**Case Study 2 : Energy Forecasting for Boston City (Contd)**

**INFO 7390 : Advances in Data Sciences/Architecture**

**Team 7**

**Chandni Sharma**

**Hari Panjwani**

**Sanath Shetty**

**Table Of Contents**

1. Problem Statement 3
2. Data Wrangling & Cleansing 4
3. Prediction 37
4. Classification 40
5. Forecast 52
6. References 65
7. **Problem Statement**

The aim of this project is to build a predictive model to forecast the energy consumption for the city of Boston. The raw data available currently has a record of power usage for every 15 minute interval in a day.

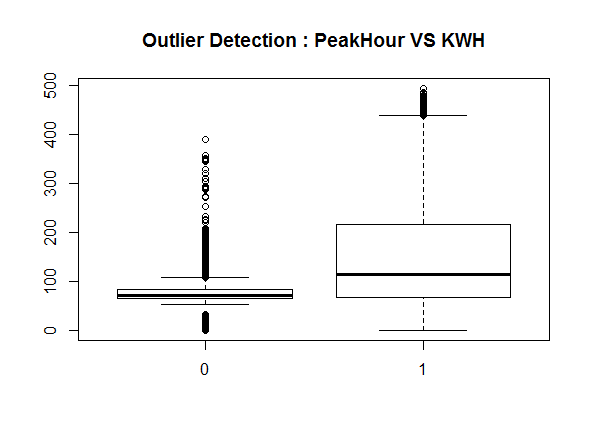
* 1. **Goal:**
* Analyze the existing energy consumption data for the city of Boston and develop an effective approach that can be used to monitor, plan and predict the power consumption for the future.
* Show trends in energy consumption based on various factors that are directly proportional to the power usage
  1. **Applications:**
* Recommend which account number or facility needs to be monitored and educate the citizens with suggestions for optimal usage of energy
* Make a note of consumption trends with respect to temperature, day of the week, month or even a particular day
* Recognizing a pattern in the usage to identify peak hour for daily energy consumption

1. **Data Wrangling & Cleansing**

**Approach 1 : Remove all the zero entities and use only the non zero entities for KWH to build a regression model.**

* 1. **Non- zero dataset :**

Reading the dataset from the source and removing all the zero values from the dataset. But before removing the zero values from the dataset, we decided to first transform and change the given dataset by using the transform and melt function from the reshape library. Further we split the date into month, year, date, dayOfWeek and WeekDay. Next thing was to aggregate the hours



to hourly since the temperature data that we acquire from WeatherData is hourly. We re- cleansed the data by removing the unwanted columns like Channel, Units, Variables and values. Finally we split the data into two parts the non-zero dataset and the dataset containing zero values. Before starting to work on the non zero dataset, we decided to test for outliers in the given dataset. We then split the non zero dataset into two one having observations for Peakhour = 0 and the other for Peakhour = 1.

Handling the outlier: We saw that there were around 26 outliers for value of KWH for Peakhour = 0. We replaced the outliers by first segregating the outliers from the non zero dataset and taking a mean of the KWH for peakhour = 0 and then replaced the outliers by the mean.

CodeSnippet:

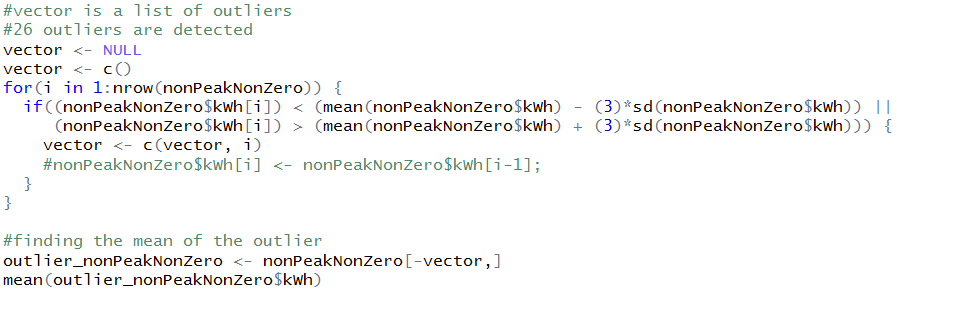


Fig: Code Snippet for handling outliers

On handling the outliers we then bided the two datasets for peakhour = 0 and peakhour = 1 into a consolidated dataset called energy dataset.

**Temperature Dataset**

On pulling the data from package of **WeatherData** in R, it is observed that there are multiple observations for a particular hour. We handled those observations by taking mean for that hour and then replacing the multiple data for that hour with the mean calculated.

For eg : For every temperature shift in an hour an entry is being made for the hour.

3.15 🡪 22F

3.35 🡪 21F  
3.55 🡪 23F

In this case the algorithm that we have written would take the mean of the temperatures and associate it to the corresponding R (3rd hour 🡪 22F)

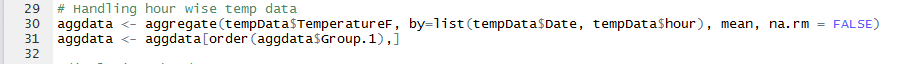
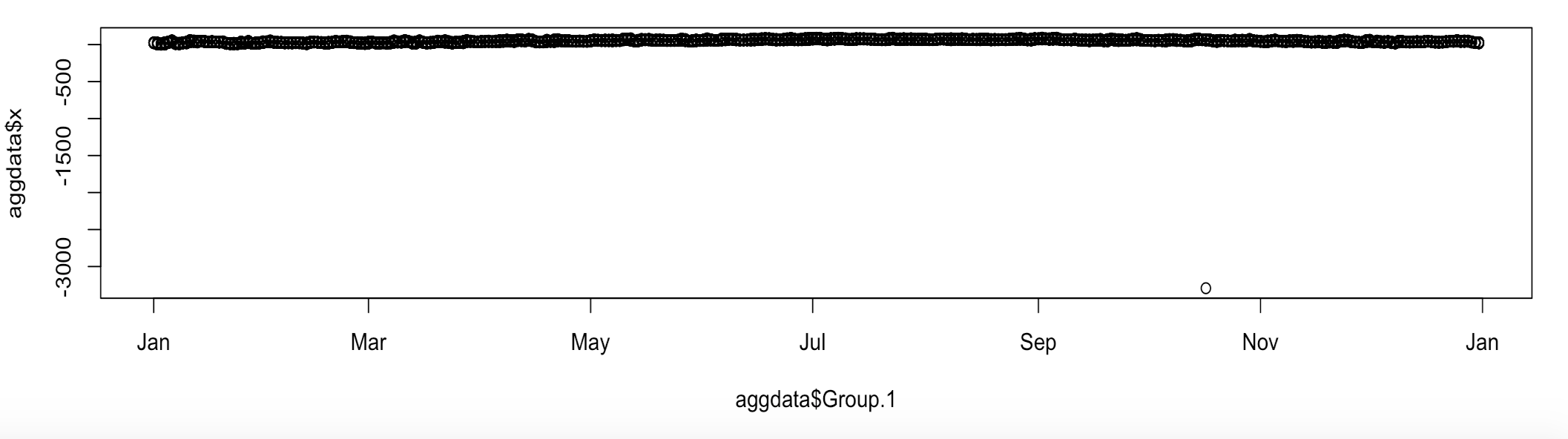


Fig : Code Snippet for handling multiple data for temp in an hour

We then proceeded with taking the summary and for the dataset and saw that there was a record for the temperature showing -324F which in this case was an obvious outlier. So we plotted the graph to check if there are more outliers.

**OutLiner Graph**

****

**Handling the outliners:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

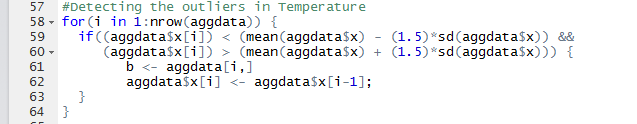


Fig : Code Snippet for Outlier function in Temperature dataset

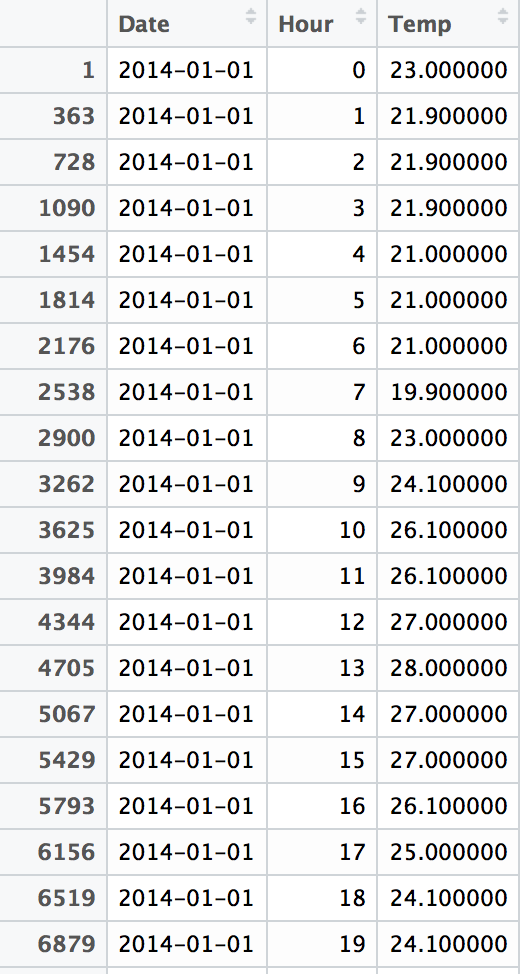
****

Fig: Final Temperature Dataset after cleaning

**Merging the two Datasets (Energy + Temperature 🡪 Consolidated )**

We finally merged the two dataset by using the dplyr package from R and applied the left outer join to achieve this. After this we took the summary for the merged dataset and it was found that there are multiple fields having “NA’s” and “0’s”

**Handling the NA’s in the merged dataset**

We used the zoo package to handle the NA’s in the consolidated dataset by using the function na.locf.

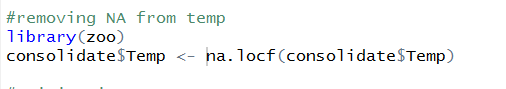


Fig : Code Snippet for handing NA’s & 0 values

**Testing for Outliners and replacing them:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

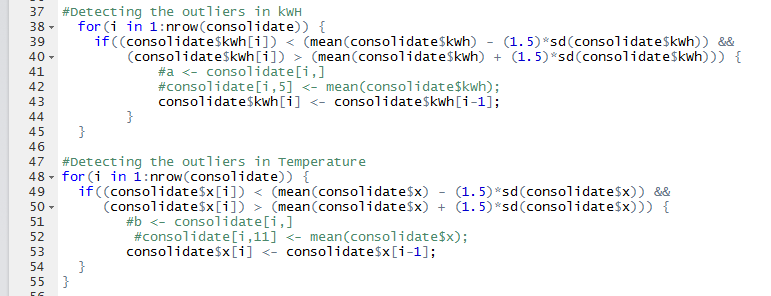


Fig : Code Snippet for handing outlier values

**Snapshot of Final Merged Dataset 🡪 Consolidated**

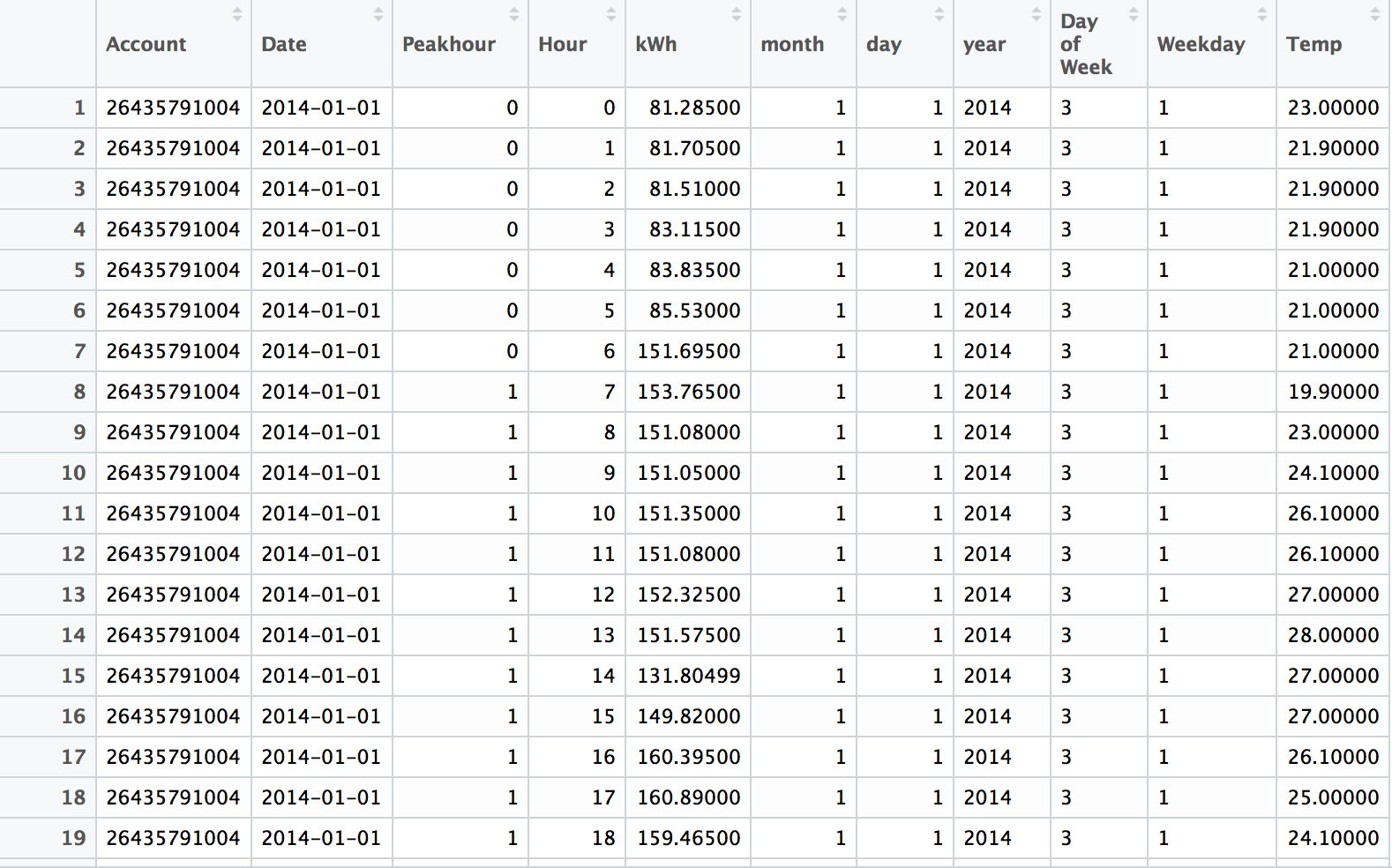
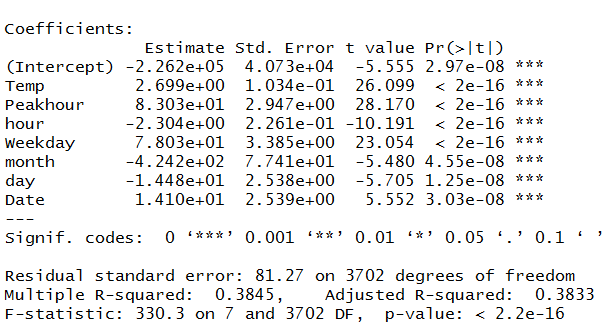
On 

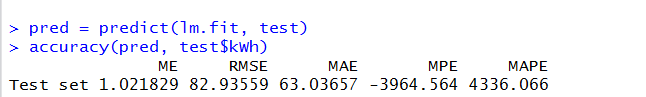
Fig : Snapshot of the final consolidated dataset.

**Observation for Approach 1:**

On evaluation of the performance metrics for the first approach we have the adjusted R square value as 0.38



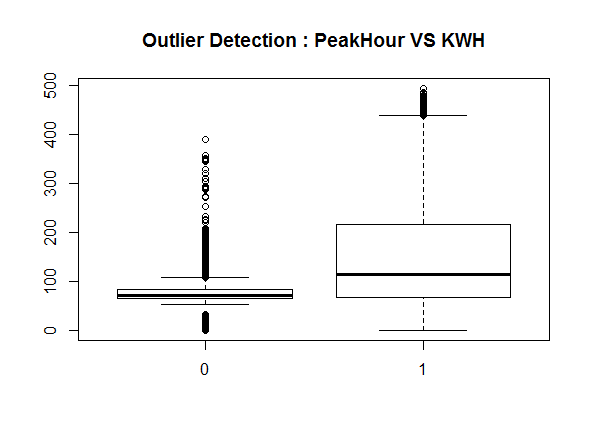
So for the model we have a forecast rate of 7.1. MAPE, RMS and MAE value as 4336, 82.93 and 63.04 respectively.



**Approach 2 : Replace all the zero entities with non zero data Use the model generated from the approach 1 topredict values for non zero KWH**

* 1. **Filled Dataset:**

Reading the dataset from the source and removing all the zero values from the dataset. But before removing the zero values from the dataset, we decided to first transform and change the given dataset by using the transform and melt function from the reshape library. Further we split the date into month, year, date, dayOfWeek and WeekDay. Next thing was to aggregate the hours



to hourly since the temperature data that we acquire from WeatherData is hourly. We re- cleansed the data by removing the unwanted columns like Channel, Units, Variables and values. Finally we split the data into two parts the non-zero dataset and the dataset containing zero values. Before starting to work on the non zero dataset, we decided to test for outliers in the given dataset. We then split the non zero dataset into two one having observations for Peakhour = 0 and the other for Peakhour = 1.

Handling the outlier: We saw that there were around 26 outliers for value of KWH for Peakhour = 0. We replaced the outliers by first segregating the outliers from the non zero dataset and taking a mean of the KWH for peakhour = 0 and then replaced the outliers by the mean.

CodeSnippet:

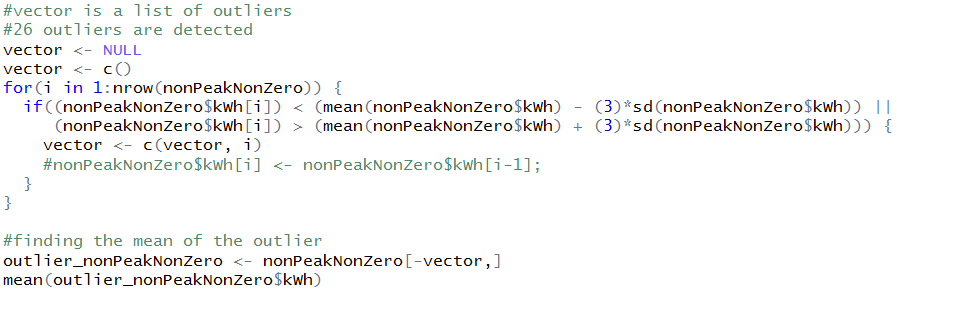


Fig: Code Snippet for handling outliers

On handling the outliers we then bided the two datasets for peakhour = 0 and peakhour = 1 into a consolidated dataset called energy dataset.

Utilizing the model from approach 1 to predict KWh

The main objective or aim of approach 2 is to build a model on the dataset containing non-zero kWh values of the consolidated and make a forecast for kWh in the dataset containing zero value for kWh.

Here we first spit the consolidated data (nonZerokWhdataset) in the ratio 0.75 : 0.25 i.e. into training and test dataset.We built a linear model on it. Below is the accuracy for the predicted model:

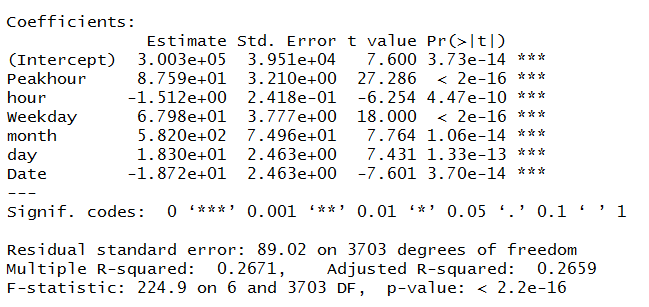


Fig: Adjusted R Square value



Fig : Performance Metrics

We then made a forecast for kWh values present in the zeroKwHDataset using the above model. We finally merged the two datasets into a consolidated energy dataset. However the forecast result was not so accurate. We were getting negative values for kWh. On further analysis we concluded that the negative values for Kwh are due to the fact that it is dependent on Date and hour which have negative intercepts in our model. The kwh values are dependent on date and hour because of the nature of the observations and our model is not mature enough to build any other relationship where peakhour is zero.

We tried handling the negative values for kWh and we concluded that the kWh are dependent on month, non peakhour and weekday.

So we finally included the forecast data which had negative values and merged with the final dataset.

**Temperature Dataset**

On pulling the data from package of **WeatherData** in R, it is observed that there are multiple observations for a particular hour. We handled those observations by taking mean for that hour and then replacing the multiple data for that hour with the mean calculated.

For eg : For every temperature shift in an hour an entry is being made for the hour.

3.15 🡪 22F

3.35 🡪 21F  
3.55 🡪 23F

In this case the algorithm that we have written would take the mean of the temperatures and associate it to the corresponding R (3rd hour 🡪 22F)

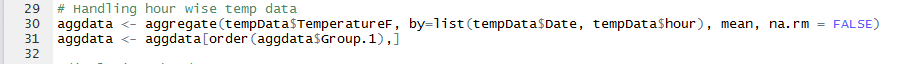
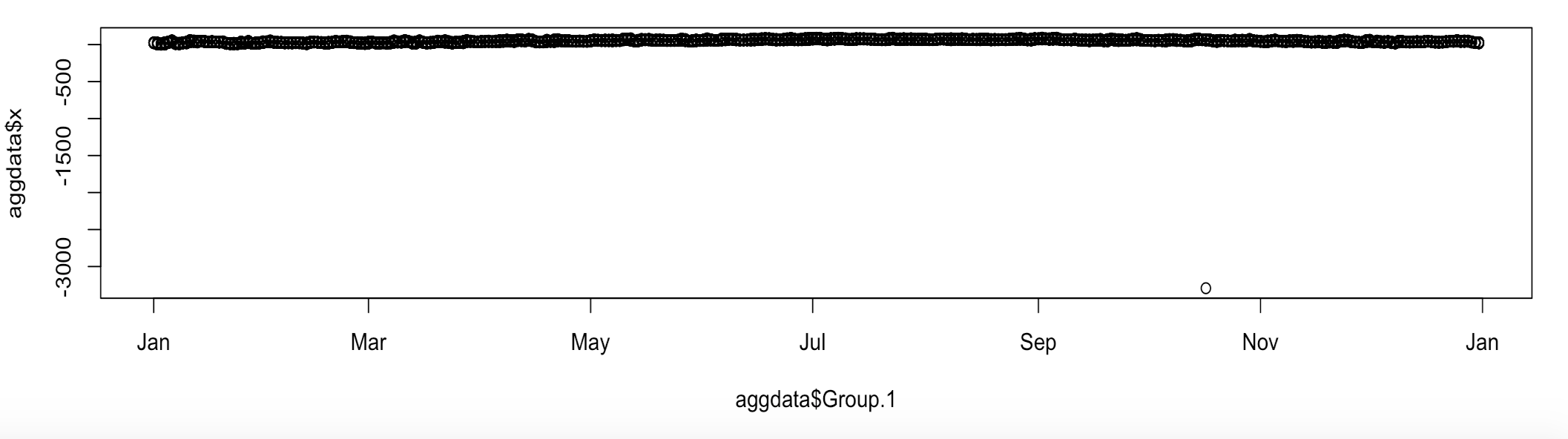


Fig : Code Snippet for handling multiple data for temp in an hour

We then proceeded with taking the summary and for the dataset and saw that there was a record for the temperature showing -324F which in this case was an obvious outlier. So we plotted the graph to check if there are more outliers.

**OutLiner Graph**

****

**Handling the outliners:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

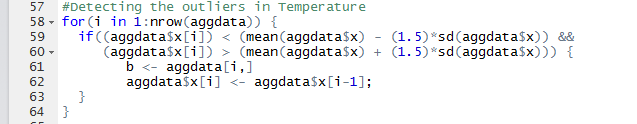


Fig : Code Snippet for Outlier function in Temperature dataset

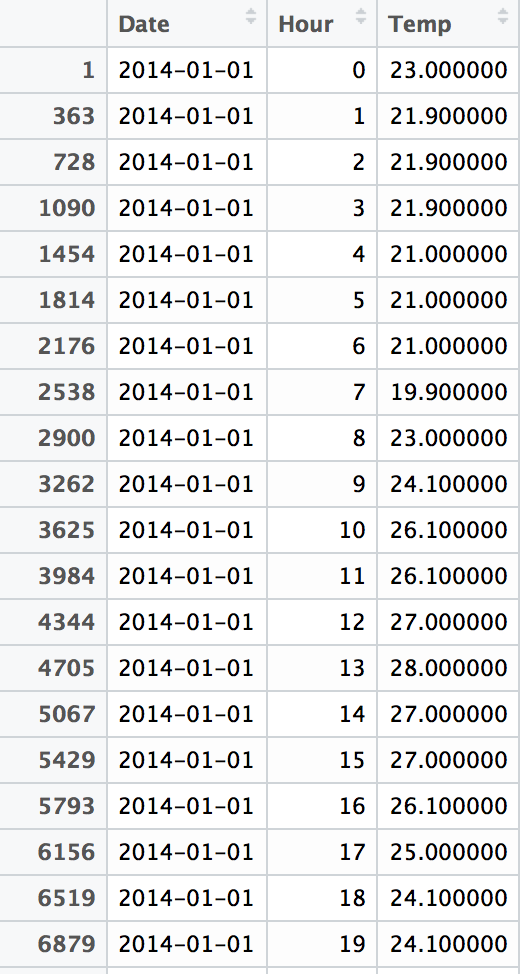
****

Fig: Final Temperature Dataset after cleaning

**Merging the two Datasets (Energy + Temperature 🡪 Consolidated )**

We finally merged the two dataset by using the dplyr package from R and applied the left outer join to achieve this. After this we took the summary for the merged dataset and it was found that there are multiple fields having “NA’s” and “0’s”

**Handling the NA’s in the merged dataset**

We used the zoo package to handle the NA’s in the consolidated dataset by using the function na.locf.

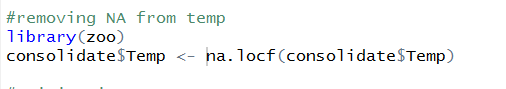


Fig : Code Snippet for handing NA’s & 0 values

**Testing for Outliners and replacing them:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

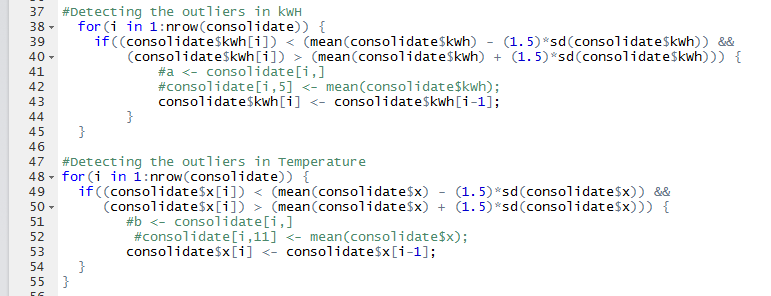


Fig : Code Snippet for handing outlier values

**Snapshot of Final Merged Dataset 🡪 Consolidated**

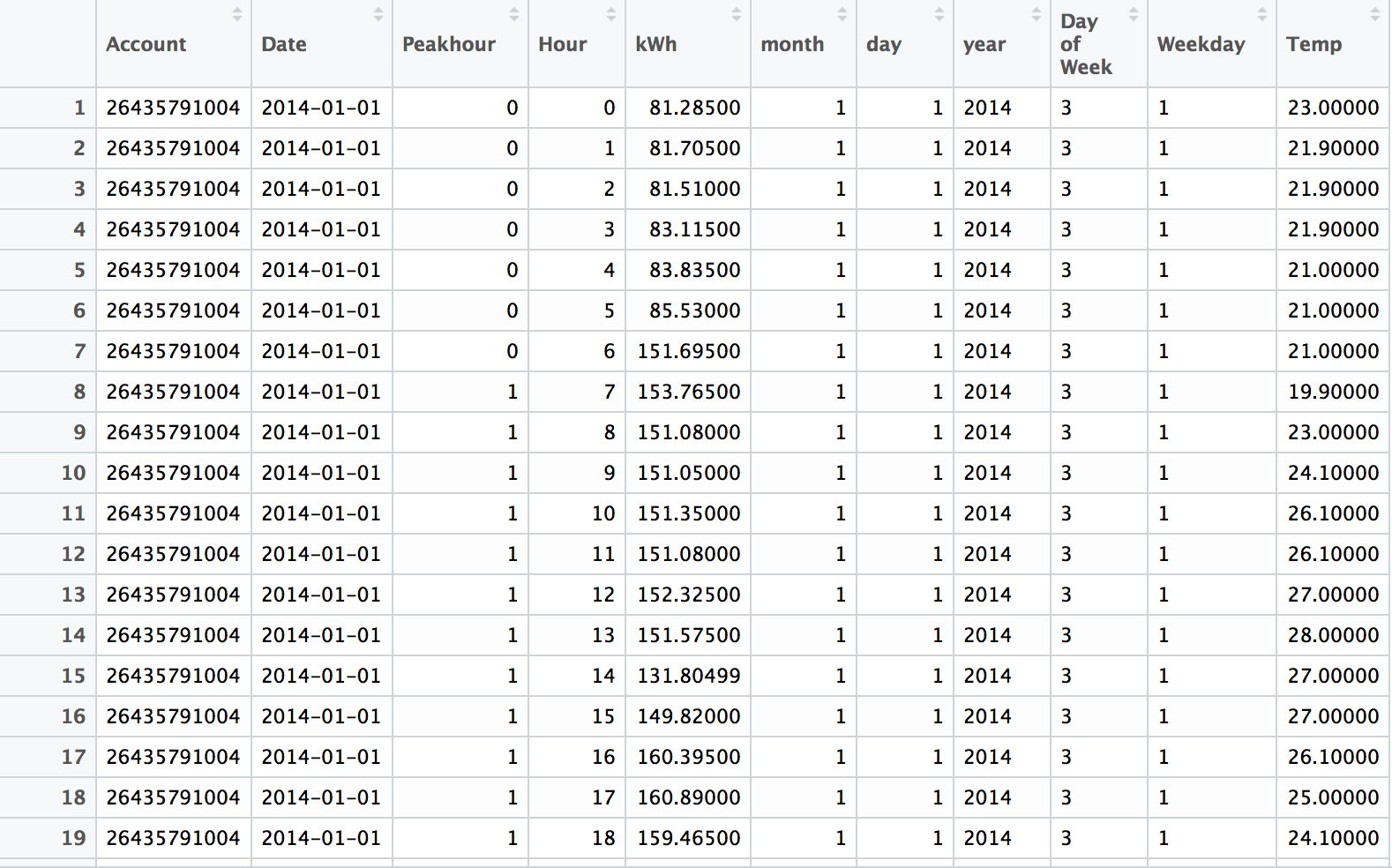
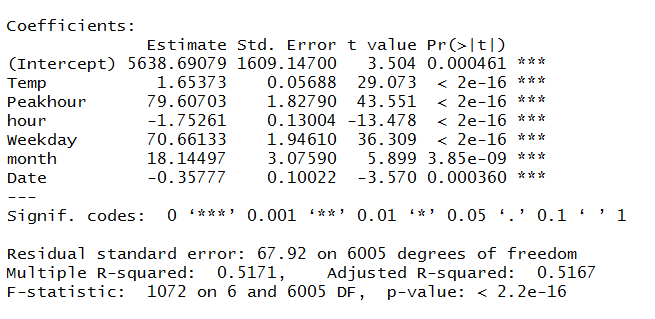


Fig : Snapshot of the final consolidated dataset.

**Observation for Approach 2:**

On evaluation of the performance metrics for the second approach we have the adjusted R square value as 0.52



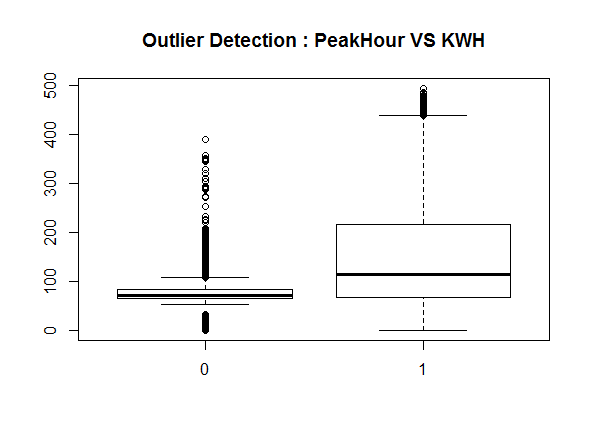
So for the model we have MAPE, RMS and MAE value as 2470, 63.26 and 42.88 respectively.



**Approach 3 : Replace all the zero entities for kWh using the zoo package and build a regression model.**

* 1. **Filled dataset :**

Reading the dataset from the source and removing all the zero values from the dataset. But before removing the zero values from the dataset, we decided to first transform and change the given dataset by using the transform and melt function from the reshape library. Further we split the date into month, year, date, dayOfWeek and WeekDay. Next thing was to aggregate the hours



to hourly since the temperature data that we acquire from WeatherData is hourly. We re- cleansed the data by removing the unwanted columns like Channel, Units, Variables and values. Finally we split the data into two parts the non-zero dataset and the dataset containing zero values. Before starting to work on the non zero dataset, we decided to test for outliers in the given dataset. We then split the non zero dataset into two one having observations for Peakhour = 0 and the other for Peakhour = 1.

Handling the outlier: We saw that there were around 26 outliers for value of KWH for Peakhour = 0. We replaced the outliers by first segregating the outliers from the non zero dataset and taking a mean of the KWH for peakhour = 0 and then replaced the outliers by the mean.

CodeSnippet:

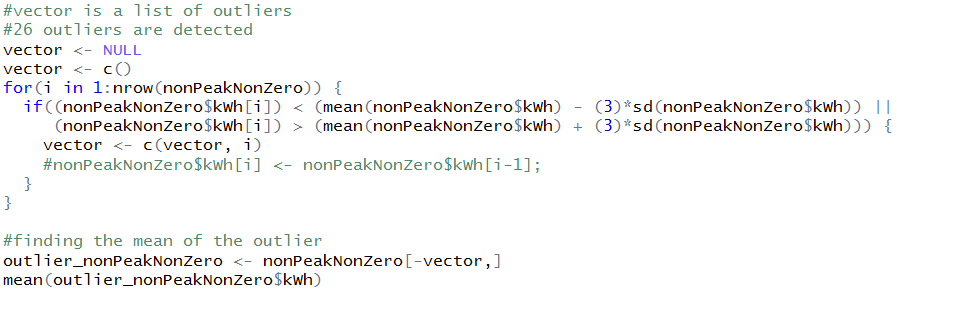


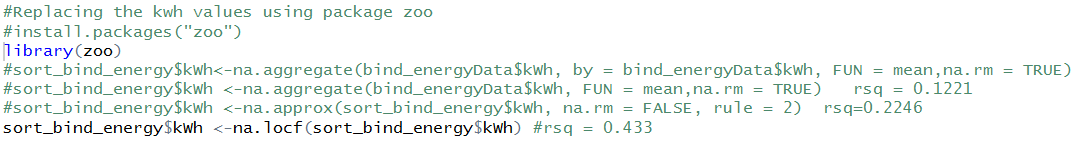
Fig: Code Snippet for handling outliers

On handling the outliers we then bided the two datasets for peakhour = 0 and peakhour = 1 into a consolidated dataset called energy dataset.

Then we split the dataset into two, one having non zero values of kWh and one having zero values of kWh. We replaced all the zero values of Kwh in the second dataset by NA’s



We used the zoo package to replace the NA’s here:



**Temperature Dataset**

On pulling the data from package of **WeatherData** in R, it is observed that there are multiple observations for a particular hour. We handled those observations by taking mean for that hour and then replacing the multiple data for that hour with the mean calculated.

For eg : For every temperature shift in an hour an entry is being made for the hour.

3.15 🡪 22F

3.35 🡪 21F  
3.55 🡪 23F

In this case the algorithm that we have written would take the mean of the temperatures and associate it to the corresponding R (3rd hour 🡪 22F)

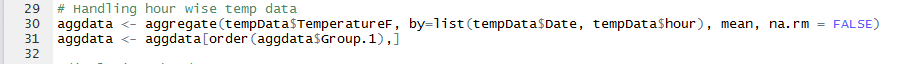
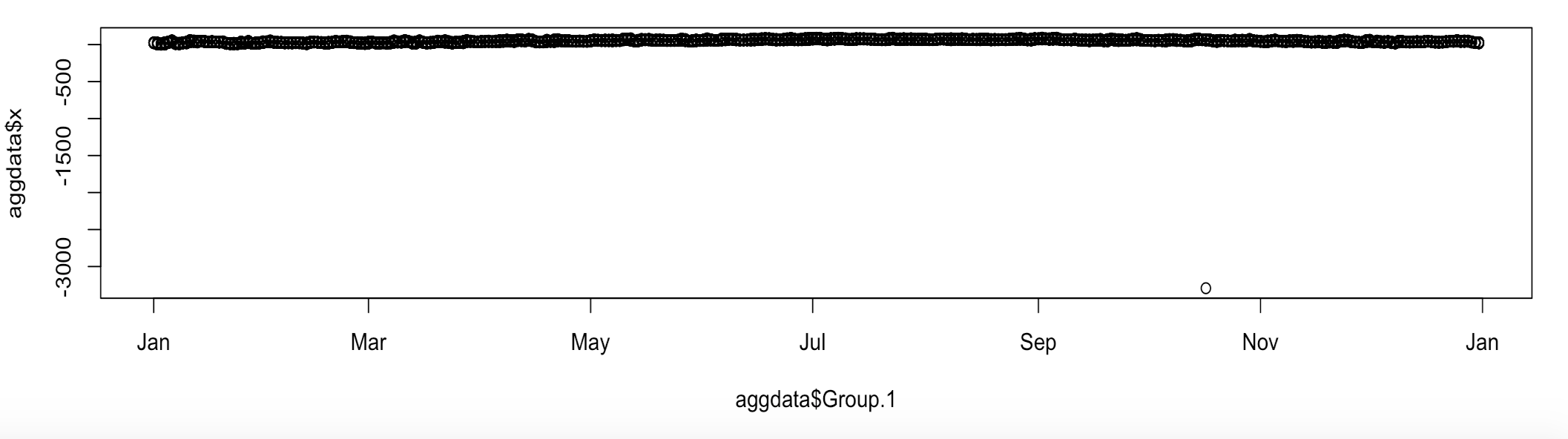


Fig : Code Snippet for handling multiple data for temp in an hour

We then proceeded with taking the summary and for the dataset and saw that there was a record for the temperature showing -324F which in this case was an obvious outlier. So we plotted the graph to check if there are more outliers.

**OutLiner Graph**

****

**Handling the outliners:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

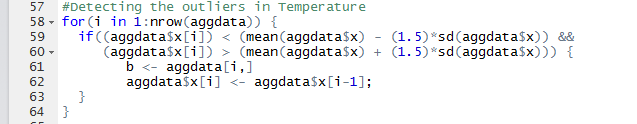


Fig : Code Snippet for Outlier function in Temperature dataset

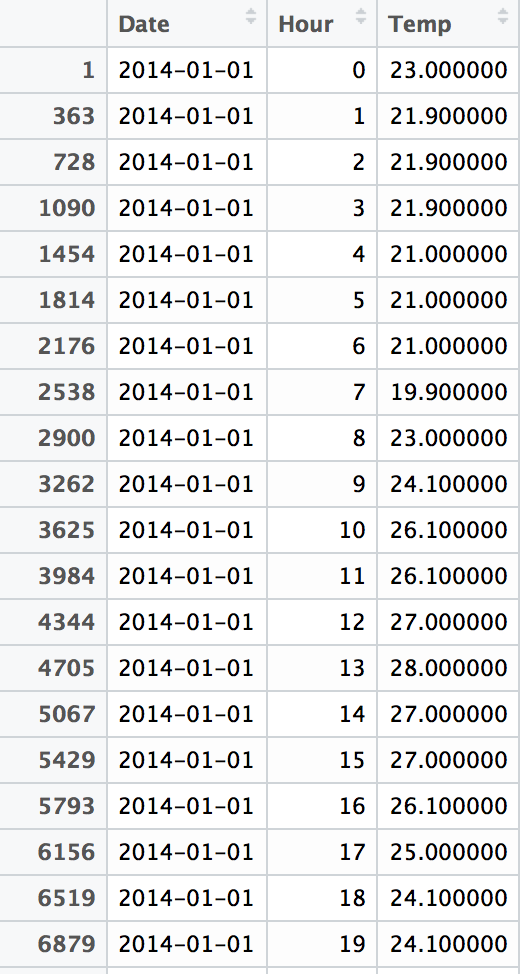
****

Fig: Final Temperature Dataset after cleaning

**Merging the two Datasets (Energy + Temperature 🡪 Consolidated )**

We finally merged the two dataset by using the dplyr package from R and applied the left outer join to achieve this. After this we took the summary for the merged dataset and it was found that there are multiple fields having “NA’s” and “0’s”

**Handling the NA’s in the merged dataset**

We used the zoo package to handle the NA’s in the consolidated dataset by using the function na.locf.

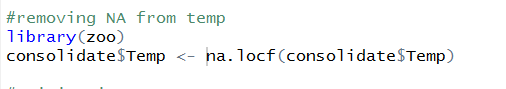


Fig : Code Snippet for handing NA’s & 0 values

**Testing for Outliners and replacing them:**

In order to detect the outliners we wrote down a function and replace the outlier if any by the previous value in the column

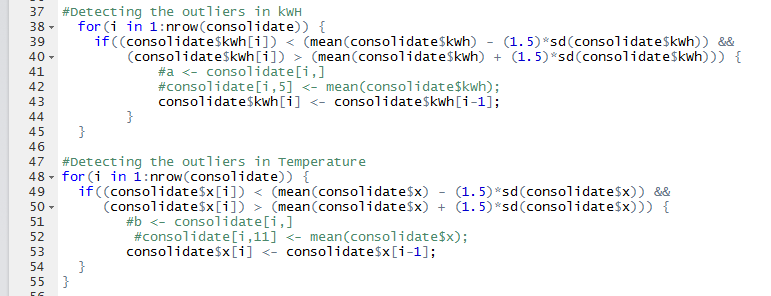


Fig : Code Snippet for handing outlier values

**Snapshot of Final Merged Dataset 🡪 Consolidated**

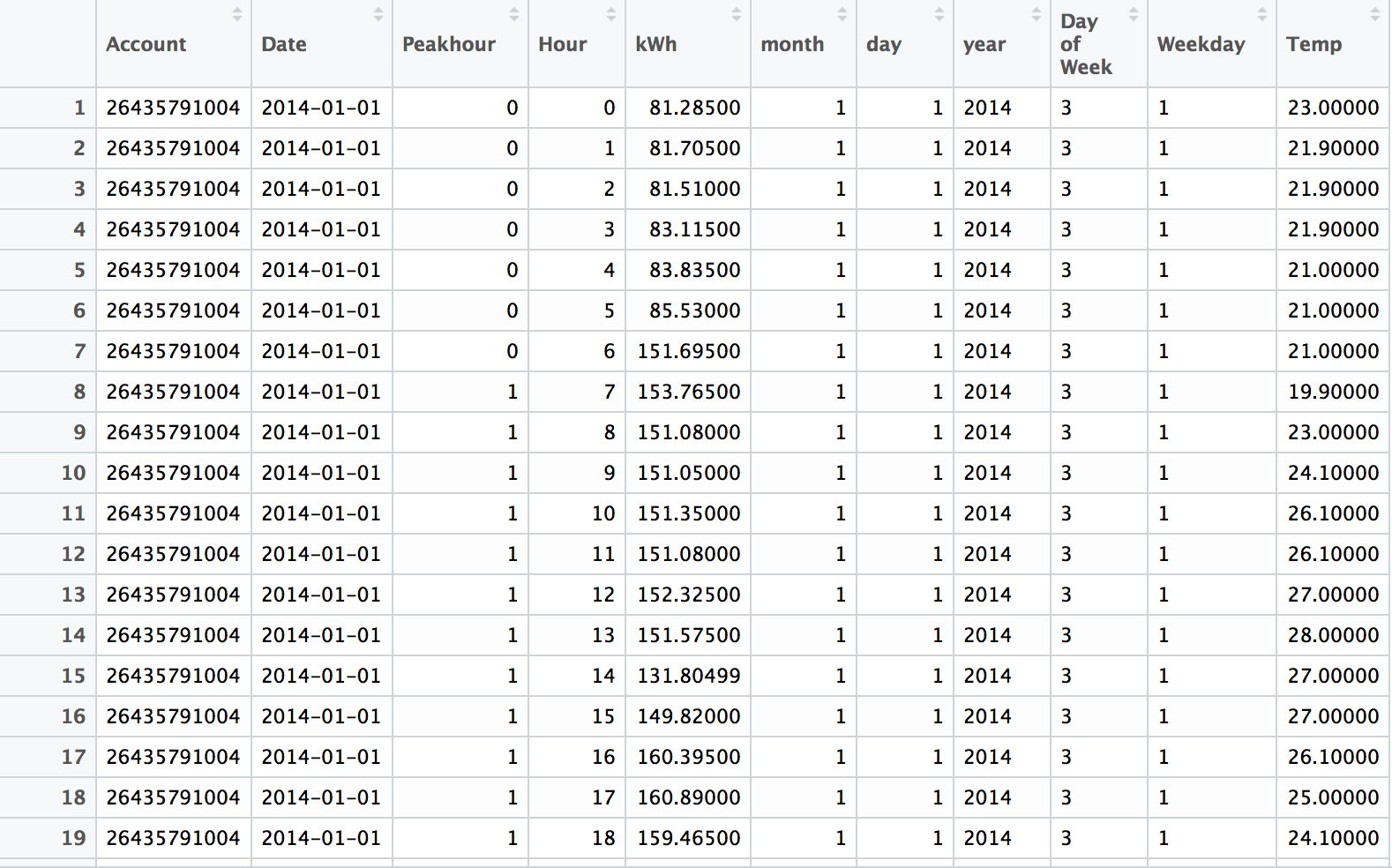
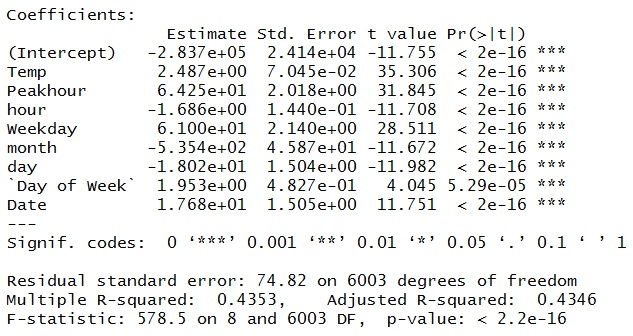


Fig : Snapshot of the final consolidated dataset.

**Observation for Approach 3:**

On evaluation of the performance metrics for the third approach we have the adjusted R square value as 0.43



So for the model we have MAPE, RMS and MAE value as 18291, 74.33 and 53.68 respectively.



**Which technique works best for you? Justify your choice.**

We have selected the third approach to continue the prediction and classification part of the assignment. Even though we have a higher value of adjusted R square value for Approach 2 which is 0.51 but we still have a lot of negative value for kWh in that dataset. Hence we have selected Approach 3 which has a lower adjusted R square value i.e. 0.43 but it gives us a better prediction.

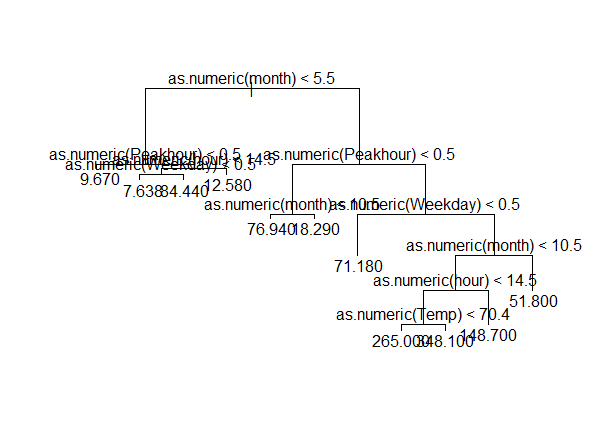
1. **Prediction**

**3.1. Regression Tree**

We are using the dataset obtained from the second approach to implement regression tree. First we split the dataset into training and test data set.

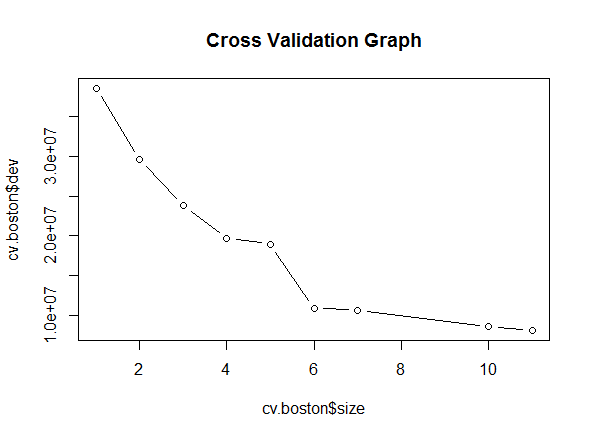
Now we run the regression tree algorithm on the train dataset for the response column kWh. We have included month, day, dayOfWeek, Weekday, hour, peakhour and temp as input parameters on the model.

Plot for Regression Tree:



**Fig: Regression Tree for the dataset**

Now we check for the cross validation graph to check for the level of pruning that is required.



**Fig: Cross Validation Graph**

Observation: On testing the accuracy for the above example of Regression tree we get the following performance metrics:





From the above graph we can see that there is no deviation on standard deviation after the 8 node. So we can decide to prune it till 8th level.

On Pruning the dataset by selecting the best 8 levels we get the following plot for Regression Tree

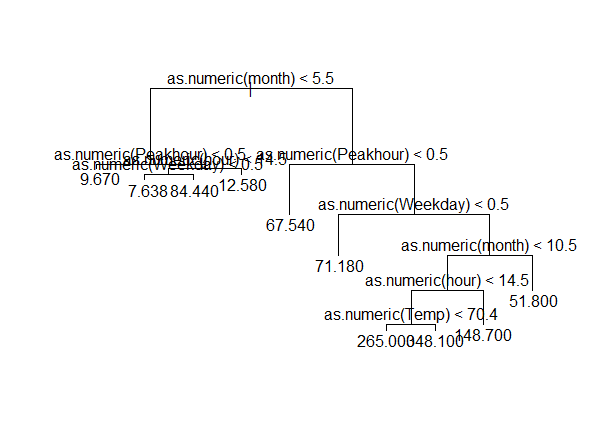


Fig: Pruned Regression Tree

Observation: On testing the accuracy for the above example of Regression tree we get the following performance metrics:





As it can be seen our mean value gets increased after applying the pruning. Hence, we stick the model of the Regression Tree without pruning.

**3.2. Neural Network**

We are using the dataset obtained from the second approach to implement neural network. First we split the dataset into training and test data set.

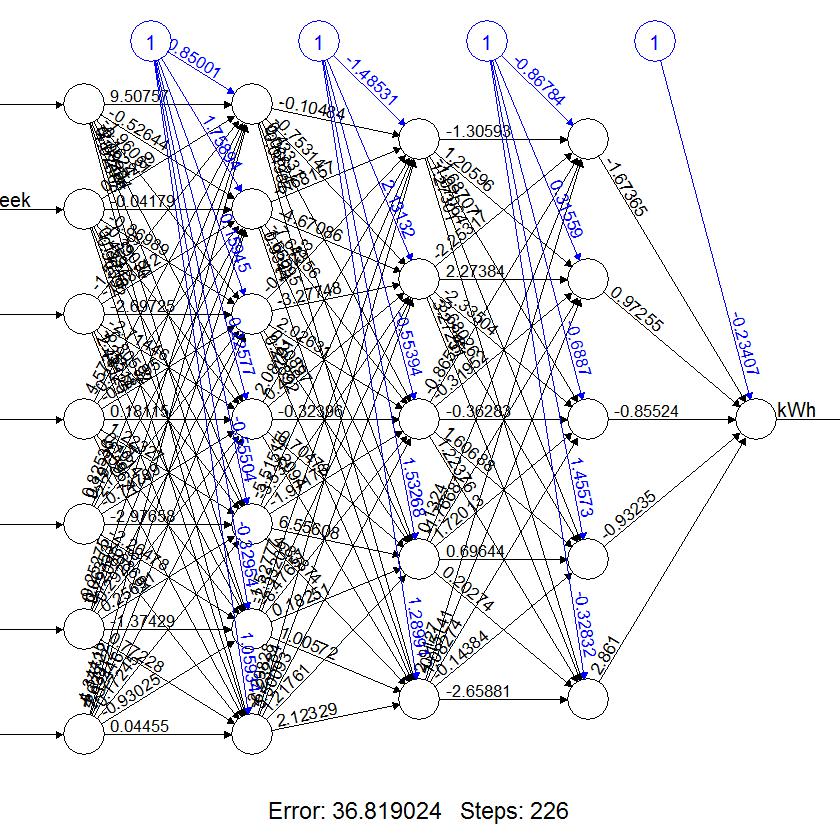
Now we run the neural network algorithm on the train dataset for the response column kWh. We have included month, day, dayOfWeek, Weekday, hour, peakhour and temp as input parameters on the model.

In the neural network we first need to normalize/scale the data before splitting it into train and test dataset. We applied the neural network function on the train dataset. Initially we were facing a lot of issues while running the neural network model due to lack of understanding of its parameters and hence we were facing the issue of always going beyond the given step size while building the network.

By iterating through this process we learnt that we were giving a low threshold value and hence it was not able to train the model. Also by increasing the parameters/neurons in the hidden layers helps our case because it easily helps to train the model.

Performance Evaluation For Neural Networks:

**Plot for the Neural Network**



1. **Classification**

**Logistic Regression**

Logistic regression is a method for fitting a regression curve, *y = f(x)*, when y is a categorical variable. The typical use of this model is predicting *y* given a set of predictors *x*. The predictors can be continuous, categorical or a mix of both.

The categorical variable *y*, in general, can assume different values. In the simplest case scenario *y* is binary meaning that it can assume either the value 1 or 0.

A classical example used in machine learning is email classification: given a set of attributes for each email such as number of words, links and pictures, the algorithm should decide whether the email is spam (1) or not (0).

**Implementation in R**

The function to be called is glm() and the fitting process is not so different from the one used in linear regression.

Step 1: reading the hour filled dataset for classification

BostonConsolidate

Step 2: we need to check for missing values and look how many unique values there are for each variable using the sapply() function which applies the function passed as argument to each column of the dataframe.

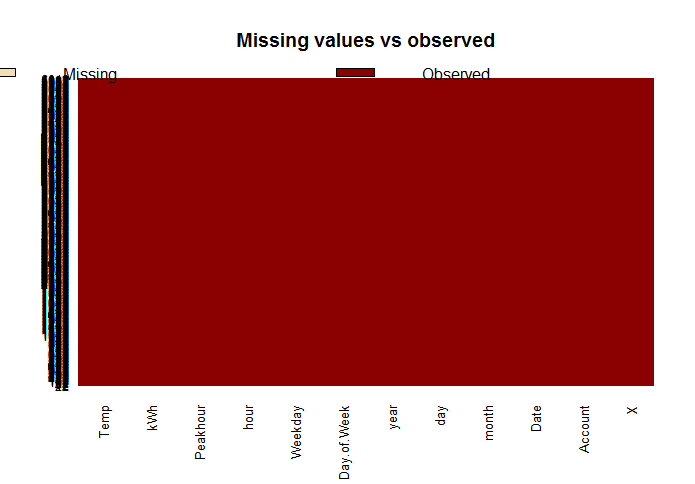
sapply(BostonConsolidate,function(x) sum(is.na(x)))

sapply(BostonConsolidate, function(x) length(unique(x)))

Step 3: Amelia package has a special plotting function missmap() that will plot your dataset and highlight missing values

library(Amelia)

missmap(BostonConsolidate, main = "Missing values vs observed")



From the above graph, we are satisfied that there are no missing values.

Step 4: we need to subset the data frame in a new data frame to use the following the data for regression. One way is to drop the columns which we are going to use here.

drops <- c("X", "Account", "Date", "year")

dataReg <- BostonConsolidate[ , !(names(BostonConsolidate) %in% drops)]

Step 5: we need to create a KWH\_Class which checks with kWh mean. we set “Above\_Normal” for kWh > mean and “Optimal” for otherwise.

However since we need to have binary factors for Logit response, we substitute the same logic with 1 and 0.

dataReg$KWH\_Class <- NULL

classList <- sapply(dataReg$kWh, function(x) {

if(x > mean(dataReg$kWh)) {dataReg$KWH\_Class <- 1}

else { dataReg$KWH\_Class <- 0} })

dataReg$KWH\_Class <- classList;

Step 6: **Model Fitting**

We split the data into two chunks: training(75 %) and testing(remaining) set. The training set will be used to fit our model which we will be testing over the testing set.

class\_smp\_size <- floor(0.75 \* nrow(BostonConsolidate))

#Set the seed to make your partition reproductible

set.seed(34)

train\_logistic <- sample(seq\_len(nrow(BostonConsolidate)), size = class\_smp\_size)

#Split the data into training and testing

train <- dataReg[train\_logistic, ]

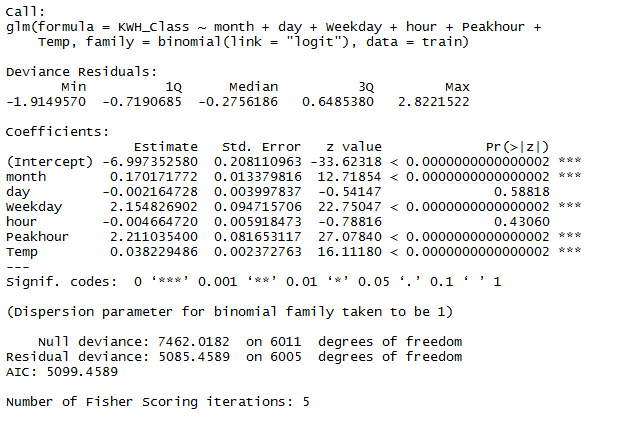
test <- dataReg[-train\_logistic, ]

Step 7: Applying the logit function glm() for getting the logistic regression. For logistic regression, the family will be binomial.

modelLogit <- glm(KWH\_Class ~ month + day + Weekday + hour + Peakhour + Temp,family=binomial(link='logit'),data=train)

Step 8: By using function summary() we obtain the results of our model.

summary(modelLogit)

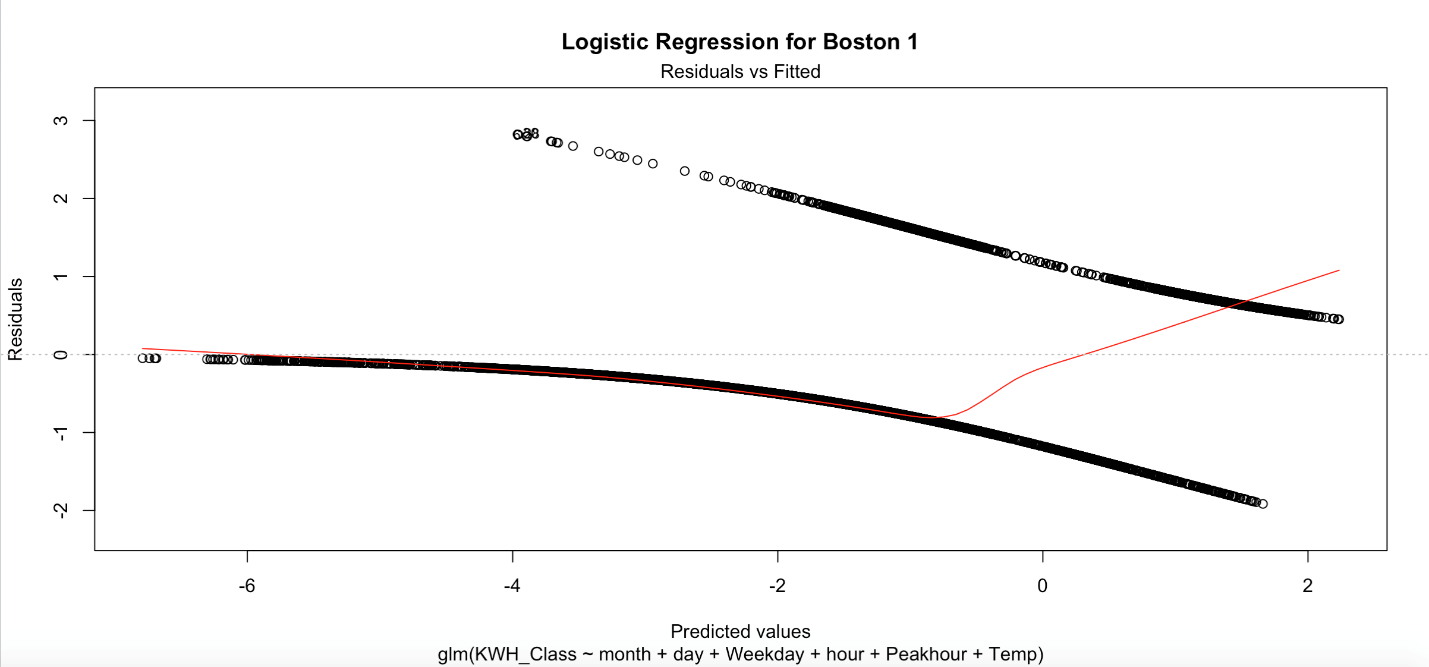
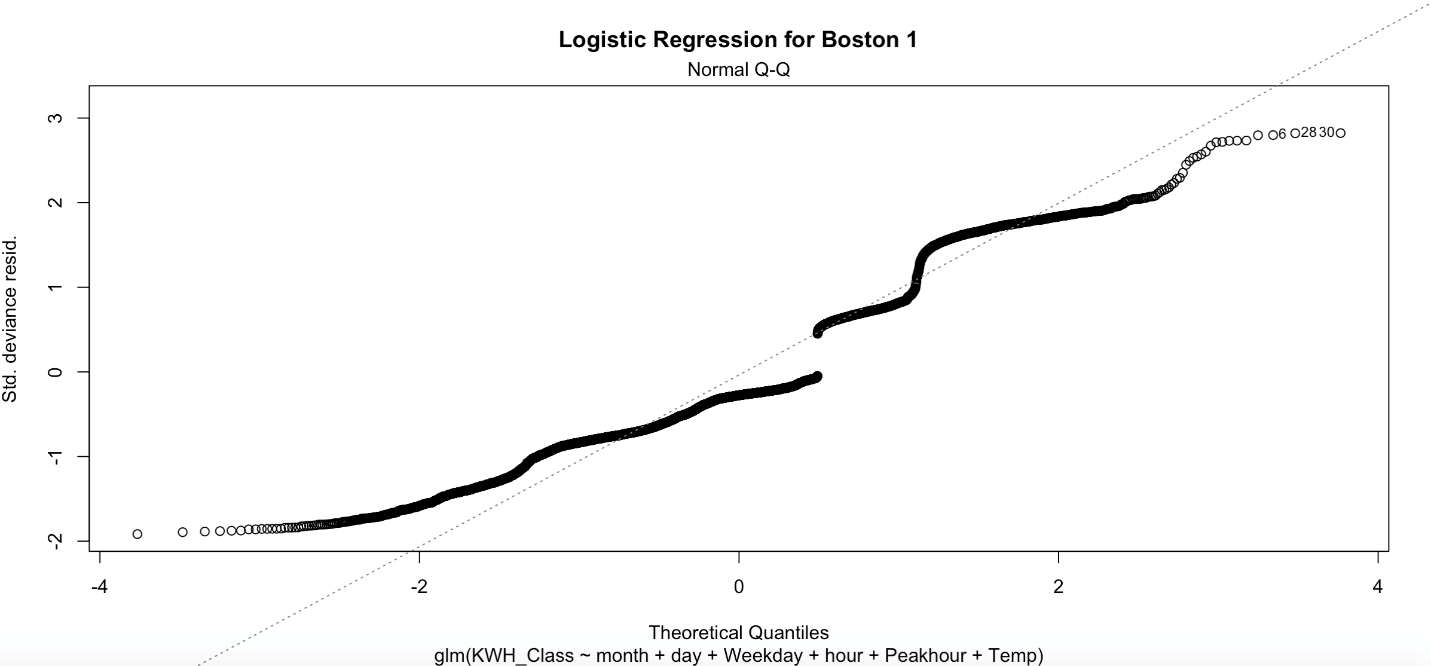
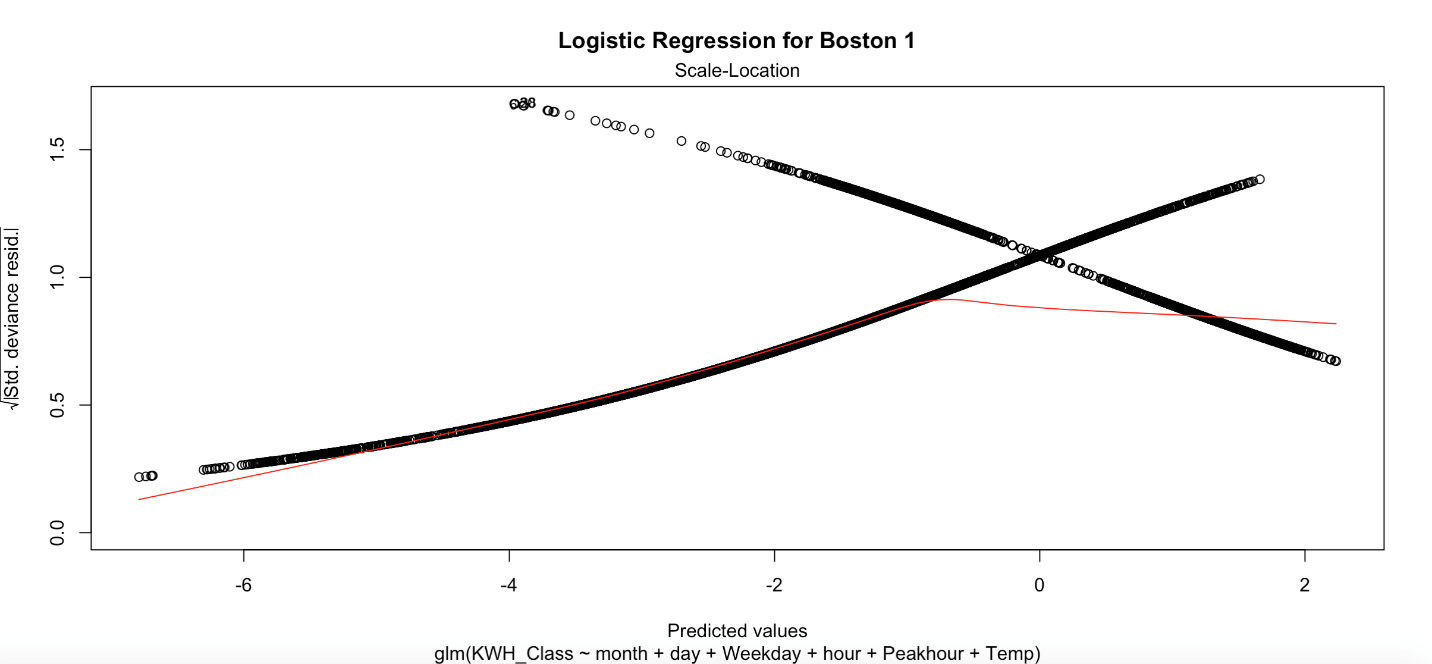
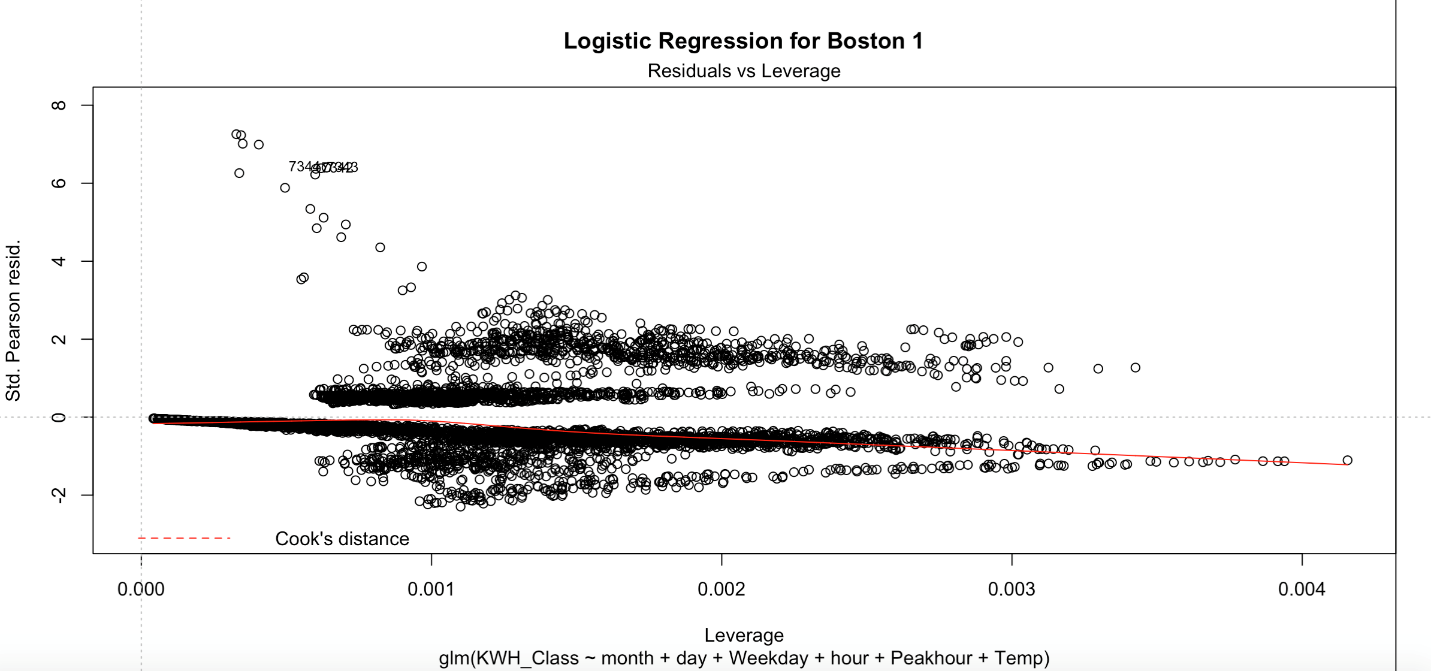


Step 9: Interpreting the results of our logistic regression model

As for the statistically significant variables, the variables which has the lowest p-value suggest a strong association with the probability of having KWH class. The negative coefficients for the predictors suggests that all other variables being equal is less likely to have survived.

Remember that in the logit model the response variable is log odds: ln(odds) = ln(p/(1p)) = a\*x1 + b\*x2 + … + z\*xn.

plot(modelLogit, uniform=TRUE, main="Logistic Regression for Boston 1")



The most common way to check these assumptions is to fit the model and then plot the residuals versus the fitted values \(\hat{y}\_i=x\_i^T \hat{\beta}\) .

* If the model assumptions are correct, the residuals should fall within an area representing a horizontal band.
* If the residuals have been standardized in some fashion (i.e., scaled by an estimate of σ), then we would expect most of them to have values within ±2 or ±3; residuals outside of this range are potential outliers.

If the plot reveals some kind of curvature, it suggests a failure of the mean model; the true relationship between μ*i* and the covariates might not be linear.

* If the variability in the residuals is not constant as we move from left to right
* then the variance V (yi) is not constant but changes as the mean μi changes.

The logistic regression model says that the mean of yi is

\(\mu\_i=n\_i \pi\_i\)

where

\(\text{log}\left(\dfrac{\pi\_i}{1-\pi\_i}\right)=x\_i^T \beta\)

and that the variance of yi is

\(V(y\_i)=n\_i \pi\_i(1-\pi\_i)\).

After fitting the model, we can calculate the Pearson’s residuals.

\(r\_i=\dfrac{y\_i-\hat{\mu}\_i}{\sqrt{\hat{V}(y\_i)}}=\dfrac{y\_i-n\_i\hat{\pi}\_i}{\sqrt{n\_i\hat{\pi}\_i(1-\hat{\pi}\_i)}}\)

or the deviance residuals. If the ni's are "large enough", these act something like standardized residuals in linear regression. To see what's happening, we can plot them against thelinear predictors,

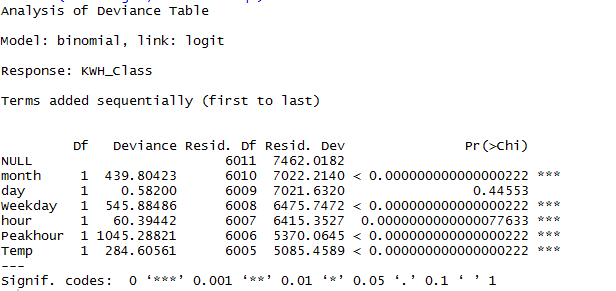
\(\hat{\eta}\_i=\text{log}\left(\dfrac{\pi\_i}{1-\pi\_i}\right)=x\_i^T \hat{\beta}\_i\)

which are the estimated log-odds of success, for cases i = 1, . . . , N.

* If the fitted logistic regression model was true, we would expect to see a horizontal band with most of the residuals falling within ± 3.
* If the ni's are small, then the plot might not have such a nice pattern, even if the model is not true.
* If outliers are present—that is, if a few residuals or even one residual is substantially larger than ± 3 — then *X*2 and *G*2 may be much larger than the degrees of freedom. In that situation, the lack of fit can be attributed to outliers, and the large residuals will be easy to find in the plot.
* If the plot of Pearson residuals versus the linear predictors reveals curvature
* **Non-constant Variance**
* Suppose that the residual plot shows non-constant variance as we move from left to right
* Another way to detect non-constancy of variance is to plot the absolute values of the residuals versus the linear predictors and look for a non-zero slope

Step 10: Now we can run the anova() function on the model to analyze the table of deviance.

anova(modelLogit, test="Chisq")



The difference between the null deviance and the residual deviance shows how our model is doing against the null model (a model with only the intercept). The wider this gap, the better. Analyzing the table we can see the drop in deviance when adding each variable one at a time. A large p-value here indicates that the model without the variable explains more or less the same amount of variation. Ultimately what you would like to see is a significant drop in deviance and the AIC.

Step 11: While no exact equivalent to the R2 of linear regression exists, the McFadden R2 index can be used to assess the model fit.

library(pscl)

pR2(modelLogit)



Step 12: we are going to plot the *ROC curve* and calculate the *AUC* (area under the curve) which are typical performance measurements for a binary classifier.

The ROC is a curve generated by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings while the AUC is the area under the ROC curve. As a rule of thumb, a model with good predictive ability should have an AUC closer to 1 (1 is ideal) than to 0.5.

library(ROCR)

p <- predict(modelLogit, newdata=test)

pr <- prediction(p, test$KWH\_Class)

prf <- performance(pr, measure = "tpr", x.measure = "fpr")

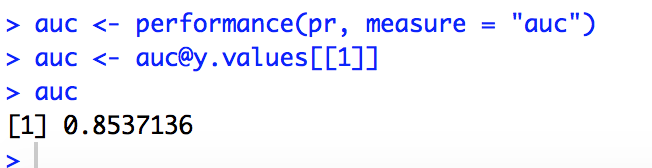
plot(prf)

auc <- performance(pr, measure = "auc")

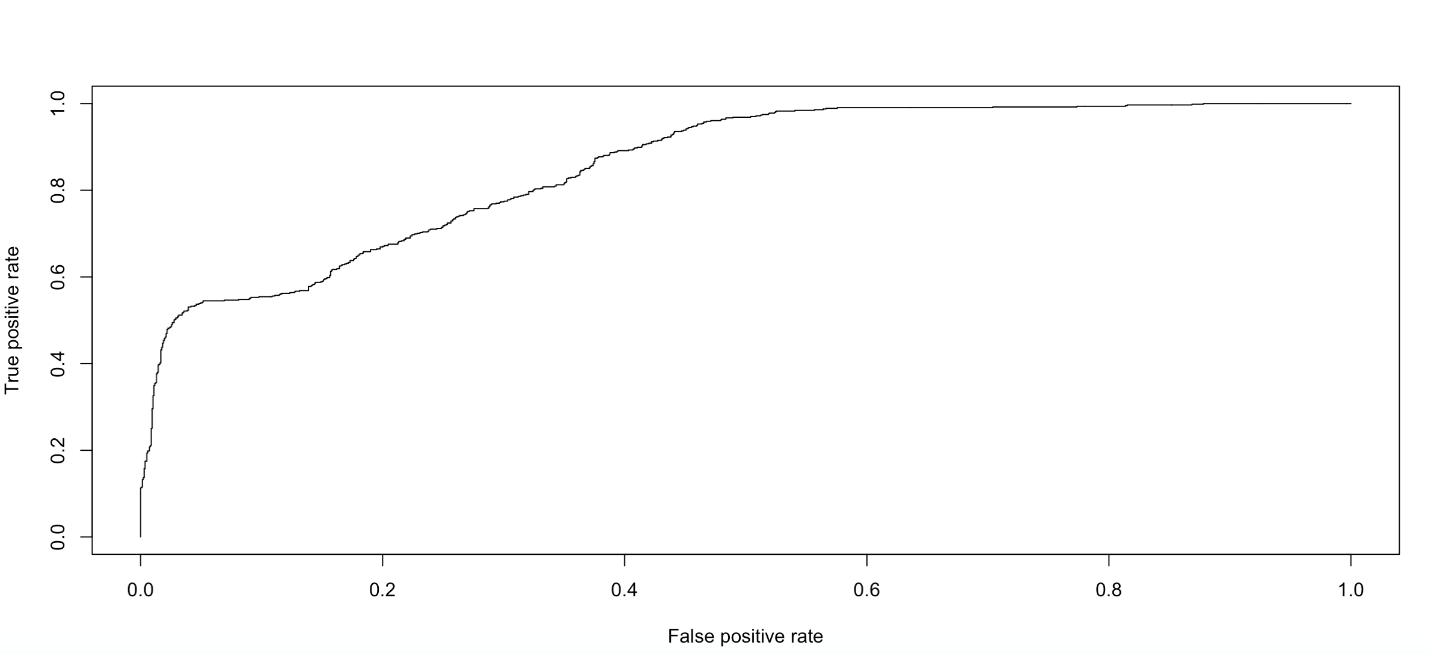
auc <- auc@y.values[[1]]

auc

We received the answer as:



Our performance ROC curve came as:



**Classification Tree:**

Decision trees have a number of advantages. They are what’s known as a glass‐box model, after the model has found the patterns in the data you can see exactly what decisions will be made for unseen data that you want to predict. They are also intuitive and can be read by people with little experience in machine learning after a brief explanation. Finally, they are the basis for some of the most powerful and popular machine learning algorithms.

Decision trees do have some drawbacks though, they are greedy. They make the decision on the current node which appear to be the best at the time, but are unable to change their minds as they grow new nodes. Perhaps a better, more pure, tree would have been grown if the gender split occurred later? It is really hard to tell, there are a huge number of decisions that could be made, and exploring every possible version of a tree is extremely computationally expensive. This is why the greedy algorithm is used.

Steps:-

We use the data set name as BostonConsolidate to read the values.

#B. classification - Decision Tree

# KWH > average KWH, KWH\_Class = "Above\_Normal" otherwise, KWH\_Class = "Optimal".

library(ISLR)

BostonConsolidate$KWH\_Class <- NULL

classList <- sapply(BostonConsolidate$kWh, function(x) {

if(x > mean(BostonConsolidate$kWh)) {BostonConsolidate$KWH\_Class <- "Above\_Normal"}

else { BostonConsolidate$KWH\_Class <- "Optimal"} })

BostonConsolidate$KWH\_Class <- classList;

#View(BostonConsolidate)

BostonConsolidate$KWH\_factor <- factor(BostonConsolidate$KWH\_Class, levels = c("Above\_Normal", "Optimal"), labels = c("1", "0"))

#test$KWH\_factor <- factor(test$KWH\_Class, levels = c("Above\_Normal", "Optimal"), labels = c("1", "0"))

#sample model fitting

#75% of the sample size

class\_smp\_size <- floor(0.75 \* nrow(BostonConsolidate))

#Set the seed to make your partition reproductible

set.seed(34)

train\_logistic <- sample(seq\_len(nrow(BostonConsolidate)), size = class\_smp\_size)

#Split the data into training and testing

train <- BostonConsolidate[train\_logistic, ]

test <- BostonConsolidate[-train\_logistic, ]

#Decision Tree model

#install.packages("rpart")

library(rpart)

classFit1 <- rpart(KWH\_Class ~ month + day + Weekday + hour + Peakhour + Temp, data=train,method="class")

summary(classFit1)

The below is the summary of our Decision Tree

Call:

rpart(formula = KWH\_Class ~ month + day + Weekday + hour + Peakhour +

Temp, data = train, method = "class")

n= 6012

CP nsplit rel error xerror xstd

1 0.18480000 0 1.0000000 1.0000000 0.01915723

2 0.11093333 2 0.6304000 0.6304000 0.01643507

3 0.04586667 3 0.5194667 0.5194667 0.01523694

4 0.02213333 4 0.4736000 0.4736000 0.01467237

5 0.01360000 8 0.3850667 0.3850667 0.01344267

6 0.01333333 13 0.3168000 0.3674667 0.01317277

7 0.01066667 14 0.3034667 0.3493333 0.01288459

8 0.01000000 16 0.2821333 0.3242667 0.01246805

Variable importance

month Temp hour Peakhour Weekday day

29 20 19 17 13 2

Node number 1: 6012 observations, complexity param=0.1848

predicted class=Optimal expected loss=0.3118762 P(node) =1

class counts: 1875 4137

probabilities: 0.312 0.688

left son=2 (3283 obs) right son=3 (2729 obs)

Primary splits:

month < 5.5 to the right, improve=381.4246, (0 missing)

Peakhour < 0.5 to the right, improve=376.8856, (0 missing)

Temp < 62.9 to the right, improve=306.3489, (0 missing)

hour < 5.5 to the right, improve=248.3620, (0 missing)

Weekday < 0.5 to the right, improve=188.7716, (0 missing)

Surrogate splits:

Temp < 48.11667 to the right, agree=0.789, adj=0.535, (0 split)

Node number 2: 3283 observations, complexity param=0.1848

predicted class=Optimal expected loss=0.4742613 P(node) =0.5460745

class counts: 1557 1726

probabilities: 0.474 0.526

left son=4 (1797 obs) right son=5 (1486 obs)

Primary splits:

Peakhour < 0.5 to the right, improve=379.2902, (0 missing)

hour < 5.5 to the right, improve=287.0951, (0 missing)

Weekday < 0.5 to the right, improve=195.2248, (0 missing)

month < 10.5 to the left, improve=169.3853, (0 missing)

Temp < 48.85 to the right, improve=122.2719, (0 missing)

Surrogate splits:

hour < 6.5 to the right, agree=0.832, adj=0.630, (0 split)

Temp < 68.05 to the right, agree=0.610, adj=0.139, (0 split)

Node number 3: 2729 observations, complexity param=0.02213333

predicted class=Optimal expected loss=0.1165262 P(node) =0.4539255

class counts: 318 2411

probabilities: 0.117 0.883

left son=6 (1490 obs) right son=7 (1239 obs)

Primary splits:

Peakhour < 0.5 to the right, improve=54.98593, (0 missing)

hour < 14.5 to the left, improve=37.20103, (0 missing)

Weekday < 0.5 to the right, improve=28.91939, (0 missing)

Temp < 32.36 to the left, improve=20.38147, (0 missing)

month < 3.5 to the left, improve=19.22186, (0 missing)

Surrogate splits:

hour < 6.5 to the right, agree=0.834, adj=0.635, (0 split)

Temp < 23.225 to the right, agree=0.582, adj=0.078, (0 split)

Node number 4: 1797 observations, complexity param=0.1109333

predicted class=Above\_Normal expected loss=0.3071786 P(node) =0.2989022

class counts: 1245 552

probabilities: 0.693 0.307

left son=8 (1499 obs) right son=9 (298 obs)

Primary splits:

month < 10.5 to the left, improve=209.746200, (0 missing)

Weekday < 0.5 to the right, improve=159.219700, (0 missing)

Temp < 49.305 to the right, improve=115.071500, (0 missing)

hour < 15.5 to the left, improve= 5.105009, (0 missing)

day < 2.5 to the left, improve= 3.073257, (0 missing)

Surrogate splits:

Temp < 48.85 to the right, agree=0.934, adj=0.601, (0 split)

Node number 5: 1486 observations, complexity param=0.0136

predicted class=Optimal expected loss=0.2099596 P(node) =0.2471723

class counts: 312 1174

probabilities: 0.210 0.790

left son=10 (962 obs) right son=11 (524 obs)

Primary splits:

hour < 3.5 to the right, improve=56.645000, (0 missing)

Weekday < 0.5 to the right, improve=41.668040, (0 missing)

Temp < 76.45 to the right, improve=15.390710, (0 missing)

month < 10.5 to the left, improve=13.242740, (0 missing)

day < 12.5 to the left, improve= 1.397427, (0 missing)

Surrogate splits:

Temp < 26.55 to the right, agree=0.65, adj=0.008, (0 split)

Node number 6: 1490 observations, complexity param=0.02213333

predicted class=Optimal expected loss=0.2080537 P(node) =0.2478377

class counts: 310 1180

probabilities: 0.208 0.792

left son=12 (795 obs) right son=13 (695 obs)

Primary splits:

hour < 13.5 to the left, improve=75.860210, (0 missing)

Weekday < 0.5 to the right, improve=50.800910, (0 missing)

Temp < 35.76667 to the left, improve=49.031100, (0 missing)

month < 3.5 to the left, improve=36.852310, (0 missing)

day < 14.5 to the left, improve= 1.773094, (0 missing)

Surrogate splits:

Temp < 65 to the left, agree=0.539, adj=0.012, (0 split)

day < 30.5 to the left, agree=0.536, adj=0.006, (0 split)

Node number 7: 1239 observations

predicted class=Optimal expected loss=0.00645682 P(node) =0.2060878

class counts: 8 1231

probabilities: 0.006 0.994

Node number 8: 1499 observations, complexity param=0.04586667

predicted class=Above\_Normal expected loss=0.1994663 P(node) =0.2493347

class counts: 1200 299

probabilities: 0.801 0.199

left son=16 (1081 obs) right son=17 (418 obs)

Primary splits:

Weekday < 0.5 to the right, improve=188.653000, (0 missing)

month < 6.5 to the right, improve= 13.794170, (0 missing)

hour < 15.5 to the left, improve= 5.352881, (0 missing)

Temp < 86.55 to the right, improve= 4.115971, (0 missing)

day < 22.5 to the right, improve= 2.332170, (0 missing)

Node number 9: 298 observations, complexity param=0.01333333

predicted class=Optimal expected loss=0.1510067 P(node) =0.04956753

class counts: 45 253

probabilities: 0.151 0.849

left son=18 (41 obs) right son=19 (257 obs)

Primary splits:

day < 4.5 to the left, improve=40.6519700, (0 missing)

Temp < 34.775 to the right, improve= 2.2917490, (0 missing)

hour < 13.5 to the left, improve= 1.5366650, (0 missing)

Weekday < 0.5 to the left, improve= 0.4500718, (0 missing)

Node number 10: 962 observations, complexity param=0.0136

predicted class=Optimal expected loss=0.3118503 P(node) =0.1600133

class counts: 300 662

probabilities: 0.312 0.688

left son=20 (676 obs) right son=21 (286 obs)

Primary splits:

Weekday < 0.5 to the right, improve=57.766900, (0 missing)

hour < 21.5 to the left, improve=35.756270, (0 missing)

month < 10.5 to the left, improve=19.584550, (0 missing)

Temp < 74.45 to the right, improve=13.563580, (0 missing)

day < 1.5 to the left, improve= 1.629732, (0 missing)

Surrogate splits:

Temp < 26.55 to the right, agree=0.704, adj=0.003, (0 split)

Node number 11: 524 observations

predicted class=Optimal expected loss=0.02290076 P(node) =0.08715902

class counts: 12 512

probabilities: 0.023 0.977

Node number 12: 795 observations, complexity param=0.02213333

predicted class=Optimal expected loss=0.3572327 P(node) =0.1322355

class counts: 284 511

probabilities: 0.357 0.643

left son=24 (562 obs) right son=25 (233 obs)

Primary splits:

Weekday < 0.5 to the right, improve=84.123850, (0 missing)

Temp < 35.86667 to the left, improve=69.800550, (0 missing)

month < 3.5 to the left, improve=64.433040, (0 missing)

hour < 11.5 to the left, improve= 4.649058, (0 missing)

day < 14.5 to the left, improve= 2.057126, (0 missing)

Node number 13: 695 observations

predicted class=Optimal expected loss=0.03741007 P(node) =0.1156021

class counts: 26 669

probabilities: 0.037 0.963

Node number 16: 1081 observations

predicted class=Above\_Normal expected loss=0.04347826 P(node) =0.1798071

class counts: 1034 47

probabilities: 0.957 0.043

Node number 17: 418 observations

predicted class=Optimal expected loss=0.3971292 P(node) =0.06952761

class counts: 166 252

probabilities: 0.397 0.603

Node number 18: 41 observations

predicted class=Above\_Normal expected loss=0.195122 P(node) =0.006819694

class counts: 33 8

probabilities: 0.805 0.195

Node number 19: 257 observations

predicted class=Optimal expected loss=0.04669261 P(node) =0.04274784

class counts: 12 245

probabilities: 0.047 0.953

Node number 20: 676 observations, complexity param=0.0136

predicted class=Optimal expected loss=0.4245562 P(node) =0.1124418

class counts: 287 389

probabilities: 0.425 0.575

left son=40 (480 obs) right son=41 (196 obs)

Primary splits:

hour < 21.5 to the left, improve=48.698950, (0 missing)

month < 10.5 to the left, improve=29.706830, (0 missing)

Temp < 43.55 to the right, improve=17.697080, (0 missing)

day < 20.5 to the right, improve= 2.180155, (0 missing)

Surrogate splits:

Temp < 27.5 to the right, agree=0.713, adj=0.01, (0 split)

Node number 21: 286 observations

predicted class=Optimal expected loss=0.04545455 P(node) =0.04757152

class counts: 13 273

probabilities: 0.045 0.955

Node number 24: 562 observations, complexity param=0.02213333

predicted class=Above\_Normal expected loss=0.4946619 P(node) =0.09347971

class counts: 284 278

probabilities: 0.505 0.495

left son=48 (322 obs) right son=49 (240 obs)

Primary splits:

month < 3.5 to the left, improve=96.090120, (0 missing)

Temp < 45.5 to the left, improve=70.774100, (0 missing)

hour < 11.5 to the left, improve= 7.707984, (0 missing)

day < 13.5 to the left, improve= 5.498973, (0 missing)

Surrogate splits:

Temp < 40.7 to the left, agree=0.884, adj=0.729, (0 split)

day < 2.5 to the right, agree=0.591, adj=0.042, (0 split)

Node number 25: 233 observations

predicted class=Optimal expected loss=0 P(node) =0.03875582

class counts: 0 233

probabilities: 0.000 1.000

Node number 40: 480 observations, complexity param=0.0136

predicted class=Above\_Normal expected loss=0.4541667 P(node) =0.07984032

class counts: 262 218

probabilities: 0.546 0.454

left son=80 (406 obs) right son=81 (74 obs)

Primary splits:

month < 10.5 to the left, improve=33.525870, (0 missing)

Temp < 47.45 to the right, improve=21.325690, (0 missing)

hour < 5.5 to the right, improve=19.271480, (0 missing)

day < 20.5 to the right, improve= 2.667094, (0 missing)

Surrogate splits:

Temp < 46.55 to the right, agree=0.935, adj=0.581, (0 split)

Node number 41: 196 observations

predicted class=Optimal expected loss=0.127551 P(node) =0.03260146

class counts: 25 171

probabilities: 0.128 0.872

Node number 48: 322 observations

predicted class=Above\_Normal expected loss=0.242236 P(node) =0.05355955

class counts: 244 78

probabilities: 0.758 0.242

Node number 49: 240 observations

predicted class=Optimal expected loss=0.1666667 P(node) =0.03992016

class counts: 40 200

probabilities: 0.167 0.833

Node number 80: 406 observations, complexity param=0.0136

predicted class=Above\_Normal expected loss=0.3743842 P(node) =0.0675316

class counts: 254 152

probabilities: 0.626 0.374

left son=160 (244 obs) right son=161 (162 obs)

Primary splits:

hour < 5.5 to the right, improve=22.847380, (0 missing)

Temp < 74.45 to the right, improve= 5.229461, (0 missing)

month < 7.5 to the right, improve= 4.132923, (0 missing)

day < 6.5 to the right, improve= 4.110448, (0 missing)

Surrogate splits:

Temp < 47.45 to the right, agree=0.608, adj=0.019, (0 split)

Node number 81: 74 observations

predicted class=Optimal expected loss=0.1081081 P(node) =0.01230872

class counts: 8 66

probabilities: 0.108 0.892

Node number 160: 244 observations

predicted class=Above\_Normal expected loss=0.2377049 P(node) =0.0405855

class counts: 186 58

probabilities: 0.762 0.238

Node number 161: 162 observations, complexity param=0.01066667

predicted class=Optimal expected loss=0.4197531 P(node) =0.02694611

class counts: 68 94

probabilities: 0.420 0.580

left son=322 (119 obs) right son=323 (43 obs)

Primary splits:

Temp < 67.45 to the left, improve=16.3097100, (0 missing)

month < 9.5 to the left, improve= 6.8460890, (0 missing)

day < 6.5 to the right, improve= 5.9636150, (0 missing)

hour < 4.5 to the right, improve= 0.8553219, (0 missing)

Node number 322: 119 observations, complexity param=0.01066667

predicted class=Above\_Normal expected loss=0.4453782 P(node) =0.01979375

class counts: 66 53

probabilities: 0.555 0.445

left son=644 (76 obs) right son=645 (43 obs)

Primary splits:

Temp < 58.88333 to the right, improve=18.2929800, (0 missing)

month < 9.5 to the left, improve=16.8137300, (0 missing)

day < 6.5 to the right, improve= 3.2508620, (0 missing)

hour < 4.5 to the right, improve= 0.4599782, (0 missing)

Surrogate splits:

month < 8.5 to the left, agree=0.807, adj=0.465, (0 split)

day < 3.5 to the right, agree=0.672, adj=0.093, (0 split)

Node number 323: 43 observations

predicted class=Optimal expected loss=0.04651163 P(node) =0.007152362

class counts: 2 41

probabilities: 0.047 0.953

Node number 644: 76 observations

predicted class=Above\_Normal expected loss=0.2368421 P(node) =0.01264138

class counts: 58 18

probabilities: 0.763 0.237

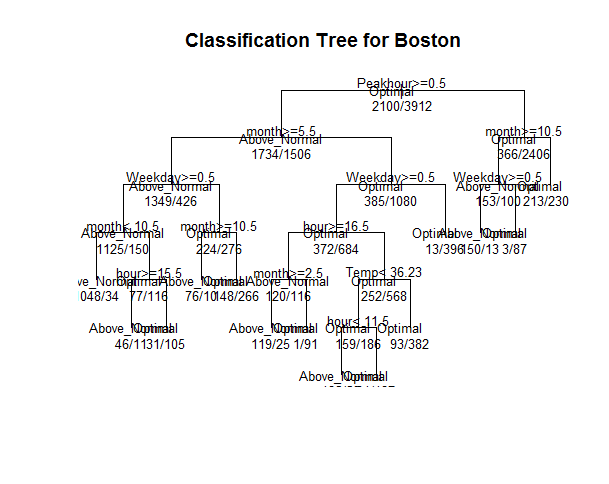
Node number 645: 43 observations

predicted class=Optimal expected loss=0.1860465 P(node) =0.007152362

class counts: 8 35

probabilities: 0.186 0.814

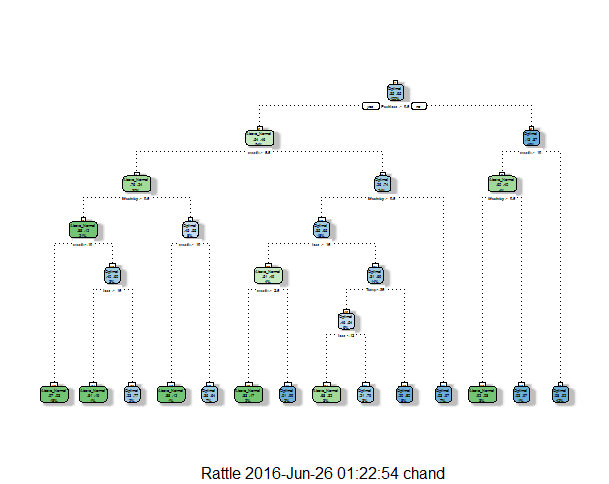
Next Step would be to plot the model:

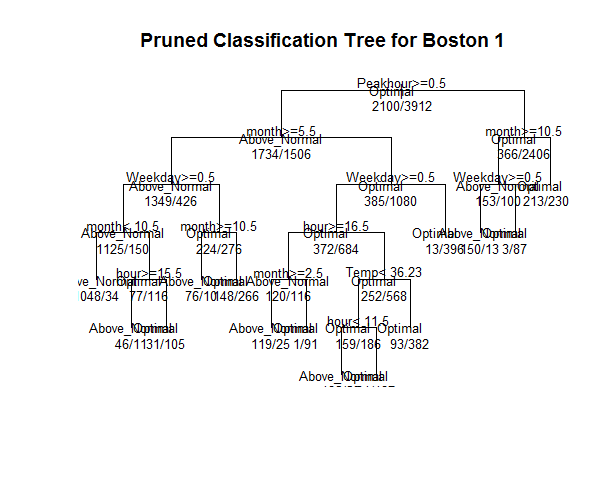


plot(classFit1)

text(classFit1, pretty=0)

fancyRpartPlot(classFit1)





We pruned the model, took the summary of the model and the cp.

Finally, we took the prediction.

**Neural Network**

**Initially we modelled using less variables and we derived a rough idea for that.**

**The steps we did for the neural network involved the similar initial steps.**

**#1. we need to check for missing values and look how many unique values there are for each variable using the sapply() function which applies the function passed as argument to each column of the dataframe.**

**sapply(trainBosnnet,function(x) sum(is.na(x)))**

**sapply(trainBosnnet, function(x) length(unique(x)))**

**#Amelia package has a special plotting function missmap() that will plot your dataset and highlight missing values**

**#install.packages("Amelia")**

**library(Amelia)**

**missmap(trainBosnnet, main = "Missing values vs observed")**

**datannet <- bostonnnet**

**datannet$KWH\_Class <- NULL**

**classList2 <- sapply(datannet$kWh, function(x) {**

**if(x > mean(datannet$kWh)) {datannet$KWH\_Class <- "Above\_Normal"}**

**else { datannet$KWH\_Class <- "Optimal"} })**

**datannet$KWH\_Class <- classList2;**

**nrow(datannet)**

**datannet$KWH\_factor <- factor(datannet$KWH\_Class, levels = c("Above\_Normal", "Optimal"), labels = c("1", "0"))**

**#normalizing the data**

**normalize2 <- function(x) {**

**return ((x - min(x)) / (max(x) - min(x))) }**

**#selecting my new subset**

**drops <- c("X", "Account", "Date", "year", "KWH\_Class")**

**dfnnet <- datannet[ , !(names(datannet) %in% drops)]**

**dfnnet$KWH\_factor<- as.numeric(dfnnet$KWH\_factor)**

**#View(dfnnet)**

**#normalizing the datannet**

**datannet\_n <- as.data.frame(lapply(dfnnet, normalize2))**

**#View(datannet\_n)**

**#import the function from Github**

**#install.packages("clusterGeneration")**

**#install.packages("neuralnet")**

**library(devtools)**

**source\_url('https://gist.githubusercontent.com/Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f1454864427e3a189c2fe33f3/nnet\_plot\_update.r')**

**source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')**

**library(clusterGeneration)**

**library(nnet)**

**require(nnet)**

**library(neuralnet)**

**#Sampling**

**#75% of the sample size**

**class\_new <- floor(0.75 \* nrow(datannet\_n))**

**#Set the seed to make your partition reproductible**

**set.seed(13)**

**train\_bostonnnet <- sample(seq\_len(nrow(datannet\_n)), size = class\_new)**

**#Split the data into training and testing**

**trainBosnnet <- datannet\_n[train\_bostonnnet, ]**

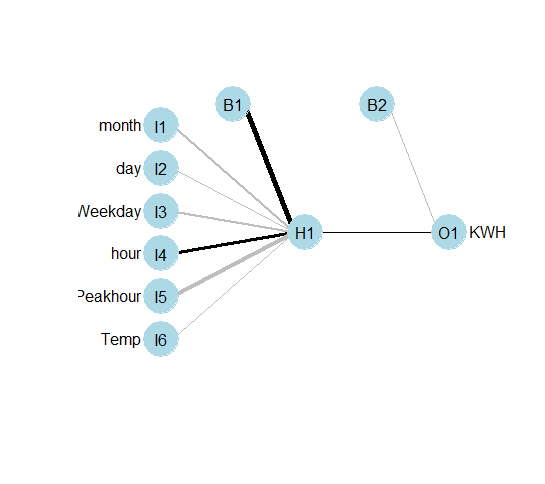
**testBosnnet<- datannet\_n[-train\_bostonnnet, ]**

**#applying the neural net algorithm**

**fitnn <- neuralnet(KWH\_factor ~ month + day + Weekday + hour + Peakhour + Temp, data = trainBosnnet, hidden=c(6), threshold=0.5, linear.output = F)**

**fitnn <- nnet(KWH\_factor ~ month + day + Weekday + hour + Peakhour + Temp, trainBosnnet, size=1, rang=0.09, hess = F, dk=15e-4, maxit = 250)**

**fitnn**

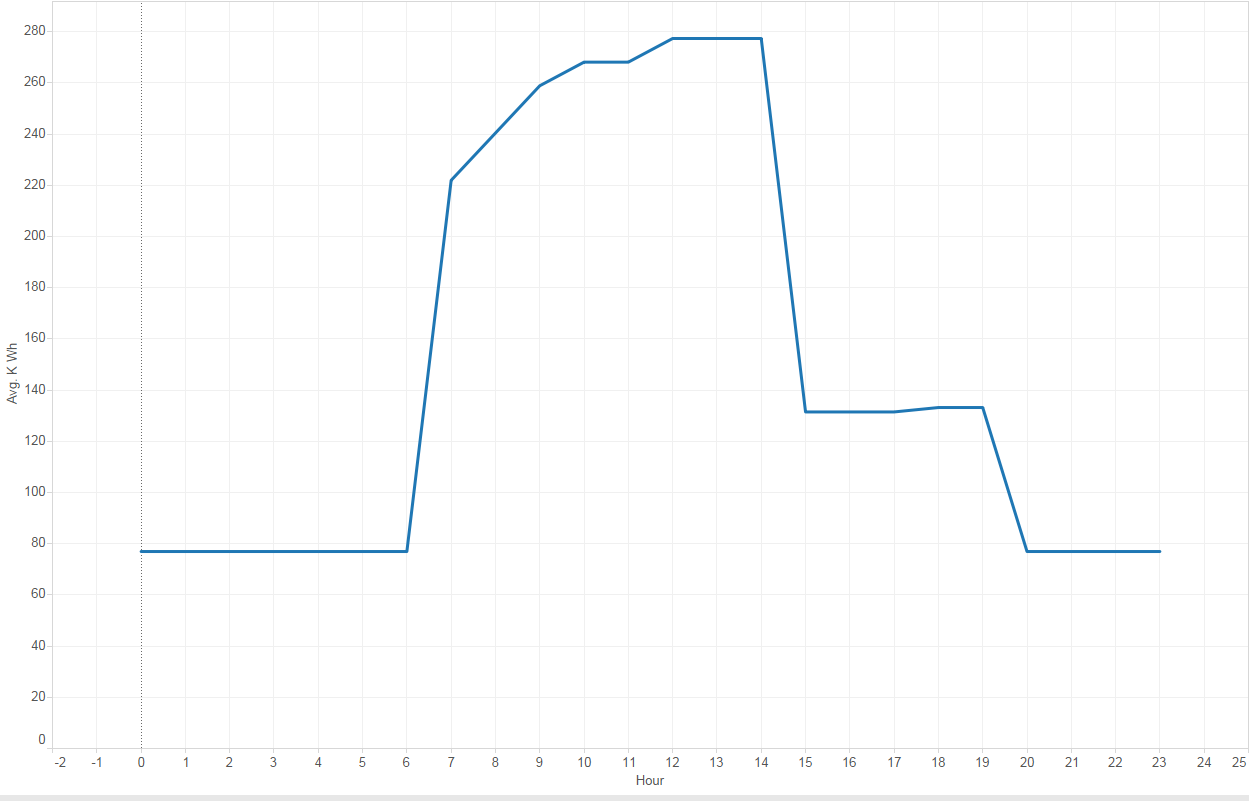
****

**summary(fitnn)**

**plot.nnet(fitnn)**

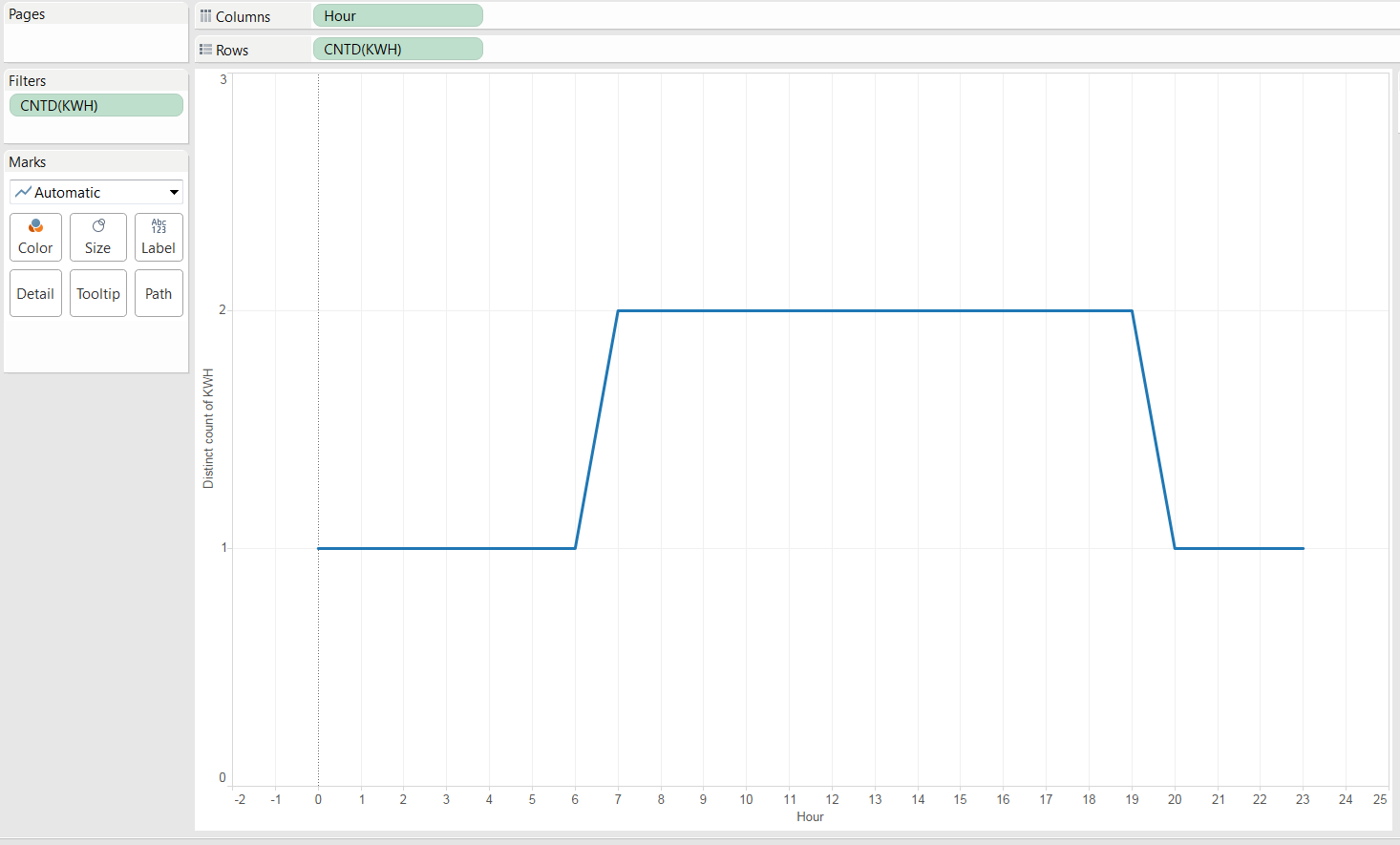
1. **Forecast**

**Regression Tree: KWH VS Hour**

****

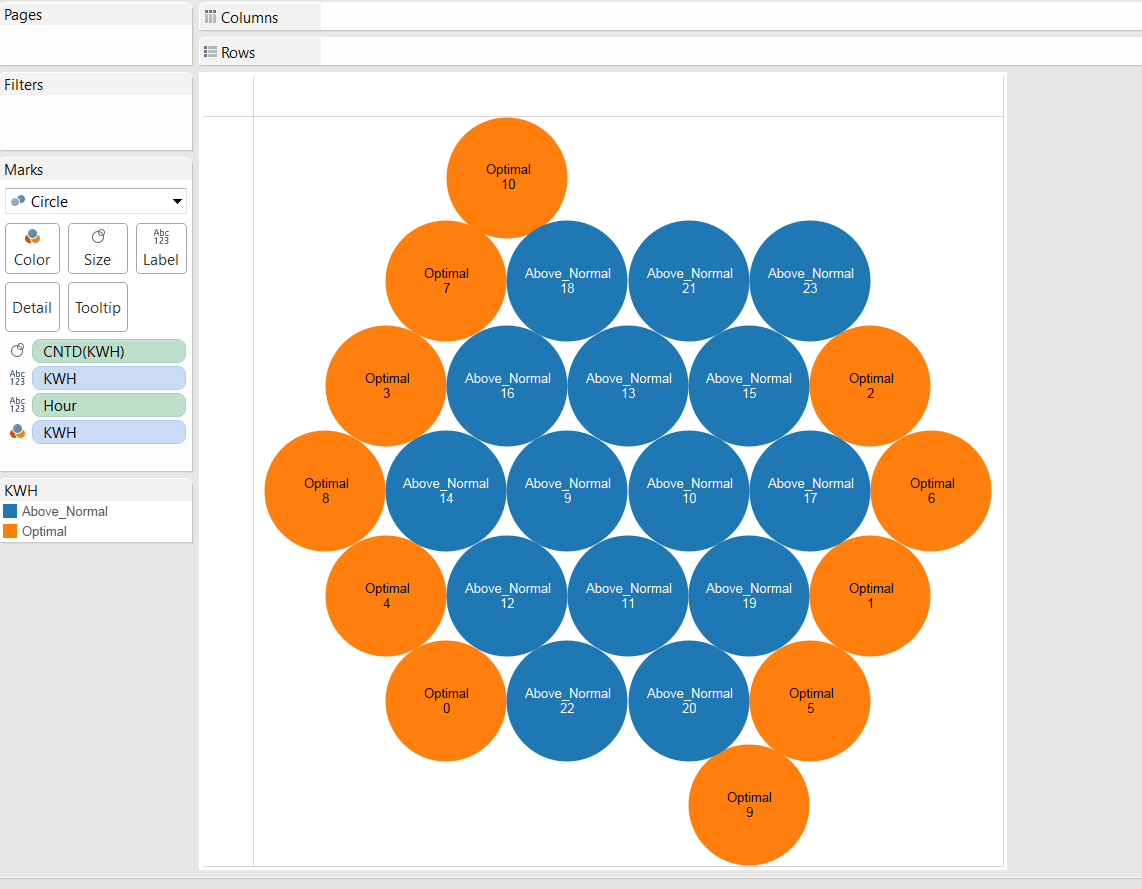
Observation: This shows us that our prediction of KWH is accurate. You can see there is a spike in KWH values after hour 6 and it drops down after hour 14 and drastically falls down after hour 19. This corresponds to the energy usage during the peakhours

**Forecasting On Classification**

****

Observation : This graph shows us that during non peak hours the KWH usage is optimal and during peak hours it is Above Optimal which corresponds to our hypothesis of more energy consumption on peak hours.

**Forecasting on Neural Networks:**



Observation : Here our hypothesis fails and you can see there is a peak in KWH usage much later during the day from hour 9 and this continues late till hour 23, which says that during non-peak hour too there is energy consumption. This may be due to deficiency in our model to predict kwh values

1. **Refrences**

* Wunderground.com: <https://www.wunderground.com/weather/api>
* WeatherData : <https://cran.r-project.org/web/packages/weatherData>
* Power Factor: <https://en.wikipedia.org/wiki/Power_factor>
* A guide to reactive power-EDF Energy: <https://www.edfenergy.com/sites/default/files/b2b-guide_to_reactive_power.pdf>