framework for R. It turns our analyses into interactive web applications. Shiny Server can deliver our R visualization to our customers via a web browser and execute R functions, including database queries, in the background.

B. Launch an amazon EC2 instance

- *Choosing an Amazon Machine Image for R*

Amazon Machine Image (AMI) contains all information required to start an instance. For example, an AMI defines which operating system is installed on your EC2 instance and which software is included. Therefore while launching an instance we have to choose an AMI.

In our project we have selected the Amazon Linux AMI, which is provided at no additional cost and has a stable version of R in the repository. This AMI is maintained by AWS and includes packages and configurations that provide seamless integration with AWS and other software.

- *Choosing an Instance Type for R*

  We have selected an EC2 instance type that matches the data size and processing that our analysis requires. By default, R runs only on one core node and, in many cases, requires a lot of memory.

  We have used t2.micro instance which is a general-purpose instance type used for programming and development.

- *Configuring instance details (user data)*

  While launching an instance in EC2, we can pass in user data that can be used to perform common automated configuration tasks and even run scripts for installation after the instance starts. In the EC2 launch wizard, we have added this in the Advanced Details pane of the Configure Instance Details step to install R, RStudio Server, the Shiny R package, and Shiny Server. This also adds a user and password that can be used for logging in later: [22]

Configuring instance details

```
#!/bin/bash
#install R
yum install -y R
#install RStudio-Server
wget https://download2.rstudio.org/rstudio-server-rhel-0.99.465-x86_64.rpm
yum install -y --nogpgcheck rstudio-server-rhel-0.99.465-x86_64.rpm
#install shiny and shiny-server
R -e "install.packages('shiny', repos='http://cran.rstudio.com/')"
wget https://download3.rstudio.org/centos5.9/x86_64/shiny-server-1.4.0.718-rh5-x86_64.rpm
yum install -y --nogpgcheck shiny-server-1.4.0.718-rh5-x86_64.rpm
#add user(s)
useradd username
echo username:password | chpasswd
```

- *Configuring instance details (IAM roles)*

  IAM roles allow your application on EC2—in this case, R—to make API requests securely or access AWS services from your instances without requiring you to manage the AWS security credentials.

  We have added an IAM role to our EC2 instance based on our requirement. For our R-based data science environment, we had to make sure that our EC2 instance had permission to read data from the S3 bucket that we created. You cannot read S3 files unless specifically given permission to do so. The policies added to our custom IAM role are:

  AmazonEC2FullAccess
  AmazonS3FullAccess
  ReadOnlyAccess
  PowerUserAccess

- *Configuring the Security Group*

  In the EC2 launch wizard, we define a security group, which acts as a virtual firewall that controls the traffic for one or more instances. For our R-based analysis environment, we have opened up port 8787 for RStudio Server and port 3838 for Shiny Server.

C. Load data into our R-based environment on AWS

Once the EC2 instance starts running, we can connect via web browser to RStudio Server and R using the public DNS of the EC2 instance and the port number configured for Rstudio. But before doing this we need to check if the R server is up and running which might require installing some Linux packages. This is done by connecting to the EC2 instance with SSH and running the following command [23]:

sudo yum install curl-devel

Once the underlying R server is set up, we can load data and make the R application.

D. Storing and retrieving data from Amazon S3

Amazon S3 is secure, durable, highly-scalable object storage. It is easy to use, with a simple web service interface to store and retrieve any amount of data from anywhere on the web.

Created a bucket to store the data on S3 for analysis, copied the data via the AWS command line interface to our EC2 instance, and read the data into R. By making our S3 data permission "Everyone", we can read the data directly into our EC2 instance.

E. Configure Shiny Server

There are some configuration changes that needs to be made to use Shiny Server. With these configuration changes, users will have the ability to manage and host their own Shiny applications. This will allow users to work directly with their applications rather than requiring an administrator to deploy and update applications on their behalf. To to this we need to connect to our EC2 instance and run the following commands:[22][24]

sudo /opt/shiny-server/bin/deploy-example user-dirs
mkdir ~/ShinyApps

Shiny Server is configured by a file stored at */etc/shiny-server/shiny-server.conf*. To use this, there is an executable in */opt/shiny-server/bin/deploy-example* that can copy the example configuration file associated with this Quick Start to that location and install a directory of example applications. This configuration enables all users on the system to host their own applications by creating a ShinyApps directory in their home directory.

Now we can copy our entire application into a directory named Project_Team7 under ShinyApps. By default, Shiny Server listens on port 3838, so our application will be available at the following URL:

http://<EC2 public DNS>:3838/ec2-user/project_team7

F. Final Web Application

Figure 4 shows the final web application which does real-time prediction and analysis of energy consumption. The frontend is made up of left sidebar and the main panel at the centre.

The left sidebar consists of user inputs like street, whether it's a new street, house type, number of occupants, past usage per month, etc.

The main panel consists of the four output tabs – Prediction, Plot, Summary and Suggestion. The prediction tab displays the expected average summer and winter predicted energy bills in dollars per month. The plot tab displays the graphical representation of this data. The summary tab explains the summary of this search. The suggestion tab gives general suggestions on how to reduce the energy bill.

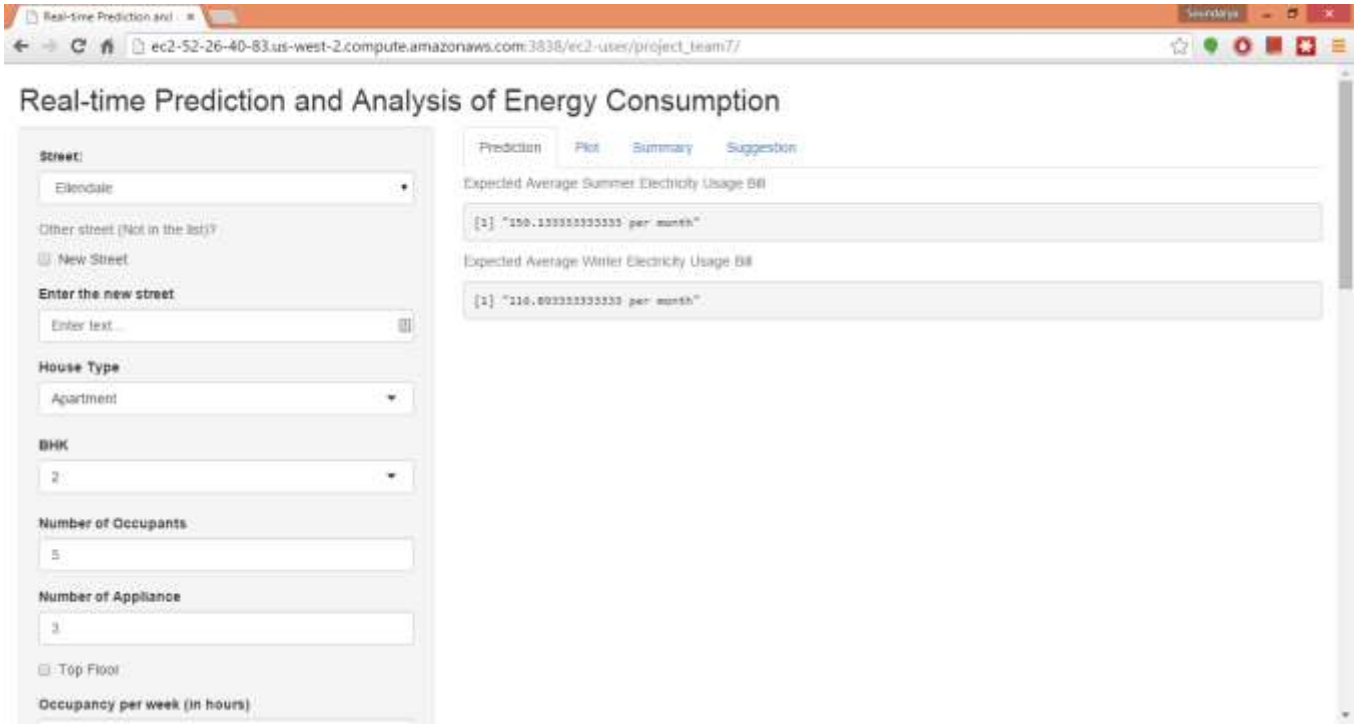

Figure 4: Final Web Application

G. Output Results

Figure 5 shows the prediction tab, which displays the expected average summer and winter predicted energy bill in dollars per month.

Figure 5: Prediction Results

Figure 6 shows the plot tab, which displays the graphical representation of the past and predicted energy usage data for the following year.

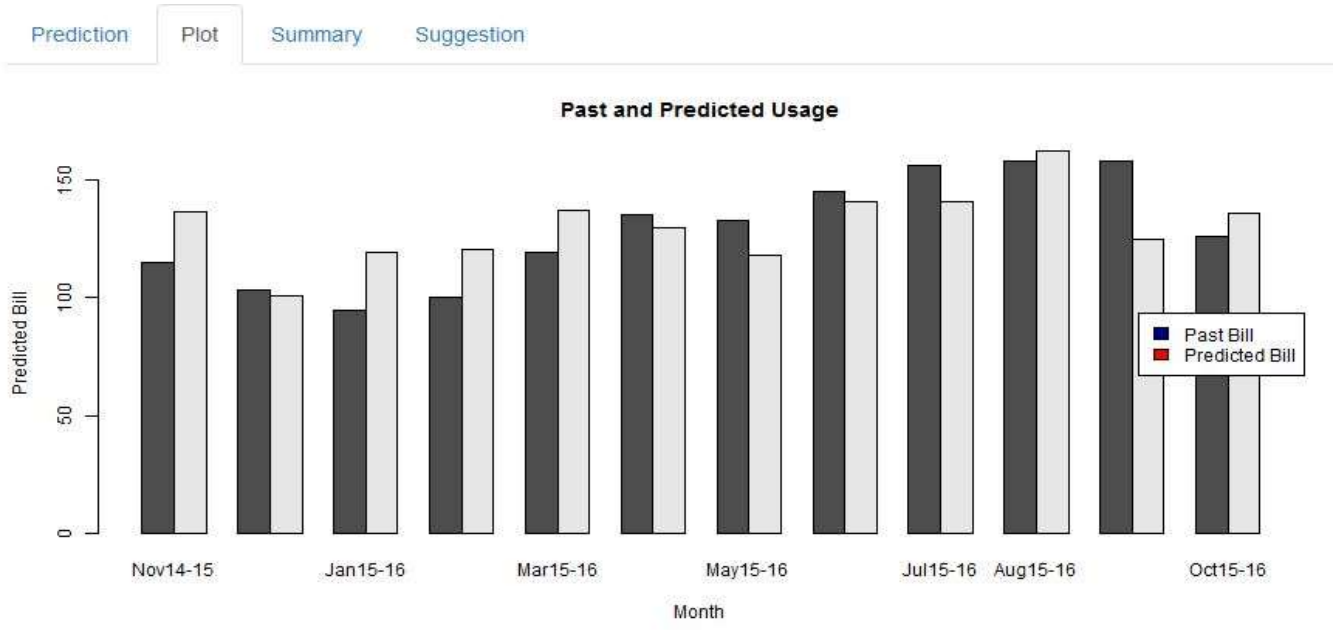

Figure 6: Analysis and Prediction Results

VI. DISCUSSION AND ANALYSIS

Considering the importance of power saving and prediction of power usage, this application provides smart prediction of power usage using machine learning algorithm. The application needs inputs from the user like the street name, number of appliances , etc. and runs the machine learning algorithm on the input variables, using the past usage information to provide smart prediction for the upcoming months. Users can add new streets which will serve as data for future predictions. It also provides suggestions like what can be improved to reduce the power usage.

For the data we took training data sets from users around the USC campus. Big-data was taken from the California Government's Report on Energy Consumption by End Use. PERL based generator was used to generate data in the range of the collected data used for training the model. We used EMR to compress the big data and the resulting data was used as an input to the application. The application then used machine learning algorithm written in R to generate the prediction. Shiny server was used to run this application as well as for the user interface. The web application gives the user a graphical interface and allows the user to feed in the data and see the results in the form of numbers and graphs.

Three services provided by AWS were used for this web application. We used S3 service provided by the AWS for storing the data, the .R files and the input given by the user. EMR service was used for processing the large input dataset. We used EC2 instance for running the final application. Server.R and ui.R were the two files used which served for the main algorithm of the application and the user interface respectively. Weather forecast information was used for the prediction and was taken from [3][4]

The primary service that the application provides is an estimated power usage bill for the future months. Along with this it also provides ways of reducing the power usage by giving smart suggestions based on the living conditions. It also helps the user to get a rough estimate of the average power bill on a particular street so that he can choose the apartment not just based on the rent but also on the power usage. This application provides the often neglected wise suggestions that can lead to less power usage. It also encourages using alternative ways of producing electricity when needed. Use of solar panels is highly encouraged in areas with good sunlight for most part of the year. The application will also help in developing Smart Grids by producing post analytics.

The web application gives the expected average summer bill and the expected average winter bill in dollars per month in prediction tab. The plot tab shows the graphical representation of this. The suggestion tab gives general suggestions on how to reduce the energy bill.

VII. CONCLUSION

This application serves as a smart means of predicting the future power usage bills. Scale out solutions have been used in the model. Linear regression has been used the develop the machine learning algorithm used for the web application. AWS services like S3, EMR and EC2 were used for storing data, for using MapReduce technique and for computing services. The Shiny server provided the graphical user interface to show the application written in R for the user.

VIII. FUTURE WORK

The application is now developed for residential sector but can be extended to commercial sector in the future. We have used scale out technique here but we can also extend it to scale up technique for future use. As the database gets bigger, database related queries like join and aggregation can be used [24][25]. The application can compare the power usage by alternative sources of energy in the same region and give a comparative study of whether the alternative sources will give a better performance in the long term or not. It can give the estimated future usage in the areas of a big region and can make changes in power distribution accordingly. This can ensure efficient use of power in any region. Electronic power conditioning and control of the production and distribution of electricity are important aspects of the smart grid.

REFERENCES

[1] W. N. Venables, D. M. Smith and the R Core Team. "An Introduction to R" Notes on R: A Programming Environment for Data Analysis and Graphics Version 3.2.2 (2015-08-14)
[2] Gareth James Daniela Witten Trevor Hastie and Robert Tibshirani. "An Introduction to Statistical Learning with Applications in R" Springer Texts in Statistics
[3] Webpage, "http://www.weather.com/weather/monthly/l/90007:4:US" accessed on Oct 14- 17 Nov 2015
[4] Webpage, "http://www.usclimatedata.com/climate/los-angeles/california/united-states/usca1339" accessed from Oct 14-17 Nov 2015
[5] Webpage, http://shiny.rstudio.com/gallery/
[6] Webpage, http://shiny.rstudio.com/reference/shiny/latest/sidebarLayout.html
[7] Webpage, http://stackoverflow.com/questions/20886800/setting-shiny-selectinput-from-data-frame
[8] Webpage, https://gist.github.com/psychemedia/9690079
[9] Webpage, http://shiny.rstudio.com/tutorial/lesson7/
[10] Webpage, http://stackoverflow.com/questions/30001211/filter-data-frame-for-use-in-multiple-renderplot-functions
[11]Webpage,http://stackoverflow.com/questions/16848352/r-and-shiny-pass-inputs-from-sliders-to-reactive-function-to-compute-output
[12] Webpage, http://stackoverflow.com/questions/19706320/create-a-data-frame-using-text-input-in-shiny
[13] Webpage, http://shiny.rstudio.com/articles/datatables.html
[14] Webpage, http://shiny.rstudio.com/reference/shiny/latest/renderDataTable.html
[15] Webpage, http://www.statmethods.net/graphs/bar.html
[16] Webpage, http://www.ats.ucla.edu/stat/r/faq/barplotplus.htm
[17] Webpage, http://www.statmethods.net/graphs/bar.html
[18] Webpage, https://stat.ethz.ch/R-manual/R-patched/library/graphics/html/barplot.html
[19] Webpage, http://www.inside-r.org/r-doc/graphics/barplot
[20] Webpage, http://rprogramming.net/write-csv-in-r/
[21] Webpage, https://bioconductor.org/help/course-materials/2015/CSAMA2015/lab/shiny.html
[22] Webpage, http://blogs.aws.amazon.com/bigdata/post/Tx3IJSB6BMHWZE5/Running-R-on-AWS
[23]Webpage, http://blogs.aws.amazon.com/bigdata/post/Tx37RSKRFDQNTSL/-Statistical-span-class-matches-Analysis-span-with-Open-Source-R-and-RStudio-on
[24] Webpage, http://rstudio.github.io/shiny-server/latest/#quick-start
[23] Aayush Pathak, "Evaluation and Characterization of AWS EMR Cluster using TeraSort HiBench Benchmark", USC Nov 2015
[24] Soundarya Singh, "Data analysis of Hibench benchmark Using varied data sets", USC Nov 2015
[25] Yesha Shah, "Analysis of Hi-Bench benchmark on AWS", USC Nov 2015
[26] Nikita Ahuja, "Evaluation of AWS EMR Cluster using Bayesian Classifier HiBench Benchmark", USC Nov 2015
[27] Ishan Patiyat, "Cloud Performance Evaluation by K-means Clustering HiBench Program", USC Nov 2015
[28] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," USENIX OSDI, December, 2004.

[29] Hwang, Shi et al, “Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies,” IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS
[30] http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasorttestdfsio-nnbench-mrbench/
[31] http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/examples/terasort/package-summary.html
[32] http://www.internetlivestats.com/google-search-statistics/
[33] Gene M Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” AFIPS spring joint computer conference 1967
[34] Webpage, http://www-935.ibm.com/services/multimedia/Managing_big_data_for_smart_grids_and_smart_meters.pdf
[35] Webpage, http://spotfire.tibco.com/blog/?p=22210