Midterm

Commercial Building Energy Consumption modeling

Team 11

Kinjal Gada
Krutika Dedhia
Ankur Vora

INTRODUCTION

As an intern at Tarja and Pasi Inc.,an energy modeling consultancy in Finland, we are supposed to work for an international project to help monitor and reduce energy consumption of 78 buildings owned by Vokia Inc. so as to make the buildings more energy efficient.

GOAL

To understand the given data, get the weather data to do feature engineering and build models for prediction, classification and clustering

MATERIALS

We were provided with two input files :

1. A RawData.csv file that has hourly power consumption information for an year with variables BuildingID, Building, Meter Number, Type, Date, Hour and Consumption(in KwH)
2. A file with Building Address and area information (in square meters).

PROCEDURE

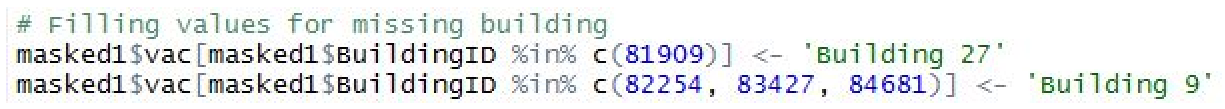
1. Data Ingestion and Wrangling
2. Modelling
3. Visualizations and Dashboard

IMPLEMENTATION

Part 1: Data Ingestion and Wrangling

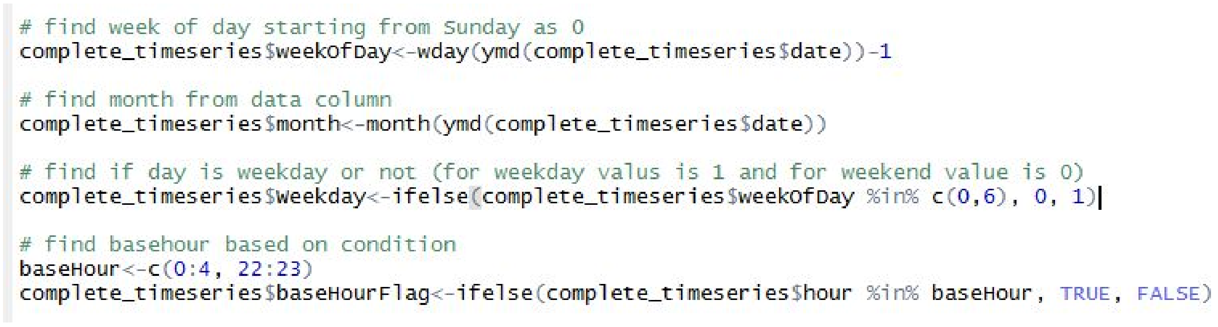
Flow is as described below:

1. Import the given RowData.csv file and and filter out the ‘elect’(Electricity) and ‘Dist_Heating’(Heat) results based on column ‘type’
2. Fill out the builing numbers which are missing based on their usage and total area (shown below). Here, we have added ‘Building 27’ and ‘Building 9’



```
# Filling values for missing building
masked1$vac[masked1$BuildingID %in% c(81909)] <- 'Building 27'
masked1$vac[masked1$BuildingID %in% c(82254, 83427, 84681)] <- 'Building 9'
```

3. We found unique combinations based on ‘BuildingID’, ‘vac’(Building Name) and ‘meternumb’. We got 78 such unique combinations.

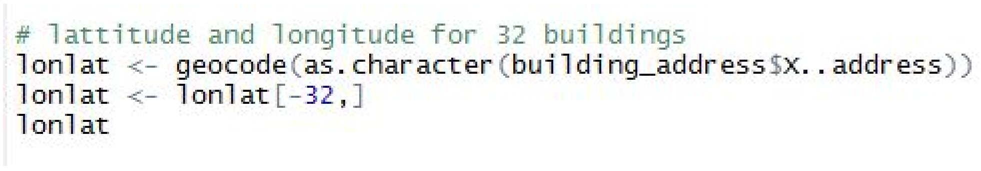


```r
# find week of day starting from Sunday as 0
complete_timeseries$weekOfDay<-wday(ymd(complete_timeseries$date))-1

# find month from data column
complete_timeseries$month<-month(ymd(complete_timeseries$date))

# find if day is weekday or not (for weekday valus is 1 and for weekend value is 0)
complete_timeseries$Weekday<-ifelse(complete_timeseries$weekOfDay %in% c(0,6), 0, 1)

# find basehour based on condition
baseHour<-c(0:4, 22:23)
complete_timeseries$baseHourFlag<-ifelse(complete_timeseries$hour %in% baseHour, TRUE, FALSE)
```

4. Merge above RowData and Address Data based on ‘Building Name’ and remove unwanted columns.
5. Find the normalised consumption using formula (Consumption/area in square meters)

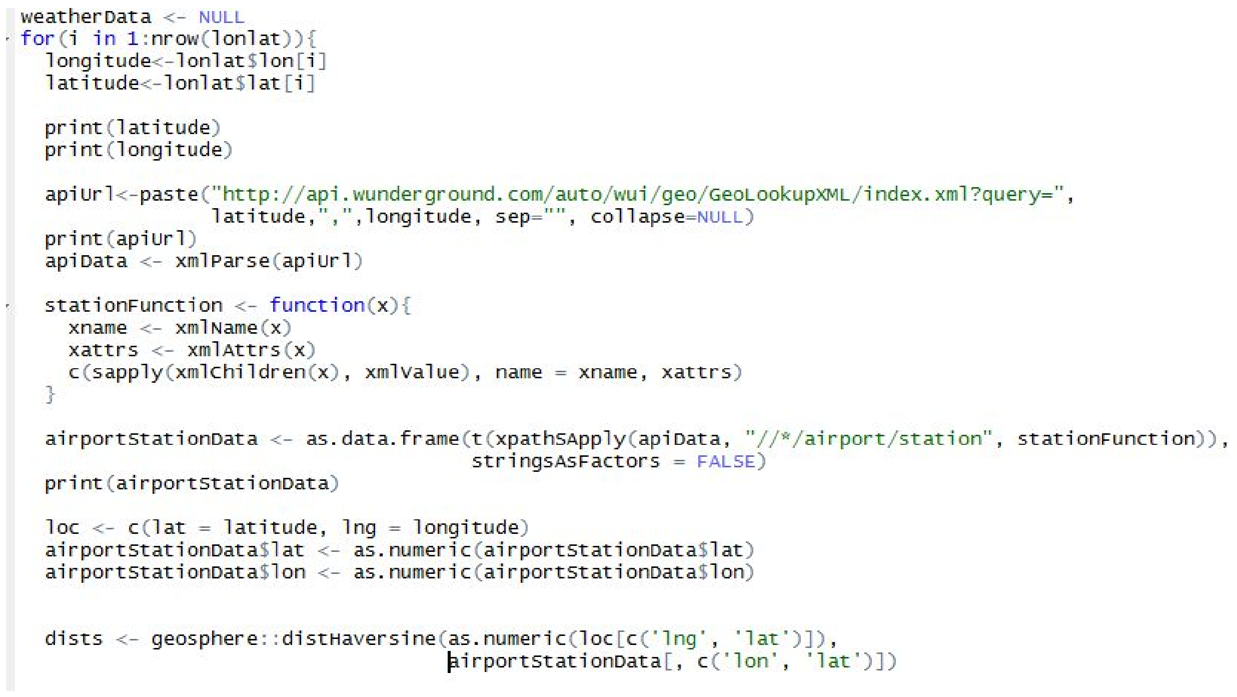
Airport Data

6. Find the latitude and longitude based on the address column

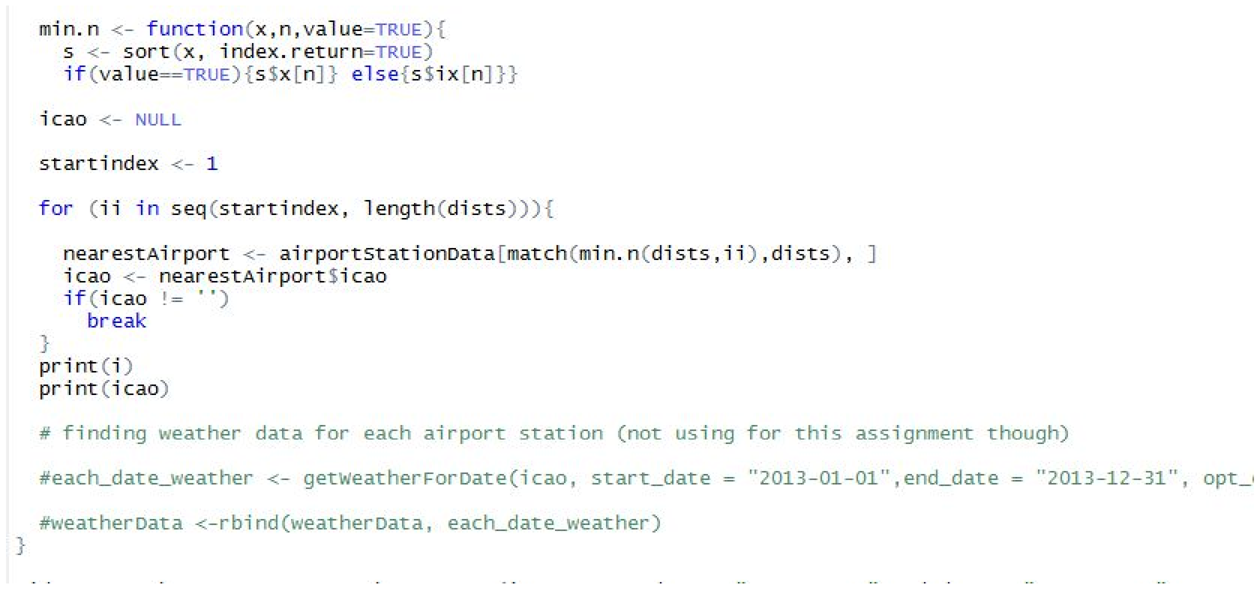


```r
# lattitude and longitude for 32 buildings
lonlat <- geocode(as.character(building_address$X..address))
lonlat <- lonlat[-32,]
lonlat
```

7. Find the nearest airport code based on the latitude and longitude.

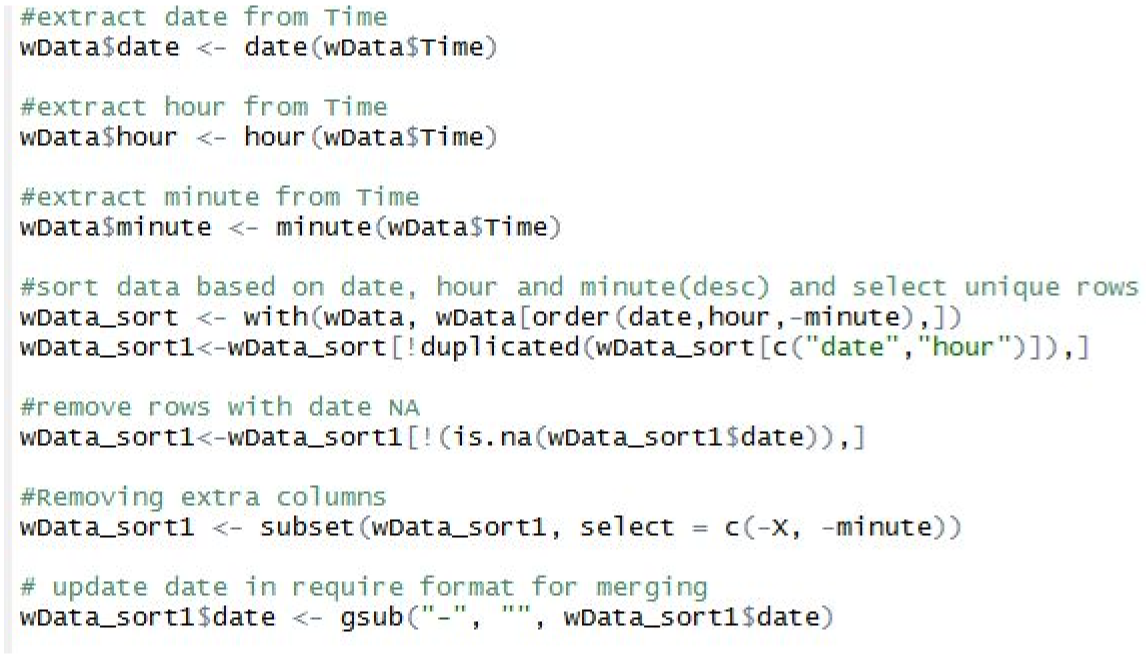


```r
weatherData <- NULL
for(i in 1:nrow(lonlat)){
  longitude<-lonlat$lon[i]
  latitude<-lonlat$lat[i]

  print(latitude)
  print(longitude)

  apiUrl<-paste("http://api.wunderground.com/auto/wui/geo/GeoLookupXML/index.xml?query=",
                latitude,",",longitude, sep="", collapse=NULL)
  print(apiUrl)
  apiData <- xmlParse(apiUrl)

  stationFunction <- function(x){
    xname <- xmlName(x)
    xattrs <- xmlAttrs(x)
    c(sapply(xmlChildren(x), xmlValue), name = xname, xattrs)
  }

  airportStationData <- as.data.frame(t(xpathSApply(apiData, "//*/airport/station", stationFunction)),
                                      stringsAsFactors = FALSE)
  print(airportStationData)

  loc <- c(lat = latitude, lng = longitude)
  airportStationData$lat <- as.numeric(airportStationData$lat)
  airportStationData$lon <- as.numeric(airportStationData$lon)


  dists <- geosphere::distHaversine(as.numeric(loc[c('lng', 'lat')]),
                                    airportStationData[, c('lon', 'lat')])
```



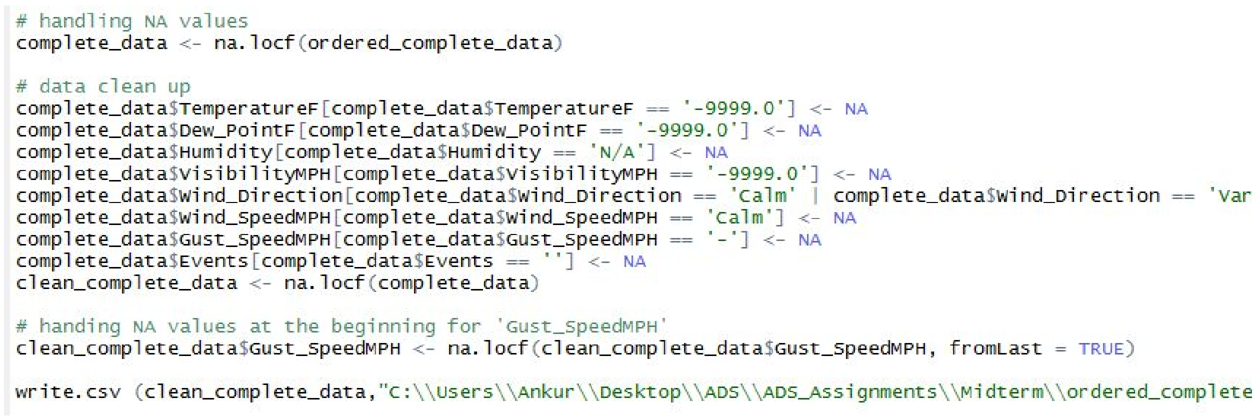
```r
  min.n <- function(x,n,value=TRUE){
    s <- sort(x, index.return=TRUE)
    if(value==TRUE){s$x[n]} else{s$ix[n]}}

  icao <- NULL

  startindex <- 1

  for (ii in seq(startindex, length(dists))){

    nearestAirport <- airportStationData[match(min.n(dists,ii),dists), ]
    icao <- nearestAirport$icao
    if(icao != '')
      break
  }
  print(i)
  print(icao)

  # finding weather data for each airport station (not using for this assignment though)

  #each_date_weather <- getWeatherForDate(icao, start_date = "2013-01-01",end_date = "2013-12-31", opt_

  #weatherData <-rbind(weatherData, each_date_weather)
}
```

8. Find the weather data for those particular airport code for all the days, 24 hours for the year 2013.
9. Perform data cleaning on weather data before merging it with the complete data for 78 unique combinations.



```
#extract date from Time
wData$date <- date(wData$Time)

#extract hour from Time
wData$hour <- hour(wData$Time)

#extract minute from Time
wData$minute <- minute(wData$Time)

#sort data based on date, hour and minute(desc) and select unique rows
wData_sort <- with(wData, wData[order(date,hour,-minute),])
wData_sort1<-wData_sort[!duplicated(wData_sort[c("date","hour")]),]

#remove rows with date NA
wData_sort1<-wData_sort1[!(is.na(wData_sort1$date)),]

#Removing extra columns
wData_sort1 <- subset(wData_sort1, select = c(-X, -minute))

# update date in require format for merging
wData_sort1$date <- gsub("-", "", wData_sort1$date)
```

10. Join the processed Raw Data, Address Data and Weather Data into one output file
11. Handle NA values and perform data cleansing before writing merged data to csv file



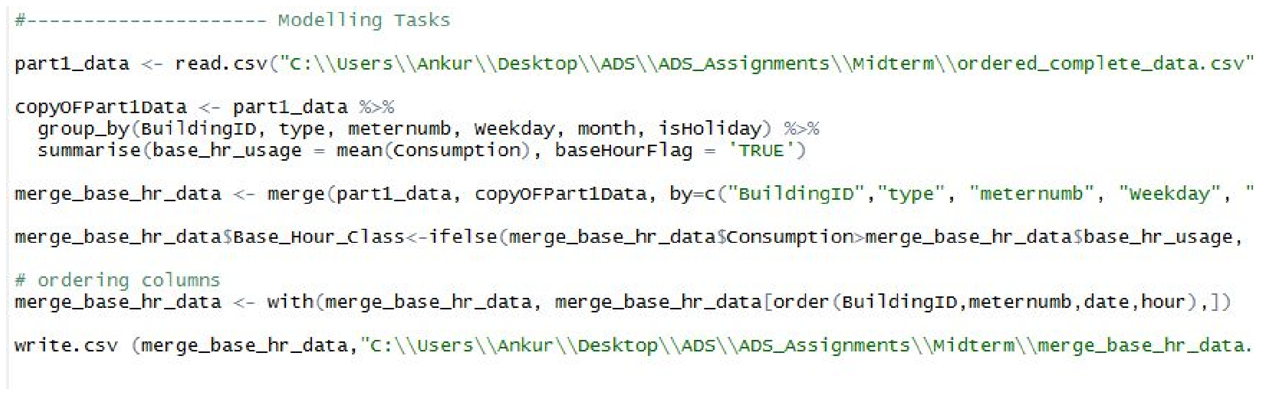
```
# handling NA values
complete_data <- na.locf(ordered_complete_data)

# data clean up
complete_data$TemperatureF[complete_data$TemperatureF == '-9999.0'] <- NA
complete_data$Dew_PointF[complete_data$Dew_PointF == '-9999.0'] <- NA
complete_data$Humidity[complete_data$Humidity == 'N/A'] <- NA
complete_data$VisibilityMPH[complete_data$VisibilityMPH == '-9999.0'] <- NA
complete_data$Wind_Direction[complete_data$Wind_Direction == 'Calm' | complete_data$Wind_Direction == 'Var
complete_data$Wind_SpeedMPH[complete_data$Wind_SpeedMPH == 'Calm'] <- NA
complete_data$Gust_SpeedMPH[complete_data$Gust_SpeedMPH == '-'] <- NA
complete_data$Events[complete_data$Events == ''] <- NA
clean_complete_data <- na.locf(complete_data)

# handing NA values at the beginning for 'Gust_SpeedMPH'
clean_complete_data$Gust_SpeedMPH <- na.locf(clean_complete_data$Gust_SpeedMPH, fromLast = TRUE)

write.csv (clean_complete_data,"C:\\Users\\Ankur\\Desktop\\ADS\\ADS_Assignments\\Midterm\\ordered_complete
```

Libraries used for Part 1:

lubridate, plyr, dplyr, stringr, httr, jsonlite, RCurl, devtools, XML, ggmap, weatherData, zoo

Part 2: Modeling tasks

1. Create a daily base_hr_usage by {building_id, type, meter_no, weekday, month, holiday}
2. Create a column base_hr_usage and input the calculated value in that column
3. Create another column called “Base_Hour_Class”. If the actual consumption is greater than the base_hr_usage, use High Else Low

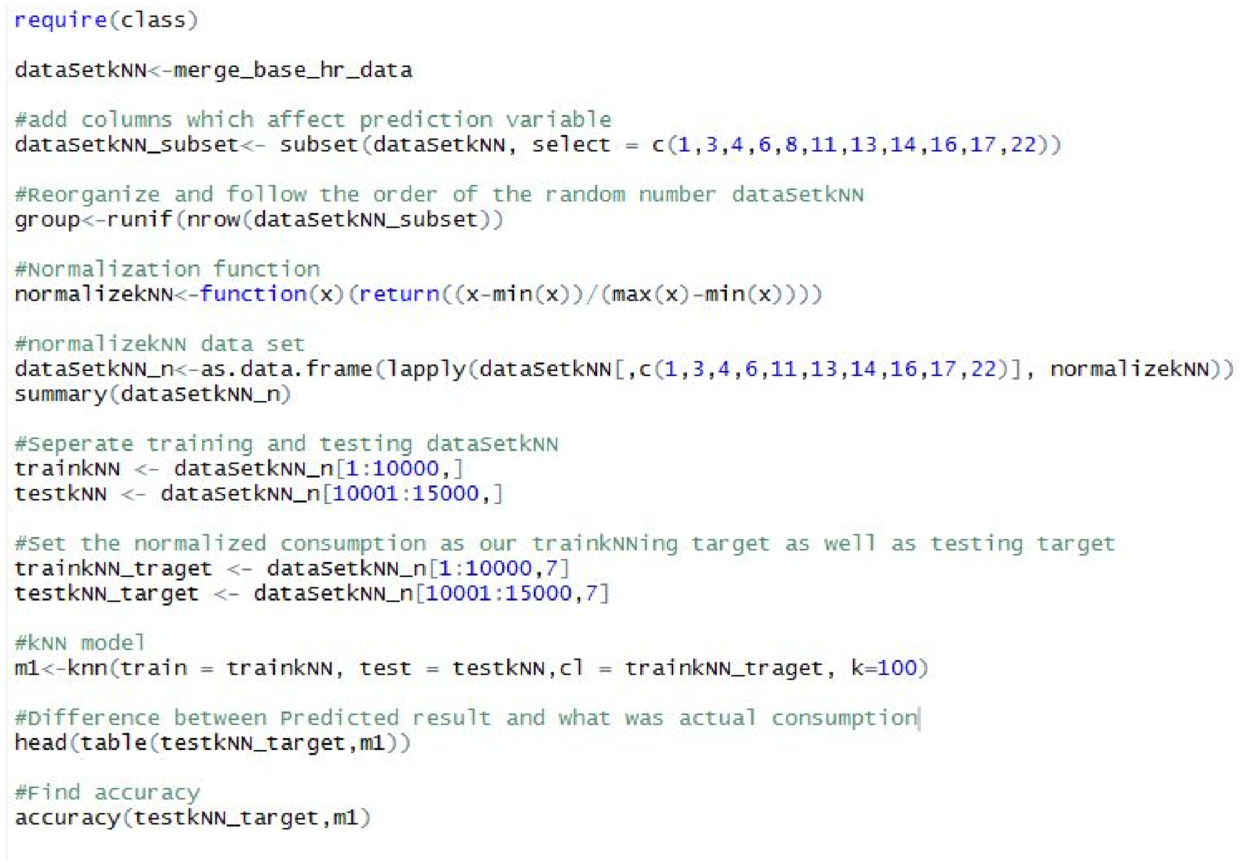


```
#--------------------- Modelling Tasks

part1_data <- read.csv("C:\\Users\\Ankur\\Desktop\\ADS\\ADS_Assignments\\Midterm\\ordered_complete_data.csv"

copyOFPart1Data <- part1_data %>%
  group_by(BuildingID, type, meternumb, Weekday, month, isHoliday) %>%
  summarise(base_hr_usage = mean(Consumption), baseHourFlag = 'TRUE')

merge_base_hr_data <- merge(part1_data, copyOFPart1Data, by=c("BuildingID","type", "meternumb", "Weekday", "

merge_base_hr_data$Base_Hour_Class<-ifelse(merge_base_hr_data$Consumption>merge_base_hr_data$base_hr_usage,

# ordering columns
merge_base_hr_data <- with(merge_base_hr_data, merge_base_hr_data[order(BuildingID,meternumb,date,hour),])

write.csv (merge_base_hr_data,"C:\\Users\\Ankur\\Desktop\\ADS\\ADS_Assignments\\Midterm\\merge_base_hr_data.
```

Prediction:

4. **Use KNN model for Prediction**

For each data point, the algorithm finds the k closest observations, and then classifies the data point to the majority. Usually, the k closest observations are defined as the ones with the smallest Euclidean distance to the data point under consideration.



```
require(class)

dataSetkNN<-merge_base_hr_data

#add columns which affect prediction variable
dataSetkNN_subset<- subset(dataSetkNN, select = c(1,3,4,6,8,11,13,14,16,17,22))

#Reorganize and follow the order of the random number dataSetkNN
group<-runif(nrow(dataSetkNN_subset))

#Normalization function
normalizekNN<-function(x)(return((x-min(x))/(max(x)-min(x))))

#normalizekNN data set
dataSetkNN_n<-as.data.frame(lapply(dataSetkNN[,c(1,3,4,6,11,13,14,16,17,22)], normalizekNN))
summary(dataSetkNN_n)

#Seperate training and testing dataSetkNN
trainkNN <- dataSetkNN_n[1:10000,]
testkNN <- dataSetkNN_n[10001:15000,]

#Set the normalized consumption as our trainkNNing target as well as testing target
trainkNN_traget <- dataSetkNN_n[1:10000,7]
testkNN_target <- dataSetkNN_n[10001:15000,7]

#kNN model
m1<-knn(train = trainkNN, test = testkNN,cl = trainkNN_traget, k=100)

#Difference between Predicted result and what was actual consumption
head(table(testkNN_target,m1))

#Find accuracy
accuracy(testkNN_target,m1)
```

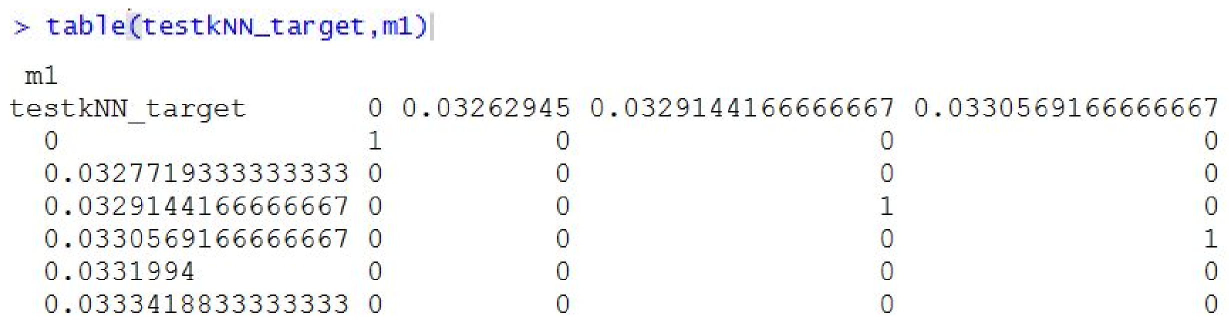
1. runif generates the random number based on the number of rows, then data will be reorganised based on those random numbers.
2. Normalize the data using min and max, which will be required for knn algorithm.
3. Separate train and test datasets out of merged data.
4. Set the normalized consumption for target variable.
5. Apply knn algorithm to the train and test simultaneously. (Value of k is taken as square root of total number of rows in the dataset)
6. Find the difference between predicted values and actual values of consumption
7. Determine accuracy of test dataset.

Feature Selection:

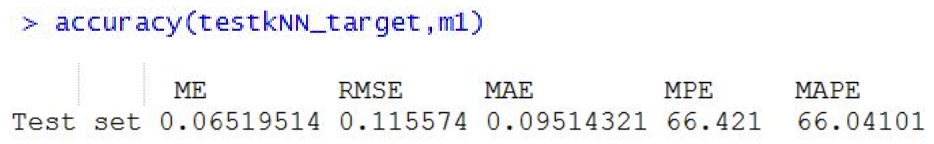
We tried and based on our experiments, we selected the variables which was majorly affecting the model.

Advantage:

The algorithm is highly unbiased in nature and makes no prior assumption of the underlying data. Being simple and effective in nature, it is easy to implement and has gained good popularity.



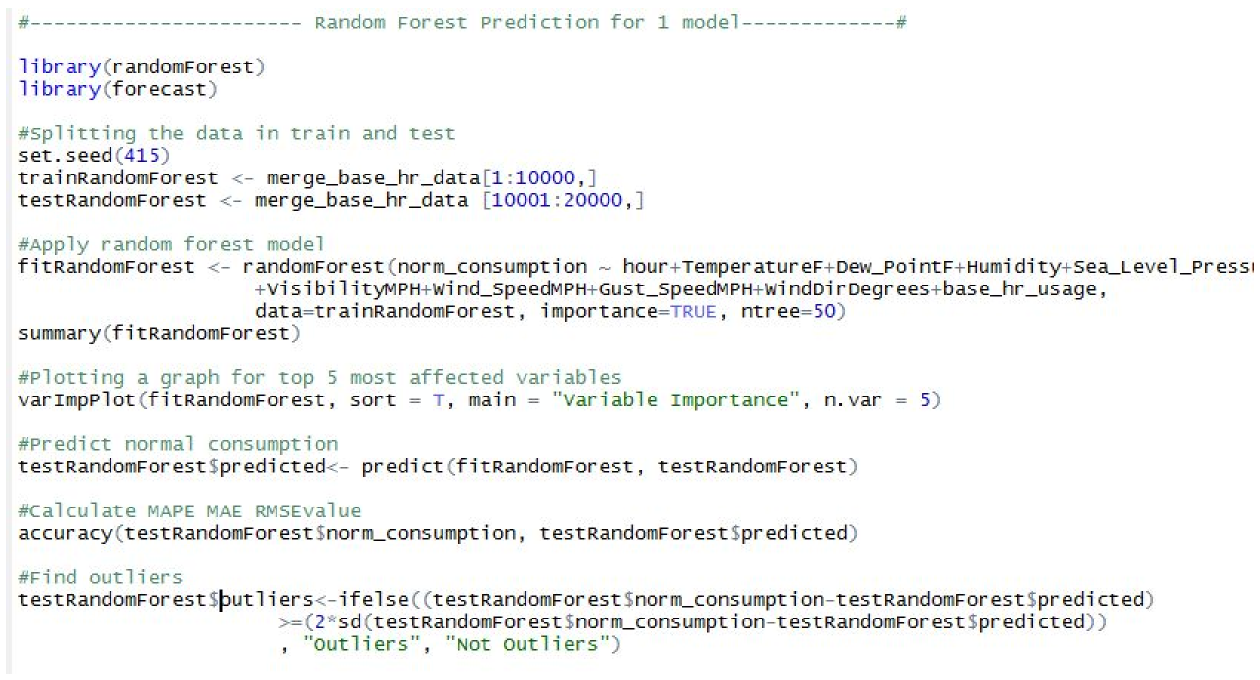
```
> table(testkNN_target,m1)

 m1
testkNN_target       0 0.03262945 0.0329144166666667 0.0330569166666667
  0                  1          0                  0                  0
  0.0327719333333333 0          0                  0                  0
  0.0329144166666667 0          0                  1                  0
  0.0330569166666667 0          0                  0                  1
  0.0331994          0          0                  0                  0
  0.0333418833333333 0          0                  0                  0
```



```
> accuracy(testkNN_target,m1)

                 ME      RMSE        MAE    MPE     MAPE
Test set 0.06519514 0.115574 0.09514321 66.421 66.04101
```

5. **Use Random Forest model for Prediction:**

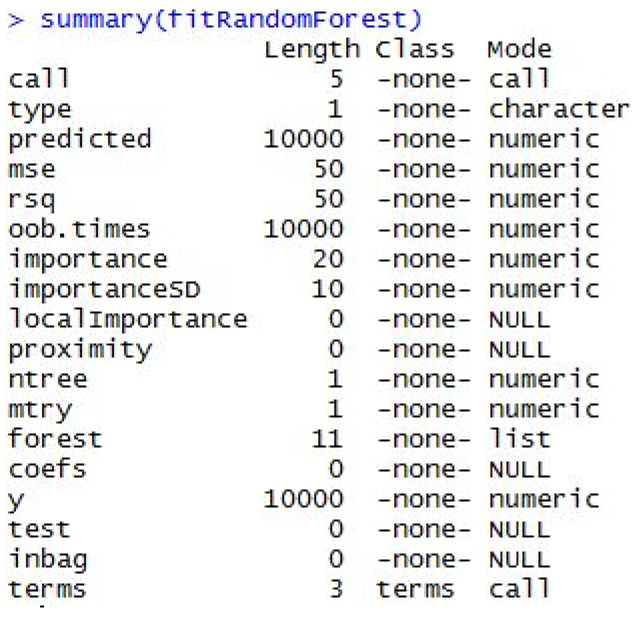
Random Forests are one way to improve the performance of decision trees. The algorithm starts by building out trees similar to the way a normal decision tree algorithm works.



```
#----------------------- Random Forest Prediction for 1 model-------------#

library(randomForest)
library(forecast)

#Splitting the data in train and test
set.seed(415)
trainRandomForest <- merge_base_hr_data[1:10000,]
testRandomForest <- merge_base_hr_data [10001:20000,]

#Apply random forest model
fitRandomForest <- randomForest(norm_consumption ~ hour+TemperatureF+Dew_PointF+Humidity+Sea_Level_Press
                   +VisibilityMPH+Wind_SpeedMPH+Gust_SpeedMPH+WindDirDegrees+base_hr_usage,
                   data=trainRandomForest, importance=TRUE, ntree=50)
summary(fitRandomForest)

#Plotting a graph for top 5 most affected variables
varImpPlot(fitRandomForest, sort = T, main = "Variable Importance", n.var = 5)

#Predict normal consumption
testRandomForest$predicted<- predict(fitRandomForest, testRandomForest)

#Calculate MAPE MAE RMSEvalue
accuracy(testRandomForest$norm_consumption, testRandomForest$predicted)

#Find outliers
testRandomForest$outliers<-ifelse((testRandomForest$norm_consumption-testRandomForest$predicted)
                     >=(2*sd(testRandomForest$norm_consumption-testRandomForest$predicted))
                     , "Outliers", "Not Outliers")
```

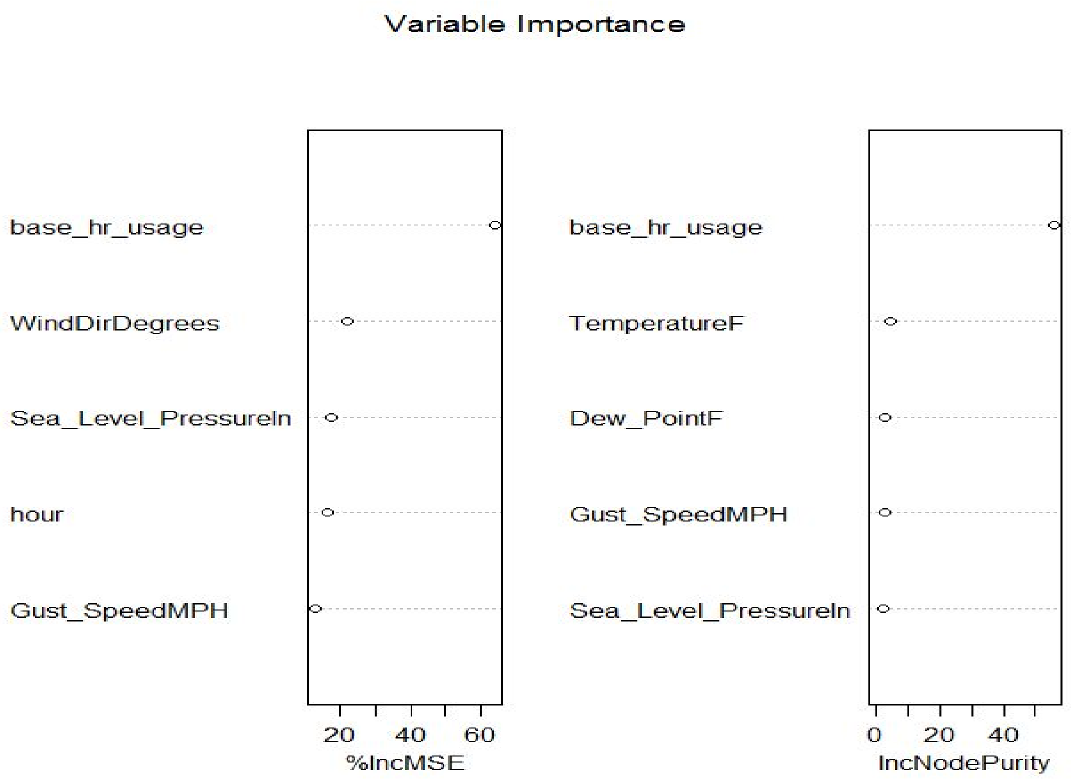
Steps:

1. Divide the merged data from part 1 into Train data and Test data.
2. Apply randomForest function to train data set passing all the variable based on feature selection along with the target variable which needs to be predicted.
3. Find out the top 5 variables which had the significance importance on the model
4. Predict the normalised consumption using model and test data set.
5. Compare the predicted values with the actual values(res to calculate MAPE, MAE and RMSE values.
6. Compute Residuals
7. Compute Standard Deviation on the Residuals
8. Tag points as outliers if the residual is >= twice the Standard deviation to determine outliers.

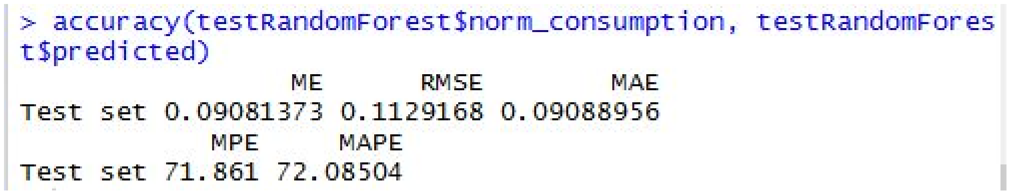


```
> summary(fitRandomForest)
                Length Class  Mode     
call                5  -none- call     
type                1  -none- character
predicted       10000  -none- numeric  
mse                50  -none- numeric  
rsq                50  -none- numeric  
oob.times       10000  -none- numeric  
importance         20  -none- numeric  
importanceSD       10  -none- numeric  
localImportance     0  -none- NULL     
proximity           0  -none- NULL     
ntree               1  -none- numeric  
mtry                1  -none- numeric  
forest             11  -none- list     
coefs               0  -none- NULL     
y               10000  -none- numeric  
test                0  -none- NULL     
inbag               0  -none- NULL     
terms               3  terms  call     
```

Variable Importance (Prediction)



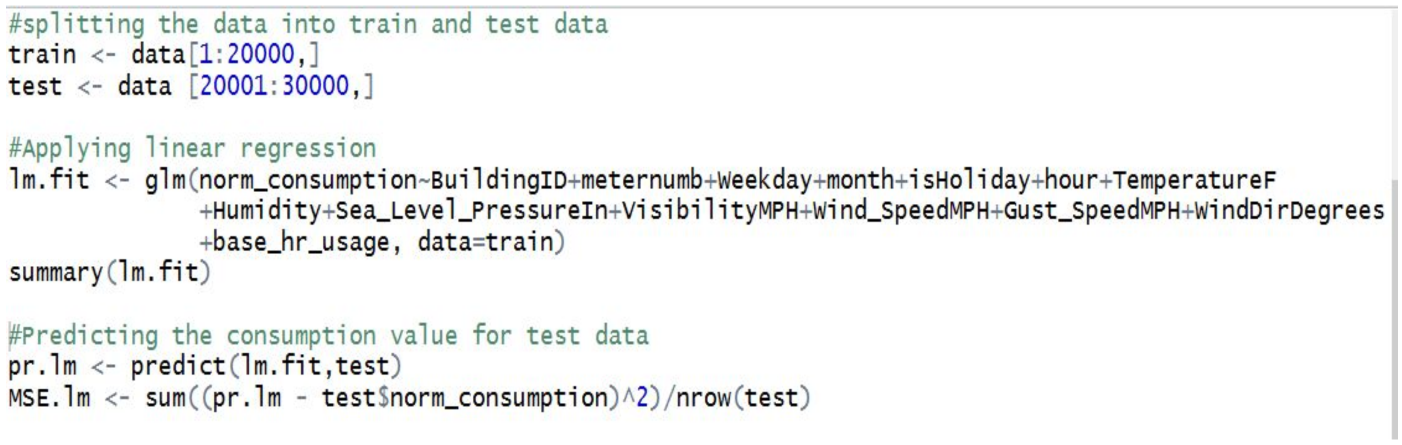
Compute MAPE, MAE, RMSE for Random Forest



```
> accuracy(testRandomForest$norm_consumption, testRandomFores
t$predicted)
                 ME      RMSE        MAE
Test set 0.09081373 0.1129168 0.09088956
            MPE     MAPE
Test set 71.861 72.08504
```

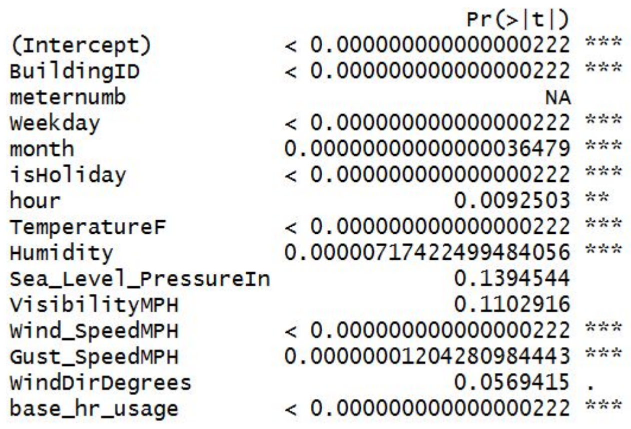
6. **Use Neural Network model for Prediction**

Reading the data file and splitting it into train and test data. Applying linear regression on train data and predicting consumption value on test data



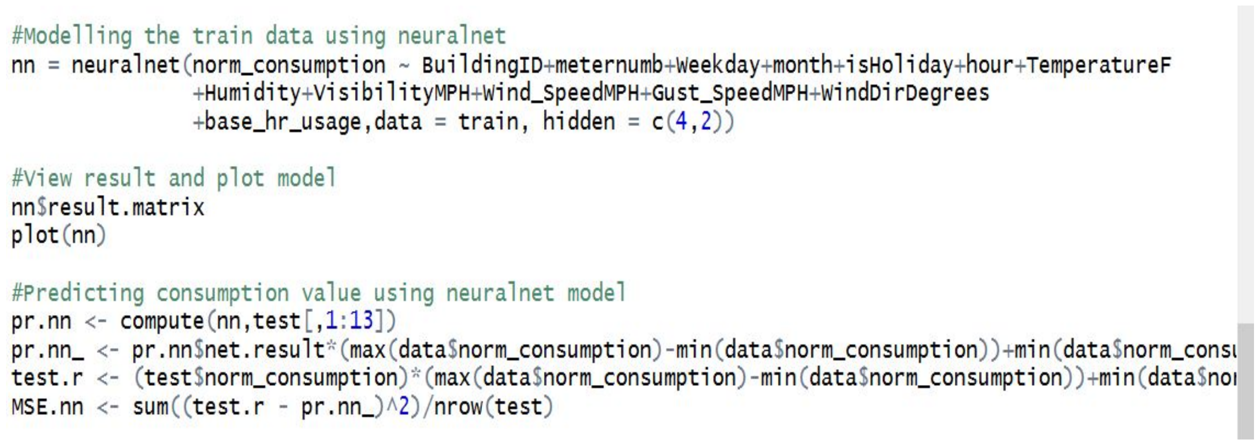
```
#splitting the data into train and test data
train <- data[1:20000,]
test <- data [20001:30000,]

#Applying linear regression
lm.fit <- glm(norm_consumption~BuildingID+meternumb+Weekday+month+isHoliday+hour+TemperatureF
              +Humidity+Sea_Level_PressureIn+VisibilityMPH+Wind_SpeedMPH+Gust_SpeedMPH+WindDirDegrees
              +base_hr_usage, data=train)
summary(lm.fit)

#Predicting the consumption value for test data
pr.lm <- predict(lm.fit,test)
MSE.lm <- sum((pr.lm - test$norm_consumption)^2)/nrow(test)
```

Output : summary(lm.fit)



```
                                   Pr(>|t|)
(Intercept)          < 0.000000000000000222 ***
BuildingID           < 0.000000000000000222 ***
meternumb                                NA
Weekday              < 0.000000000000000222 ***
month                0.0000000000000036479 ***
isHoliday            < 0.000000000000000222 ***
hour                              0.0092503 **
TemperatureF         < 0.000000000000000222 ***
Humidity             0.00000717422499484056 ***
Sea_Level_PressureIn              0.1394544
VisibilityMPH                     0.1102916
Wind_SpeedMPH        < 0.000000000000000222 ***
Gust_SpeedMPH        0.0000001204280984443 ***
WindDirDegrees                    0.0569415 .
base_hr_usage        < 0.000000000000000222 ***
```

Modelling using Neural Network for the variables selected from linear regression model and predicting the consumption value. Library used for this function is ‘neuralnet’.

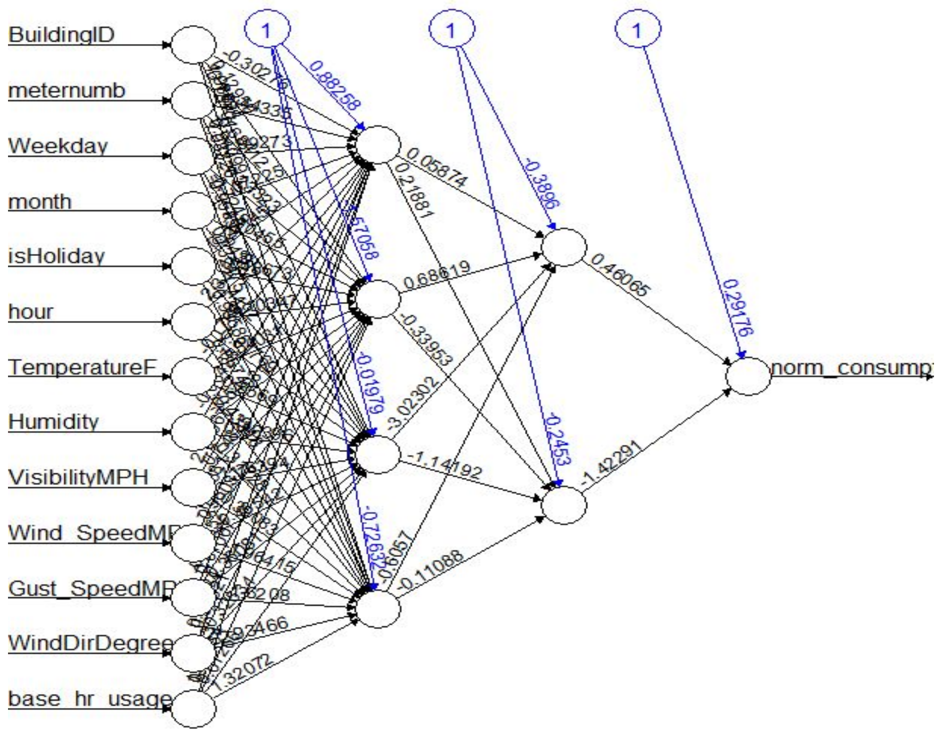


```
#Modelling the train data using neuralnet
nn = neuralnet(norm_consumption ~ BuildingID+meternumb+Weekday+month+isHoliday+hour+TemperatureF
              +Humidity+VisibilityMPH+Wind_SpeedMPH+Gust_SpeedMPH+WindDirDegrees
              +base_hr_usage,data = train, hidden = c(4,2))

#View result and plot model
nn$result.matrix
plot(nn)

#Predicting consumption value using neuralnet model
pr.nn <- compute(nn,test[,1:13])
pr.nn_ <- pr.nn$net.result*(max(data$norm_consumption)-min(data$norm_consumption))+min(data$norm_consu
test.r <- (test$norm_consumption)*(max(data$norm_consumption)-min(data$norm_consumption))+min(data$nor
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test)
```

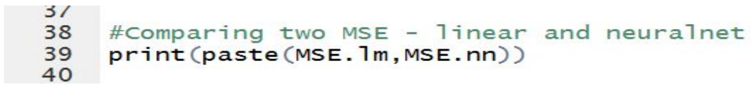
Output : Error is 38 and steps required to compute it are 40.

```
error                          38.665794753553
reached.threshold               0.005914735428
steps                          40.000000000000
Intercept.to.1layhid1          -0.002379880655
BuildingID.to.1layhid1          0.045691452963
meternumb.to.1layhid1          -0.179148758141
Weekday.to.1layhid1             1.987035992847
month.to.1layhid1              -0.025830426913
isHoliday.to.1layhid1          -1.666438139297
hour.to.1layhid1                0.478542303731
TemperatureF.to.1layhid1        0.236840268856
Humidity.to.1layhid1            0.273999754719
VisibilityMPH.to.1layhid1       1.938473289106
Wind_SpeedMPH.to.1layhid1      -1.715176111893
Gust_SpeedMPH.to.1layhid1       1.519510192607
WindDirDegrees.to.1layhid1      1.627825537646
base_hr_usage.to.1layhid1       0.137473519255
Intercept.to.1layhid2           0.739547688487
BuildingID.to.1layhid2         -0.646415826187
meternumb.to.1layhid2           1.002388457168
Weekday.to.1layhid2            -0.233521155799
month.to.1layhid2              -0.073213268054
isHoliday.to.1layhid2           0.525681073549
hour.to.1layhid2               -0.084031455259
TemperatureF.to.1layhid2        0.809650036825
Humidity.to.1layhid2            1.035382109693
VisibilityMPH.to.1layhid2      -1.043339170226
Wind_SpeedMPH.to.1layhid2       0.470699715558
Gust_SpeedMPH.to.1layhid2      -0.101522941579
WindDirDegrees.to.1layhid2     -0.447053616366
base_hr_usage.to.1layhid2      -0.066211585261
Intercept.to.norm_consumption  -0.955229005852
1layhid.1.to.norm_consumption   1.144885068483
1layhid.2.to.norm_consumption  -0.556043698064
```

Graph



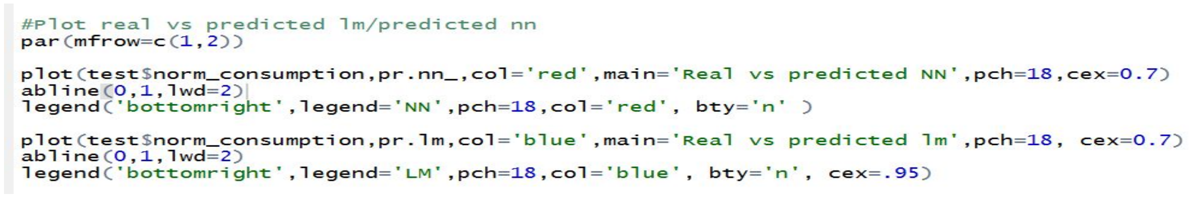
We are using 2 hidden layers with neurons 4 and 2 in each layer. The number of hidden layers does not need to be specific but normally depends on the input variables. Here the ratio we are using is 13:4:2:1 indicating we have taken 13 input variables, 2 hidden layers computing to a single output.



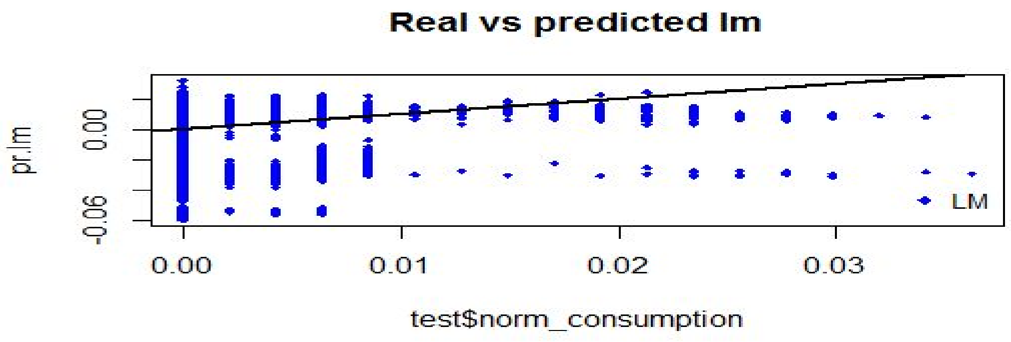
```
#Comparing two MSE - linear and neuralnet
print(paste(MSE.lm,MSE.nn))
```

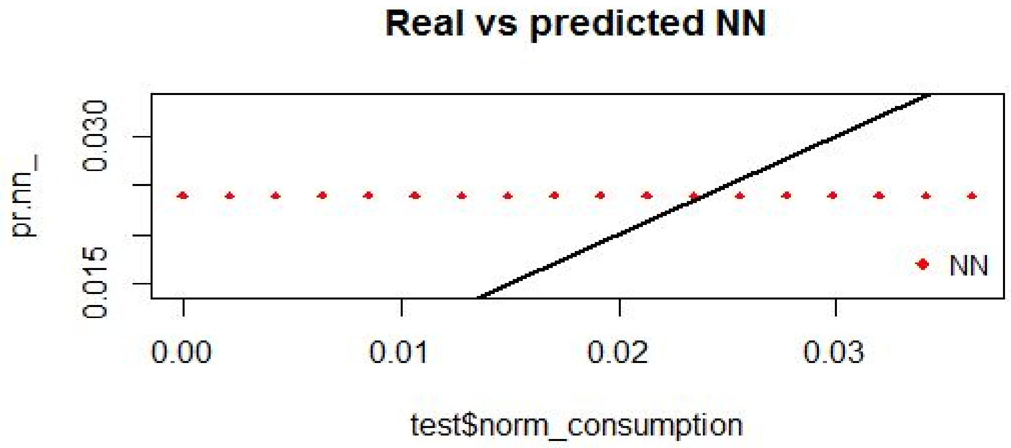
Output : [1] "0.000386801383301534 0.000511974666288221"

This compares the two MSE values and gives the difference between the one computed from linear regression and computed from neural network.



```
#Plot real vs predicted lm/predicted nn
par(mfrow=c(1,2))

plot(test$norm_consumption,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n' )

plot(test$norm_consumption,pr.lm,col='blue',main='Real vs predicted lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)
```





Best Model:

Based on all the MAPE, MAE and RMSE values, we think Random Forest is the best model.

Furthermore, It has some advantages over other algorithms.

1. Always observed lower prediction error than KNN and Neural Network.

2. Deals really well with uneven data sets that have missing variables.

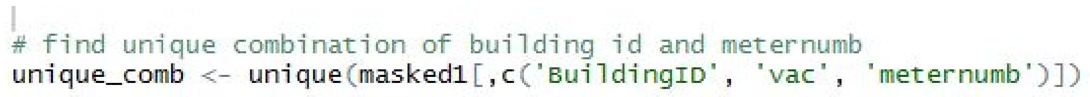
3. Gives you a really good idea of which features in your data set are the most important for free.

4. Generally trains faster than Neural Network.

Implementation for 78 models

Steps:

1. Taking the input file as the output from part1, first get unique 78 models with following unique function.



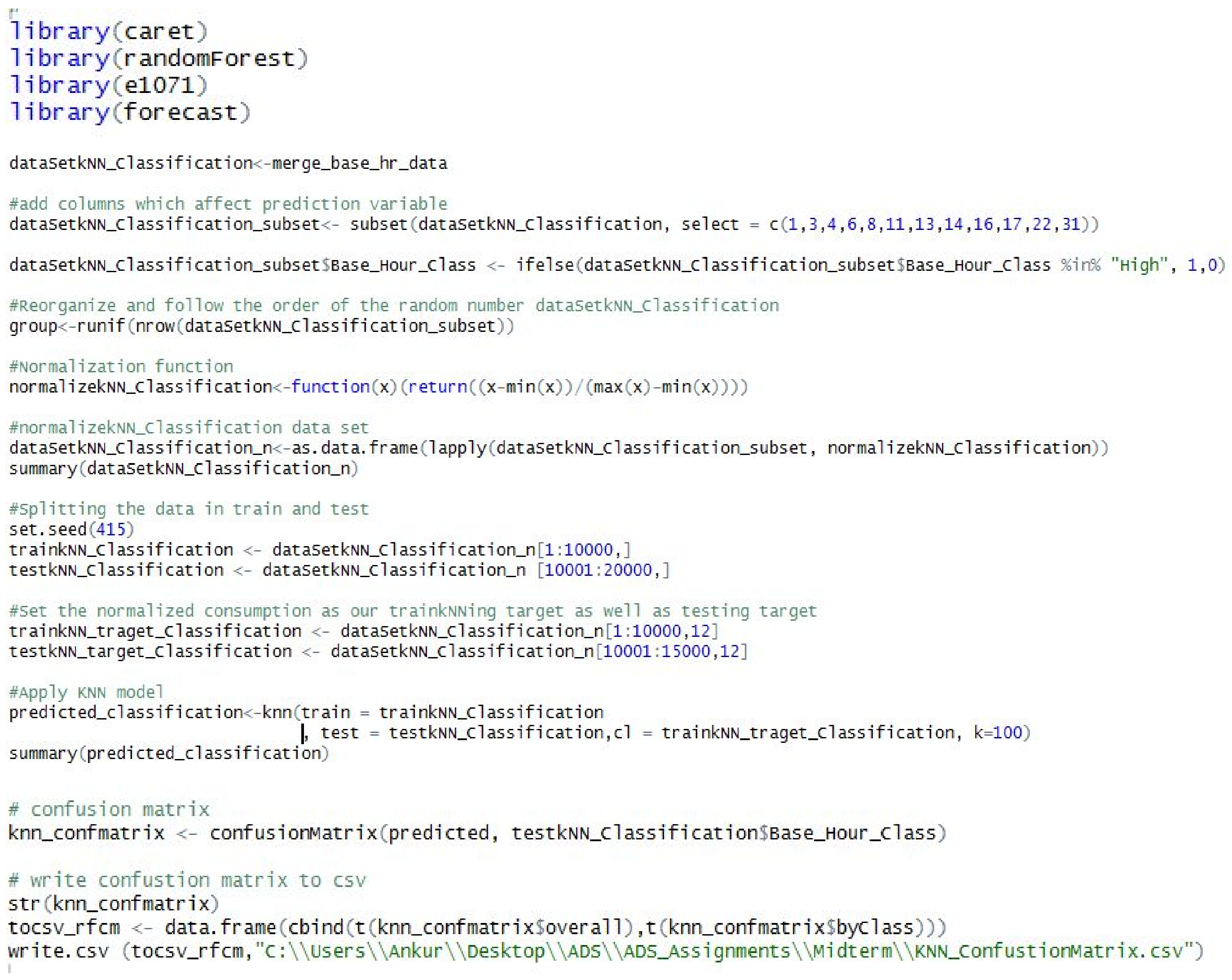
```
# find unique combination of building id and meternumb
unique_comb <- unique(masked1[,c('BuildingID', 'vac', 'meternumb')])
```

2. After getting the list of unique 78 model combination iterate through entire file to get the subset of rows belonging to each model
3. After getting the subset, repeat entire process which is done in prediction & classification i.e
   I. Clean the data if required
   II. Normalize the data if required
   III. Divide the data in training and testing dataset
   IV. Implement KNN/Random Forest/Neural Network
   V. Predict the target value
   VI. Analyze accuracy
   VII. Implement confusion matrix
   VIII. Implement ROC curve
   IX. Implement other graphs
   XI. Write the output to a csv and compare the best model
4. This will be repeated for all the 78 models in a foreach loop using foreach package.

Classification:

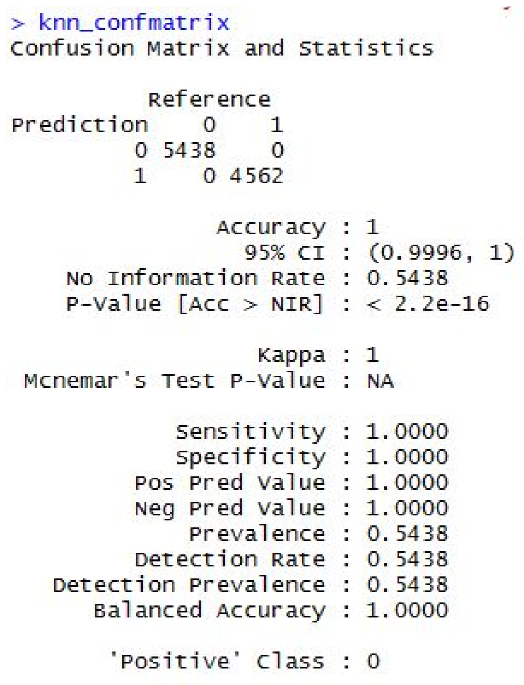
K Nearest Neighbors:

For each data point, the algorithm finds the k closest observations, and then classifies the data point to the majority. If k is even, there might be ties. To avoid this usually weights are given to observations. KNN function is present in class library. kNN requires variables to be normalized or scaled.



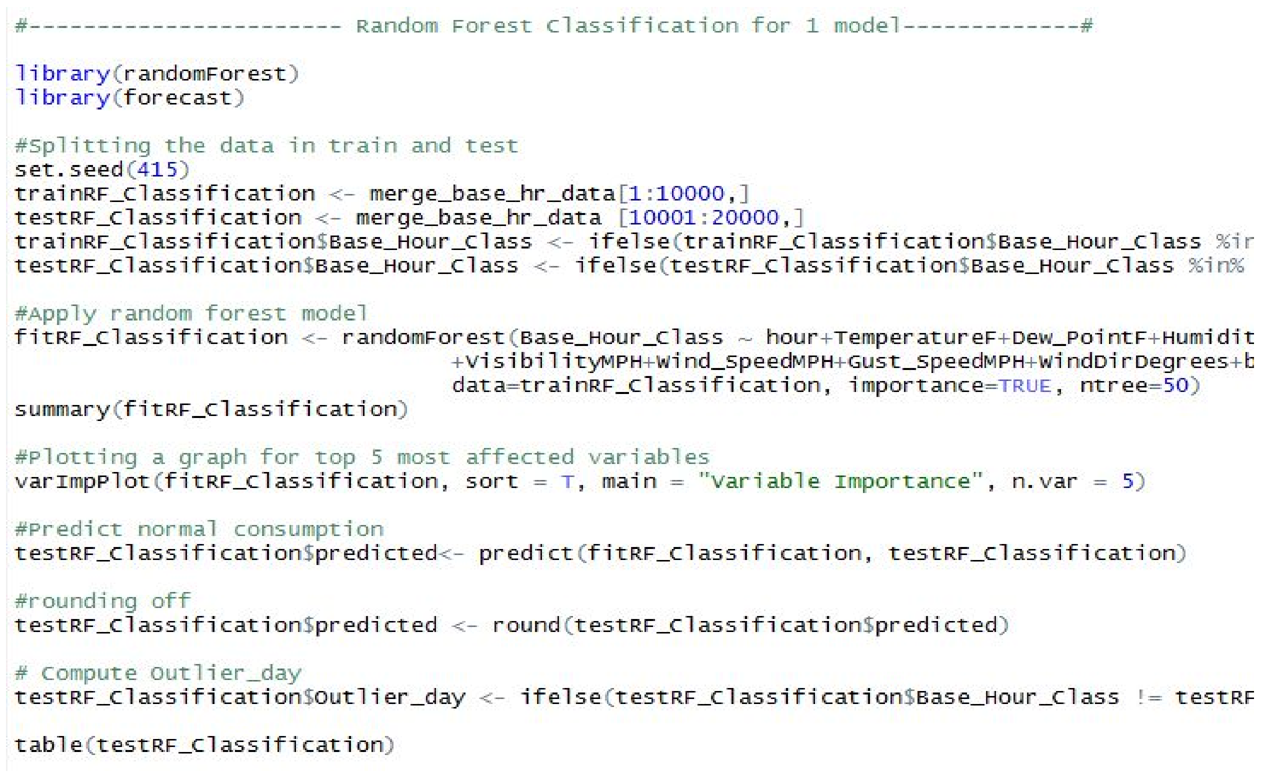
```
library(caret)
library(randomForest)
library(e1071)
library(forecast)

dataSetkNN_Classification<-merge_base_hr_data

#add columns which affect prediction variable
dataSetkNN_Classification_subset<- subset(dataSetkNN_Classification, select = c(1,3,4,6,8,11,13,14,16,17,22,31))

dataSetkNN_Classification_subset$Base_Hour_Class <- ifelse(dataSetkNN_Classification_subset$Base_Hour_Class %in% "High", 1,0)

#Reorganize and follow the order of the random number dataSetkNN_Classification
group<-runif(nrow(dataSetkNN_Classification_subset))

#Normalization function
normalizekNN_Classification<-function(x)(return((x-min(x))/(max(x)-min(x))))

#normalizekNN_Classification data set
dataSetkNN_Classification_n<-as.data.frame(lapply(dataSetkNN_Classification_subset, normalizekNN_Classification))
summary(dataSetkNN_Classification_n)

#Splitting the data in train and test
set.seed(415)
trainkNN_Classification <- dataSetkNN_Classification_n[1:10000,]
testkNN_Classification <- dataSetkNN_Classification_n [10001:20000,]

#Set the normalized consumption as our trainkNNing target as well as testing target
trainkNN_traget_Classification <- dataSetkNN_Classification_n[1:10000,12]
testkNN_target_Classification <- dataSetkNN_Classification_n[10001:15000,12]

#Apply KNN model
predicted_classification<-knn(train = trainkNN_Classification
                                , test = testkNN_Classification,cl = trainkNN_traget_Classification, k=100)
summary(predicted_classification)


# confusion matrix
knn_confmatrix <- confusionMatrix(predicted, testkNN_Classification$Base_Hour_Class)

# write confustion matrix to csv
str(knn_confmatrix)
tocsv_rfcm <- data.frame(cbind(t(knn_confmatrix$overall),t(knn_confmatrix$byClass)))
write.csv (tocsv_rfcm,"C:\\Users\\Ankur\\Desktop\\ADS\\ADS_Assignments\\Midterm\\KNN_ConfustionMatrix.csv")
```

Confusion matrix for KNN determines here

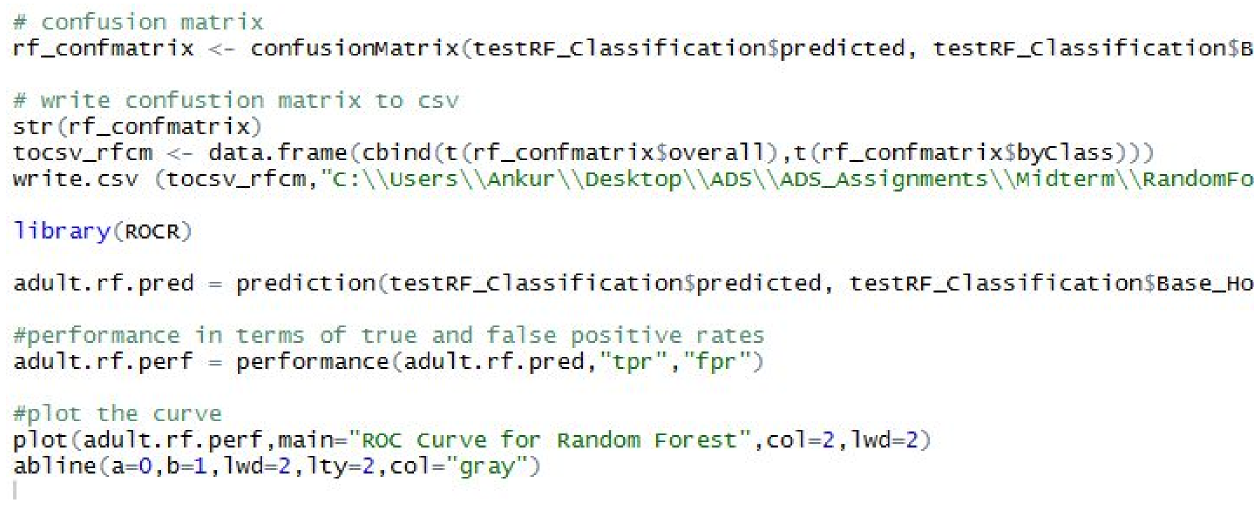


```
> knn_confmatrix
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5438    0
         1    0 4562

               Accuracy : 1
                 95% CI : (0.9996, 1)
    No Information Rate : 0.5438
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5438
         Detection Rate : 0.5438
   Detection Prevalence : 0.5438
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

Random Forest:



```
#----------------------- Random Forest Classification for 1 model-------------#

library(randomForest)
library(forecast)

#Splitting the data in train and test
set.seed(415)
trainRF_Classification <- merge_base_hr_data[1:10000,]
testRF_Classification <- merge_base_hr_data [10001:20000,]
trainRF_Classification$Base_Hour_Class <- ifelse(trainRF_Classification$Base_Hour_Class %ir
testRF_Classification$Base_Hour_Class <- ifelse(testRF_Classification$Base_Hour_Class %in%

#Apply random forest model
fitRF_Classification <- randomForest(Base_Hour_Class ~ hour+TemperatureF+Dew_PointF+Humidit
                                +VisibilityMPH+Wind_SpeedMPH+Gust_SpeedMPH+WindDirDegrees+b
                                data=trainRF_Classification, importance=TRUE, ntree=50)
summary(fitRF_Classification)

#Plotting a graph for top 5 most affected variables
varImpPlot(fitRF_Classification, sort = T, main = "Variable Importance", n.var = 5)

#Predict normal consumption
testRF_Classification$predicted<- predict(fitRF_Classification, testRF_Classification)

#rounding off
testRF_Classification$predicted <- round(testRF_Classification$predicted)

# Compute Outlier_day
testRF_Classification$Outlier_day <- ifelse(testRF_Classification$Base_Hour_Class != testRF

table(testRF_Classification)
```

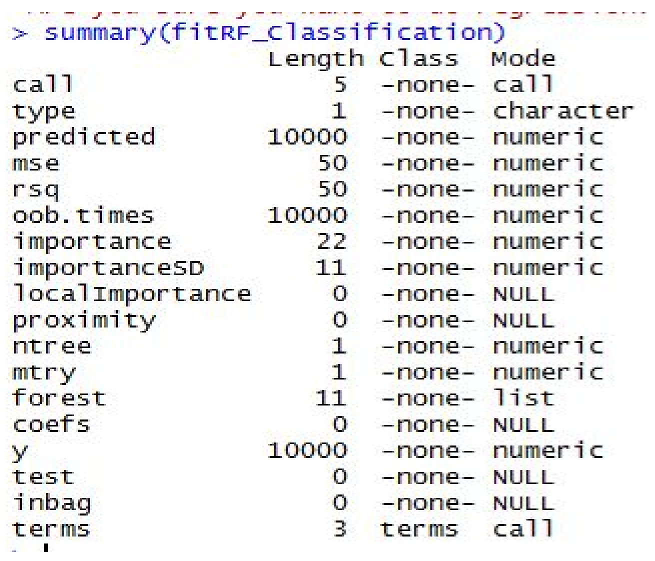


```
# confusion matrix
rf_confmatrix <- confusionMatrix(testRF_Classification$predicted, testRF_Classification$B

# write confustion matrix to csv
str(rf_confmatrix)
tocsv_rfcm <- data.frame(cbind(t(rf_confmatrix$overall),t(rf_confmatrix$byClass)))
write.csv (tocsv_rfcm,"C:\\Users\\Ankur\\Desktop\\ADS\\ADS_Assignments\\Midterm\\RandomFo

library(ROCR)

adult.rf.pred = prediction(testRF_Classification$predicted, testRF_Classification$Base_Ho

#performance in terms of true and false positive rates
adult.rf.perf = performance(adult.rf.pred,"tpr","fpr")

#plot the curve
plot(adult.rf.perf,main="ROC Curve for Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```

In Random Forest Classification,

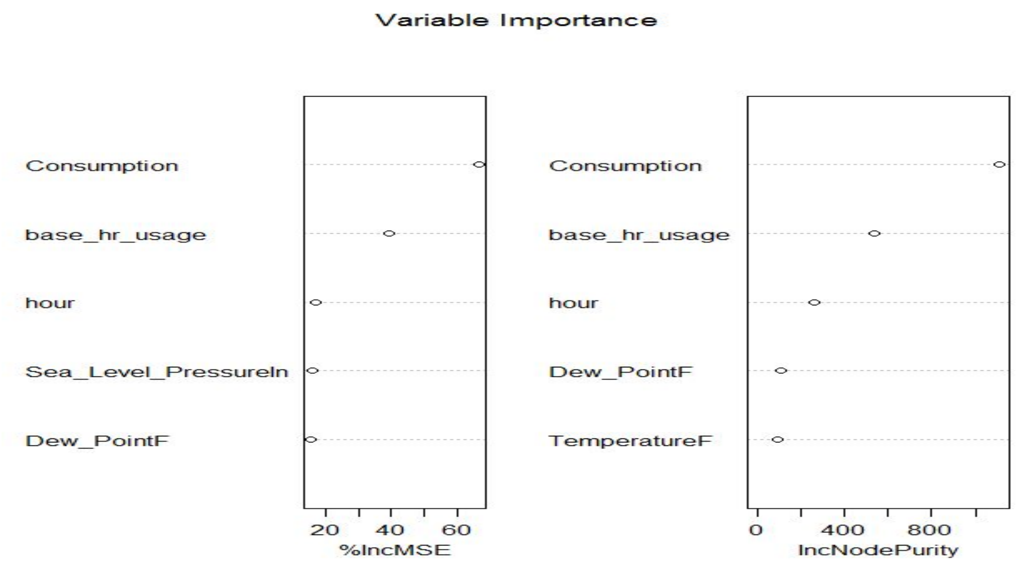
1. Separate the train and test data.
2. Convert the ‘Base-Hour-Class’ values to 0 and 1.
3. Apply Random Forest function selecting the most important variables which were affecting the model majorly.
4. Compute the confusion matrix to determine accuracy
5. Plot ROC Curves.

Summary(Random Forest)

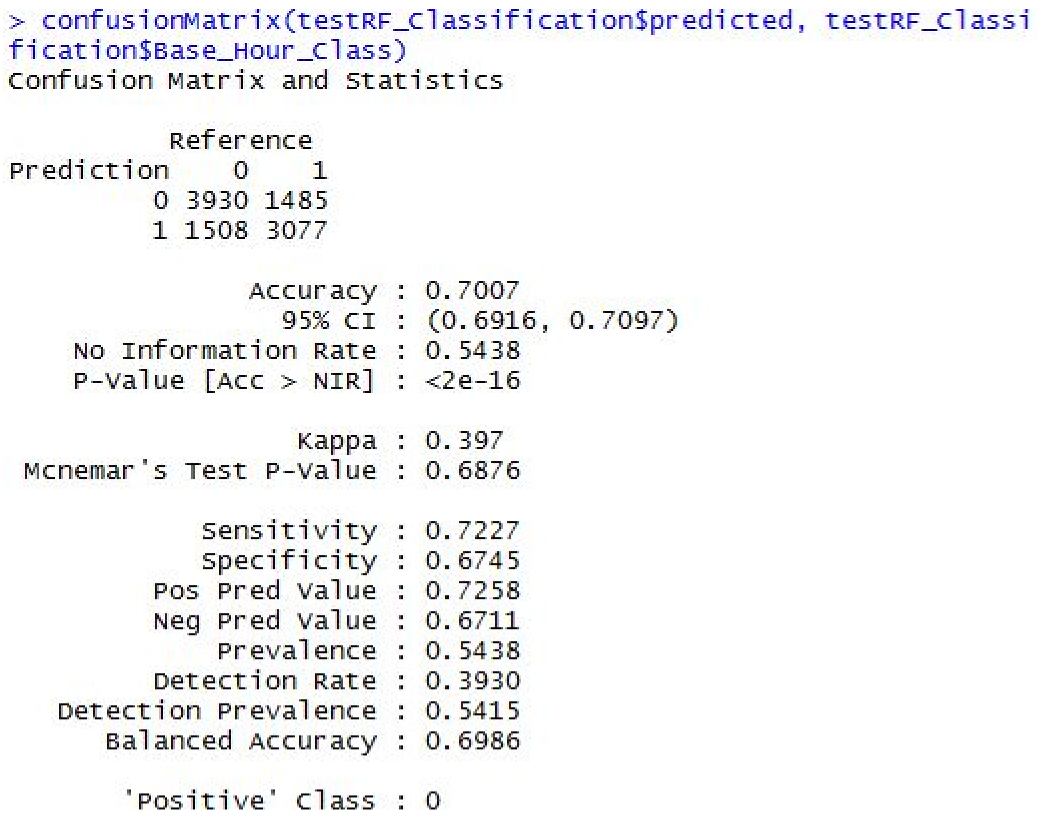


```
> summary(fitRF_Classification)
                Length Class  Mode     
call                5  -none- call     
type                1  -none- character
predicted       10000  -none- numeric  
mse                50  -none- numeric  
rsq                50  -none- numeric  
oob.times       10000  -none- numeric  
importance         22  -none- numeric  
importanceSD       11  -none- numeric  
localImportance     0  -none- NULL     
proximity           0  -none- NULL     
ntree               1  -none- numeric  
mtry                1  -none- numeric  
forest             11  -none- list     
coefs               0  -none- NULL     
y               10000  -none- numeric  
test                0  -none- NULL     
inbag               0  -none- NULL     
terms               3  terms  call     
```

Variable Importance(Random Forest Classification)



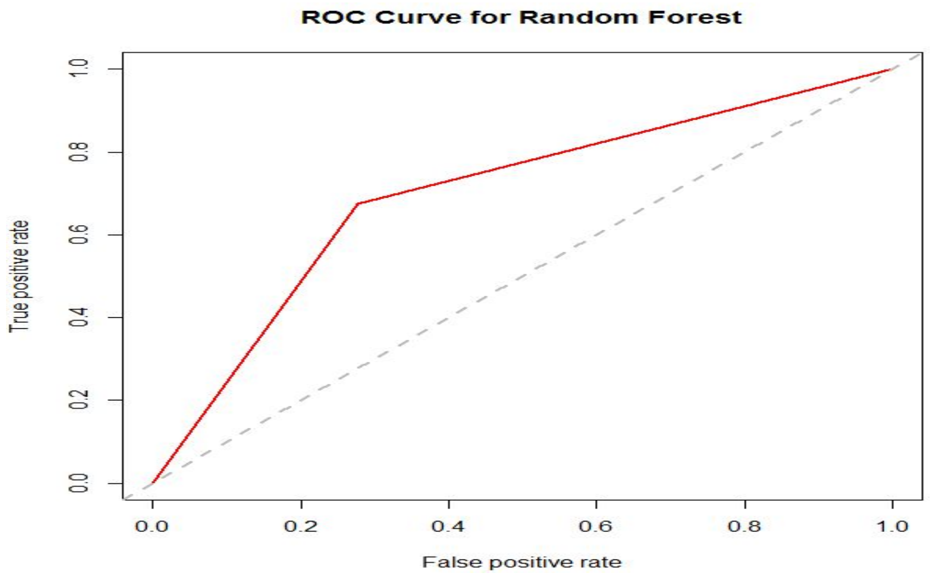
Confusion Matrix:



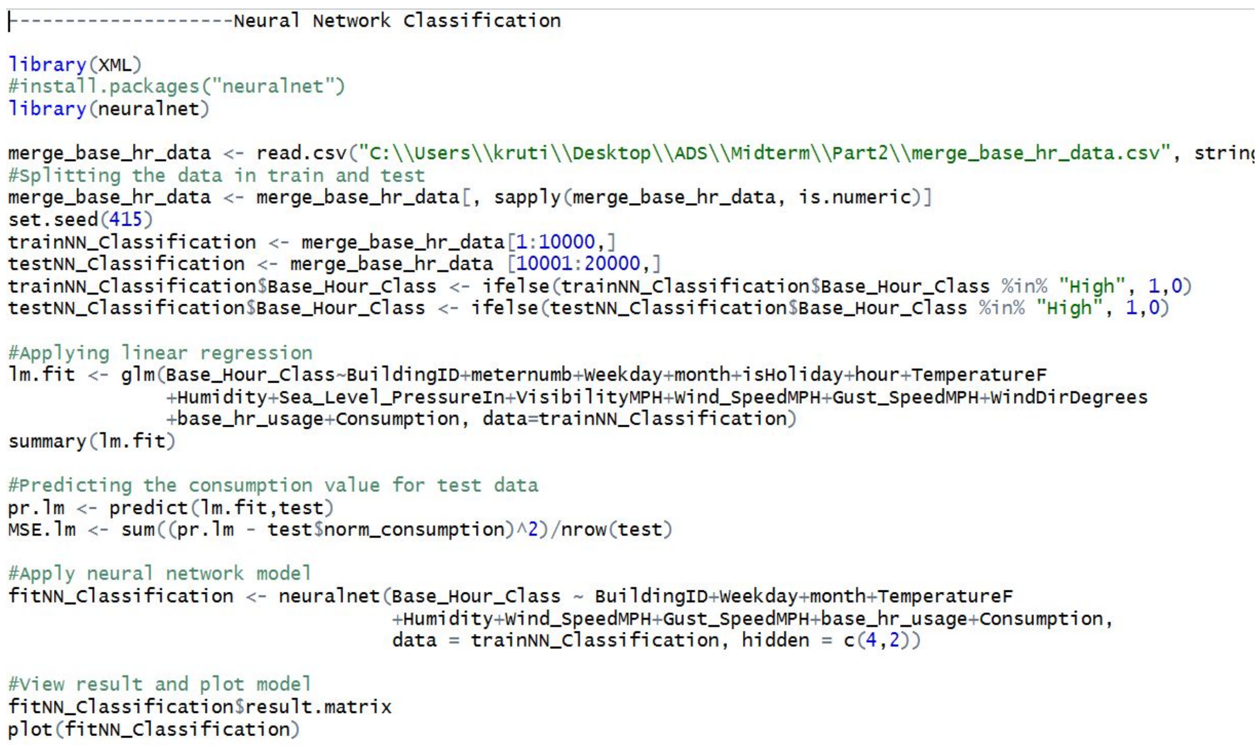
```
> confusionMatrix(testRF_Classification$predicted, testRF_Classi
fication$Base_Hour_Class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3930 1485
         1 1508 3077

               Accuracy : 0.7007
                 95% CI : (0.6916, 0.7097)
    No Information Rate : 0.5438
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.397
 Mcnemar's Test P-Value : 0.6876

            Sensitivity : 0.7227
            Specificity : 0.6745
         Pos Pred Value : 0.7258
         Neg Pred Value : 0.6711
             Prevalence : 0.5438
         Detection Rate : 0.3930
   Detection Prevalence : 0.5415
      Balanced Accuracy : 0.6986

       'Positive' Class : 0
```

Here, we are getting accuracy as 0.7 which is good enough.

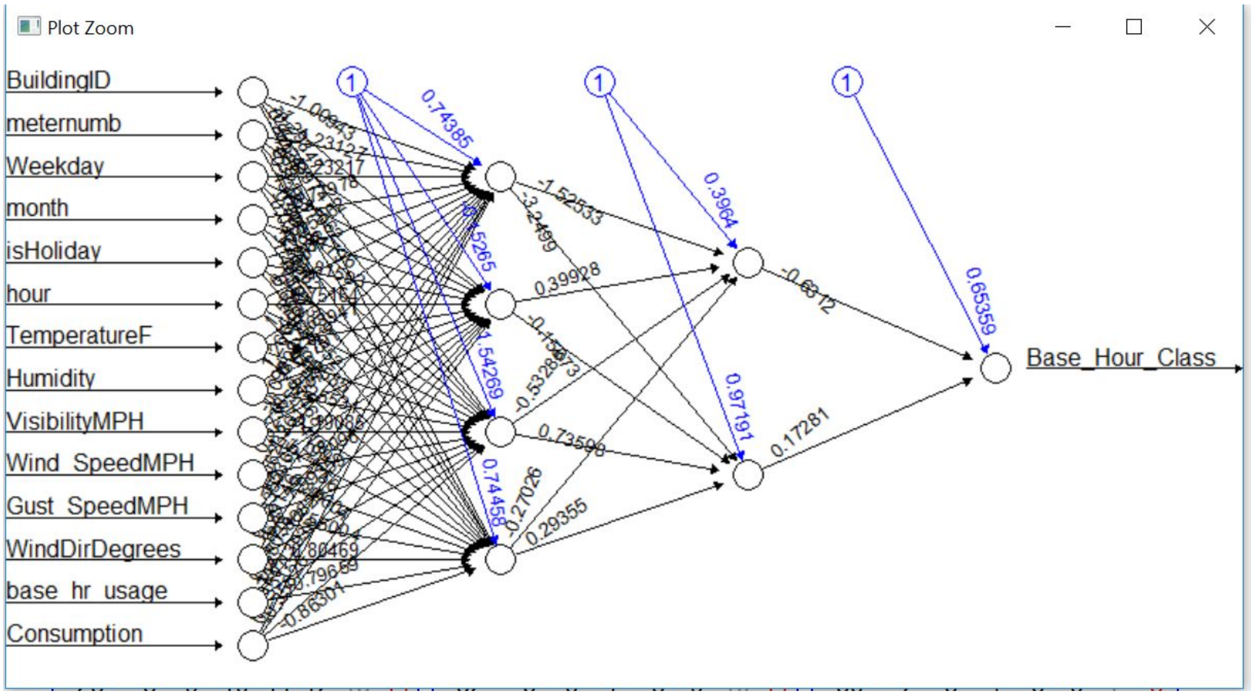
ROC Curve:



Neural Network :



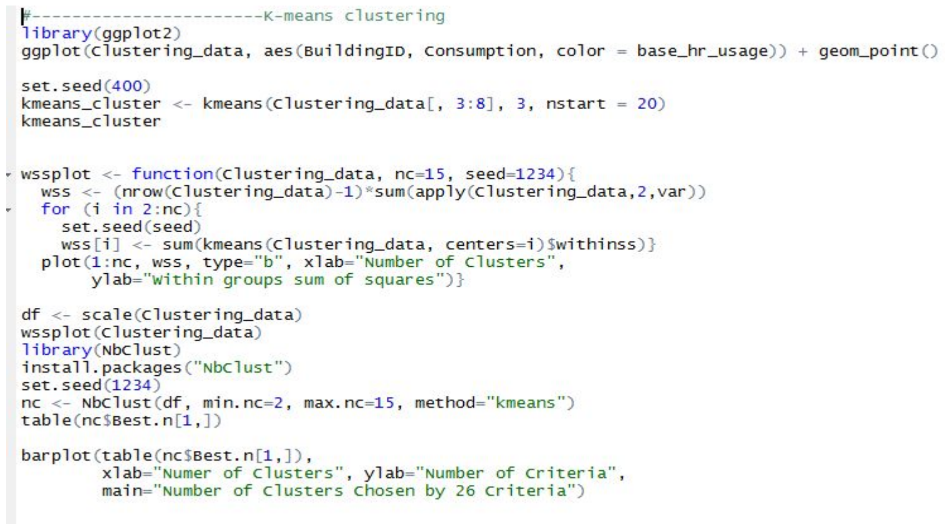
```
|--------------------Neural Network Classification

library(XML)
#install.packages("neuralnet")
library(neuralnet)

merge_base_hr_data <- read.csv("C:\\Users\\kruti\\Desktop\\ADS\\Midterm\\Part2\\merge_base_hr_data.csv", strin
#Splitting the data in train and test
merge_base_hr_data <- merge_base_hr_data[, sapply(merge_base_hr_data, is.numeric)]
set.seed(415)
trainNN_Classification <- merge_base_hr_data[1:10000,]
testNN_Classification <- merge_base_hr_data [10001:20000,]
trainNN_Classification$Base_Hour_Class <- ifelse(trainNN_Classification$Base_Hour_Class %in% "High", 1,0)
testNN_Classification$Base_Hour_Class <- ifelse(testNN_Classification$Base_Hour_Class %in% "High", 1,0)

#Applying linear regression
lm.fit <- glm(Base_Hour_Class~BuildingID+meternumb+Weekday+month+isHoliday+hour+TemperatureF
              +Humidity+Sea_Level_PressureIn+VisibilityMPH+Wind_SpeedMPH+Gust_SpeedMPH+WindDirDegrees
              +base_hr_usage+Consumption, data=trainNN_Classification)
summary(lm.fit)

#Predicting the consumption value for test data
pr.lm <- predict(lm.fit,test)
MSE.lm <- sum((pr.lm - test$norm_consumption)^2)/nrow(test)

#Apply neural network model
fitNN_Classification <- neuralnet(Base_Hour_Class ~ BuildingID+Weekday+month+TemperatureF
                                  +Humidity+Wind_SpeedMPH+Gust_SpeedMPH+base_hr_usage+Consumption,
                                  data = trainNN_Classification, hidden = c(4,2))

#View result and plot model
fitNN_Classification$result.matrix
plot(fitNN_Classification)
```



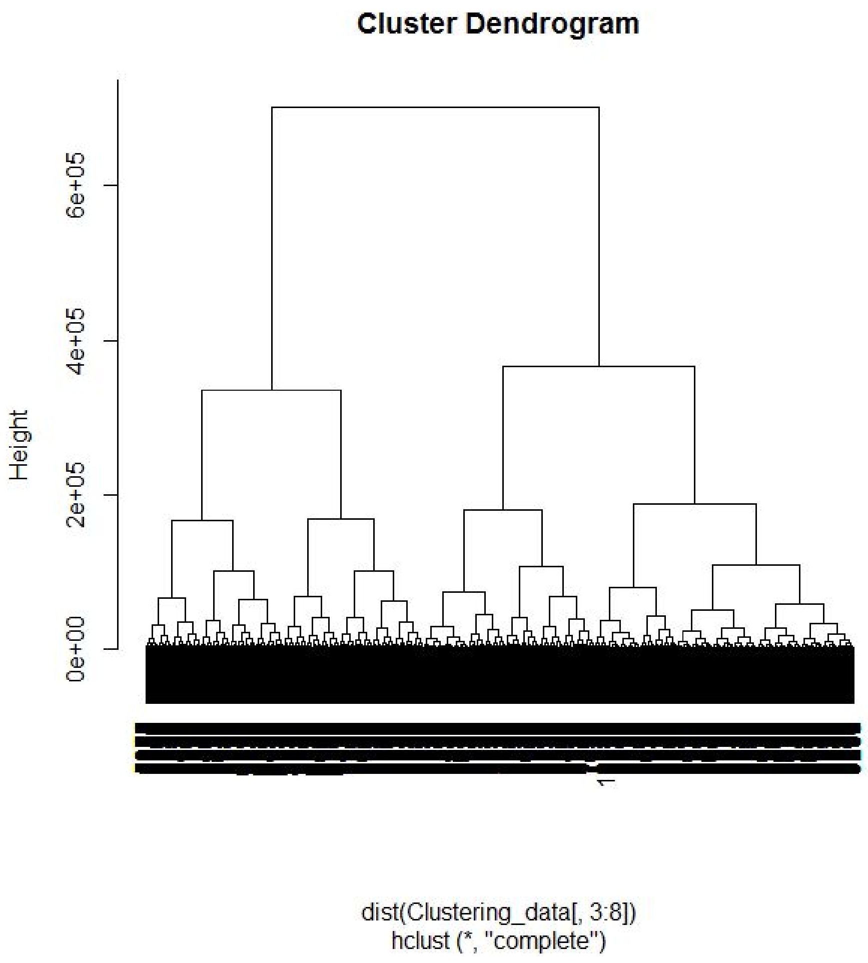
We found drawbacks for Neural Network like the model is complex and difficult to understand. We cannot interpret the best results from it. Also it does not have any direct function to check for its accuracy.

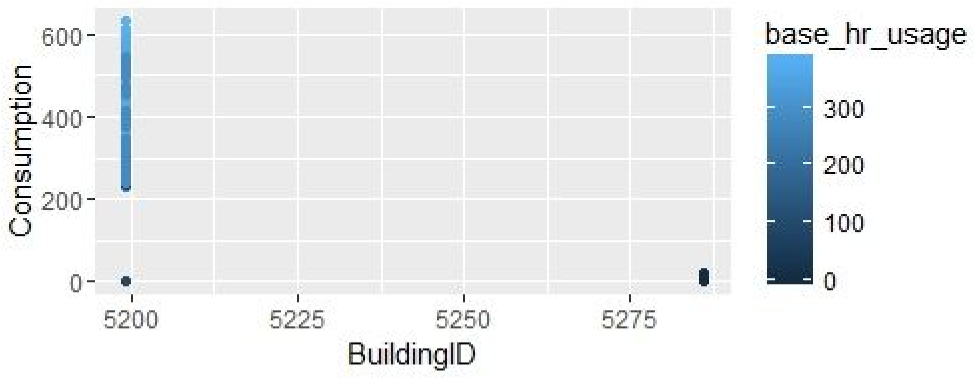
CLUSTERING

K-means and Hierarchical Clustering :



```
#-----------------------K-means clustering
library(ggplot2)
ggplot(Clustering_data, aes(BuildingID, Consumption, color = base_hr_usage)) + geom_point()

set.seed(400)
kmeans_cluster <- kmeans(Clustering_data[, 3:8], 3, nstart = 20)
kmeans_cluster


wssplot <- function(Clustering_data, nc=15, seed=1234){
  wss <- (nrow(Clustering_data)-1)*sum(apply(Clustering_data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(Clustering_data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

df <- scale(Clustering_data)
wssplot(Clustering_data)
library(NbClust)
install.packages("NbClust")
set.seed(1234)
nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
table(nc$Best.n[1,])

barplot(table(nc$Best.n[1,]),
        xlab="Numer of Clusters", ylab="Number of Criteria",
        main="Number of Clusters Chosen by 26 Criteria")
```





Unlike hierarchical clustering, K-means clustering requires that the number of clusters to extract be specified in advance. Since K-means cluster analysis starts with k randomly chosen centroids, a different solution can be obtained each time the function is invoked. The kmeans() function has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart=25 will generate 25 initial configurations. This approach is often recommended.

REFERENCES

Part 1:

1. Extract latitude longitude from address : http://stackoverflow.com/questions/20937682/r-trying-to-find-latitude-longitude-data-for-cities-in-europe-and-getting-geocode
2. Nearest airport : http://stackoverflow.com/questions/39909142/r-weather-underground-how-to-use-longitude-and-latitude-to-get-the-closest-a

Part 2:

1. https://www.youtube.com/watch?v=lTMqXSSjCvk&t=18s (Neural Network)
2. https://www.r-bloggers.com/using-knn-classifier-to-predict-whether-the-price-of-stock-will-increase/ (knn Prediction & classification)
3. http://rstudio-pubs-static.s3.amazonaws.com/16444_caf85a306d564eb490eebdbaf0072df2.html (knn Prediction & classification)
4. https://www.r-bloggers.com/r-code-example-for-neural-networks/
5. https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/
6. http://www.kdnuggets.com/2016/08/begineers-guide-neural-networks-r.html/2
7. https://www.r-bloggers.com/predicting-wine-quality-using-random-forests/
8. http://www.bios.unc.edu/~dzeng/BIOS740/randomforest.pdf

THANK YOU