Midterm Project

Advances in Data Science

Summary

You have been offered an internship at Tarja and Pasi Inc. in Finland and you are excited to work on your first “international” project. Tarja and Pasi Inc. is an energy modeling consultancy and they have been approached by a Vokia Inc. to help monitor and reduce energy consumption in 78 of their buildings. Vokia Inc. owns these buildings and wants to understand and reduce energy usage and wants to make the buildings more energy efficient.

Data:

You have been given two input files.

RawData.csv

BuildingID	Building	Meter number	type	date	hour	Consumption (in KwH)

- Note that the data covers 78 buildings and has hourly power consumption info for close to 1 year
- Use only the **elect and dist_heat data** which covers electricity usage and heating energy consumption data

building	address	area_floor _m.sqr(in square meter)

- This data has the building address (of course in Finland) and the area of the building

Table of Contents

DATA INGESTION & WRANGLING

- The Data set contains 12 million entries which is hard to read as a csv file.
- So , Initially the data is read as data table and atomic variable to increase speed of processes that we do on the data.
- It is then converted to a data frame and retained only electric and dist_heating .
- Data set contains some missing data in the columns of VAC which is filled by Building names (9 & 27).
- Merge the address details along with the area per sq. unit to our data.
- To do any sort of data manipulation in a data table , we use the keyword 'setkey(datatablename,columnname)'
- Clean the data by
    - converting date column from int to date format
    - add year month and day column
    - get day of the week , weekday , peak hour

```
# Read the data set as data table and atomic variable for large dataset
mydata<-fread("Finland_addresses_area.csv")
mydata1<-fread("Finland_masked.csv")

#convert it to dplyr data frame
dplyr::tbl_df(mydata)
dplyr::tbl_df(mydata1)

# transpose meter number to columns
setkey(mydata1,type)
abc1<-mydata1[c("elect","Dist_Heating")]
#abc1[,.N,by=type]

##############################################################################utils::View(abc2)
#abc<-dcast(mydata1,BuildingID+vac+date+hour+meternumb~type,value.var = "Consumption",sum)
# melt it the meter no. back to rows
#abc1<- abc[,.(BuildingID,vac,date,hour,meternumb,Dist_Heating,elect)]
###########################################################################################
#assign building 9 and 27
abc1[, vac := ifelse(BuildingID==81909, "Building 27", vac)]
abc1[, vac := ifelse(BuildingID==82254|BuildingID==83427|BuildingID==84681, "Building 9", vac)]
#remove blank
abc2<-abc1[!(vac=="                                   ")]
names(abc2)[2]<-"building"

#merge the address and area data with your data
setkey(abc2,building)
setkey(mydata,building)
Data_merge <- merge(abc2,mydata, all.x=TRUE)
```

```
#convert date column from int to date format
D <- transform(Data_merge, date = as.Date(as.character(date), "%Y%m%d"))
##...........................................................................utils::View(Data_merge)
#add year month and day column
E<-tidyr::separate(D, date, c("year","month","day"))
#get date column
E$Date <- as.Date(with(E, paste(year, month, day,sep="/")), "%Y/%m/%d")#add Date column
# get Day of week
E$DayofWeek<-as.POSIXlt(E$Date)$wday
#check for weekday
E$WeekDay<-ifelse(E$DayofWeek > 0 & E$DayOfWeek < 6, "1", "0")
#peak hour column
E1 <- E %>% mutate(hour = as.numeric(hour))
E1$Peakhour<-ifelse(E1$hour > 6 & E1$hour < 20, "1", "0")
# convert day and month to numeric column
E1$month<-as.numeric(E1$month);E1$day<-as.numeric(E1$day)

E0<-as.data.table(E1)
E0[,.N,by=Building12]
```

To get list of holidays in a calendar year of 2013

- We use library(rvest) to scrape the data from the HTML page using the CSS Node
- Merge the holiday data and find the days which are holidays in our Building data .

```
#get holidays from webpage
htmlpage<-read_html("http://www.timeanddate.com/calendar/?year=2013&country=24")
#scrape CSS
holiday<-html_nodes(htmlpage,"#ch1 .lpad td:nth-child(1) , #ch1 .co1")
#get the text out of node
holiday1<-html_text(holiday)
#convert text to table
holiday2<-read.table(text = holiday1, sep = ".", colClasses = "character")
#name the column
names(holiday2)[1]<-"day"
names(holiday2)[2]<-"mont"
#translate finnish to numeric month
fin_month = matrix(c(" tam"," maa"," huh"," tou"," kes"," mar"," jou",1,3,4,5,6,11,12),nrow=7,ncol=2)
fin_month<-as.data.frame(as.matrix(fin_month))
#name the columns
names(fin_month)[1]<-"mont"
names(fin_month)[2]<-"month"
fin_month$mont<-as.character(fin_month$mont)
fin_month$month<-as.character(fin_month$month)
#convert month to number
holidays<-merge(holiday2,fin_month,by=c("mont"))
holidays$mont<-1
holidays<-as.data.frame(holidays)
#make the day and month column numeric
holidays$month<-as.numeric(holidays$month);holidays$day<-as.numeric(holidays$day)
holidays<-holidays[!duplicated(holidays[,c('day','month')]),]
#merge the holdays data and find which days are a holiday
E2<-E1
E2<- merge(E2,holidays,by=c("day","month"), all.x=TRUE,all.y = FALSE)
E2[is.na(E2)] <- 0
#final data of PART 1.1
View(E2)
```

To look up for weather station

- We get the geocode for the address corresponding to building.(Get the value of latitude & longitude)
- Using latitude and longitude values , we get the nearest airport code.
- For distinct airport codes , we scrape the weather data for (365 - 23)days and then we merge the weather data with complete building data with keywords sta_code , for a particular date , hour.
    - To Get the weather data
        - Create a date sequence from Jan 1st 2013 to Dec 8th 2013
        - Get the number of distinct airport
        - Create a for loop to scrape the contents of HTML from the wunderground.com from the created time sequence.
        - Store in weather.csv
    - Since there are 33 buildings we traverse the loop that many number of times get the weather data corresponding to the building.
    - We use xmlTreeParse (Parses an XML or HTML file or string containing XML/HTML content, and generates an R structure representing the XML/HTML tree) and xpathApply (create an R object and pass it) , which in our case is xmlValue .

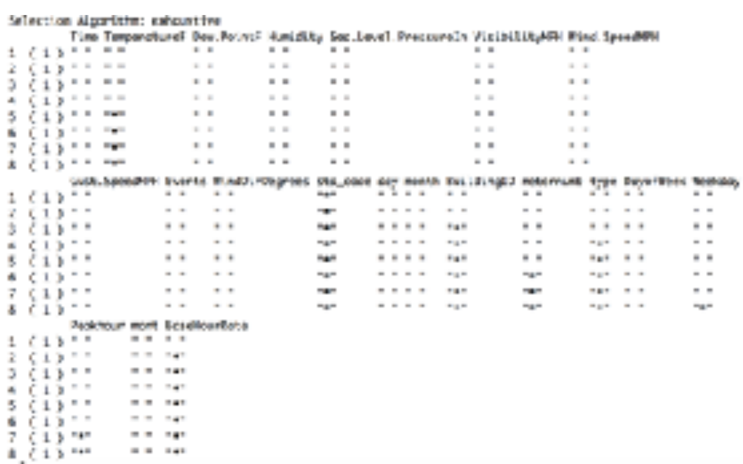
```
#get the date range
date <-  as.Date('20130101',format = "%Y%m%d")
date2 <-  as.Date('20131208',format = "%Y%m%d")
# create a sequence of every day in this year
s <- seq(date,to = date2, by='days')
#modify the date format
datereplace<-str_replace_all(s,"-","/")
#get the dist number of airport
n=nrow(distinct_airport)
#scrape the html from URL
weathers<-data.frame(Time=as.character())
for(j in 1:n)
{
  code<-distinct_airport[j,1]
  dat<-paste0("https://www.wunderground.com/history/airport/",code,"/2013/1/1/DailyHistory.html?req_city=Mariehamn&req_statename=Finland&format=1")
  htmlpage<-read_html(dat)
  #get the text out of HTML
  weathe<-html_text(htmlpage)
  #make it a table
  weath<-read.table(text = weathe, sep = ",", colClasses = "character")
  #remove extra row with heading
  colnames(weath)=weath[1,]
  weath = weath[-1, ]
  #name 1st column as TIME
  names(weath)[1]<-"Time"
  #convert it to data frame
  weather<-data.frame(weath)
  #add a column ad date
  weather<- weather %>% rowwise() %>% mutate(date = "2013/1/1")
  #get the dist number of airport
  n=nrow(distinct_airport)
  #for loop to scrape the weather data of a particular airport code frm Jan 1 2013 to Dec 8 2013
(Top Level) :

  for (i in 2:342)
  {
    data<-paste0("https://www.wunderground.com/history/airport/",code,"/",datereplace[i],"/DailyHistory.html?req_city=Mariehamn&req_statename=Finland&format=1")
    htmlpage<-read_html(data)
    weathe<-html_text(htmlpage)
    weath<-read.table(text = weathe, sep = ",", colClasses = "character")
    colnames(weath)=weath[1,]
    weath = weath[-1, ]
    names(weath)[1]<-"Time"
    weather1<-data.frame(weath)
    dat = datereplace[i]
    weather1<-weather1 %>% rowwise() %>% mutate(date=dat)
    weather1<-weather1 %>% rowwise() %>% mutate(sid_code = code)


    dplyr::tbl_df(weather)
    dplyr::tbl_df(weather1)
    weather<-dplyr::bind_rows(weather, weather1)
  }
  weathers<-dplyr::bind_rows(weathers, weather)

}
utils::View(weathers)
write.csv(weather,file="weather.csv")
```

Modeling Tasks

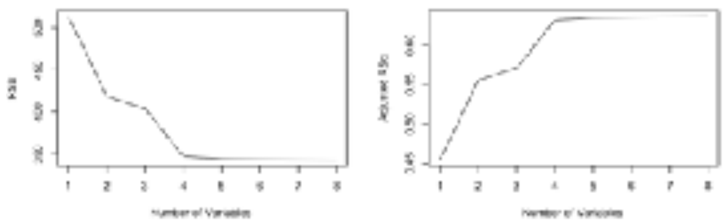
Prediction

Exhaustive Search Regression

```r
#regression
#Exhaustive Search
regfit.full=regsubsets (Normalised~.,data=trainingData ,really.big = TRUE)
exhaustiveRegSummary =summary(regfit.full)
print(exhaustiveRegSummary)
par(mfrow=c(2,2))
plot(exhaustiveRegSummary$rss ,xlab="Number of Variables ",ylab="RSS", type="l")
plot(exhaustiveRegSummary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
```



- Looking at the selection algorithm we can come to a conclusion that 5 independent variables have major influence on the Base Hour Usage (Variable name : Normalised)
- The below image confirms our conclusion.



Step-wise Search Regression

```
#Forward Search
regfit.fwd=regsubsets(Normalised~.,data=trainingData, method="forward")
forwardSearchSummary=summary(regfit.fwd)
names(forwardSearchSummary)
forwardSearchSummary$rss
forwardSearchSummary$adjr2
coef(regfit.fwd,5)

par(mfrow=c(2,2))
plot(forwardSearchSummary$rss ,xlab="Number of Varidbles ",ylab="RSS", type="l")
plot(forwardSearchSummary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
```

- We get similar results for stepwise regression . Therefore , we will build the linear regression model with the lowest 'p' value.

Linear Regression for the chosen independent variables

```
#Linear Regression

lm.fit = lm(Normalised ~., data = trainingData)
summary(lm.fit)
require(forecast)
accuracy(lm.fit)
L<-predict(lm.fit,testData)
accuracy(L,trainingData$Normalised)
```

```
Analysis of Variance Table

Response: Normalised
                        Df Sum Sq Mean Sq    F value    Pr(>F)
Time                     1   0.00    0.00 2.1571e+00    0.1419
TemperatureF             1  14.87   14.87 1.8871e+04 < 2.2e-16 ***
Dew.PointF               1   1.53    1.53 1.9414e+03 < 2.2e-16 ***
Humidity                 1   0.83    0.83 1.0517e+03 < 2.2e-16 ***
Sea.Level.PressureIn     1   0.00    0.00 5.2600e-02    0.8185
VisibilityMPH            1  72.10   72.10 9.1525e+04 < 2.2e-16 ***
Wind.SpeedMPH            1   2.92    2.92 3.7096e+03 < 2.2e-16 ***
Gust.SpeedMPH            1   2.27    2.27 2.8850e+03 < 2.2e-16 ***
Events                   1   0.76    0.76 9.6450e+02 < 2.2e-16 ***
WindDirDegrees           1   0.73    0.73 9.2141e+02 < 2.2e-16 ***
sta_code                 1 347.22  347.22 4.4077e+05 < 2.2e-16 ***
day                      1   0.04    0.04 5.2783e+01 3.731e-13 ***
month                    1   0.19    0.19 2.4186e+02 < 2.2e-16 ***
BuildingID               1  19.81   19.81 2.5149e+04 < 2.2e-16 ***
meternumb                1   0.23    0.23 2.8945e+02 < 2.2e-16 ***
type                     1  63.05   63.05 8.0036e+04 < 2.2e-16 ***
DayofWeek                1   0.00    0.00 1.9500e-02    0.8889
Weekday                  1   1.55    1.55 1.9651e+03 < 2.2e-16 ***
Peakhour                 1   2.85    2.85 3.6126e+03 < 2.2e-16 ***
mont                     1   0.04    0.04 4.6155e+01 1.094e-11 ***
BaseHourRate             1  67.15   67.15 8.5237e+04 < 2.2e-16 ***
Residuals           435249 342.88    0.00
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> accuracy(lm.fit)
                        ME       RMSE        MAE      MPE     MAPE      MASE
Training set -2.017104e-19 0.02806659 0.01402648 -67.1175 150.0256 0.6894058
```

- The RMSE value is 2% , which indicates the absolute fit of the model to the data and shows how accurate our dependent variable of train data set is equal to the prediction of our train data set.

KNN Regression

```
#KNN REGRESSION
sqrt(24)
trainingData1<-as.matrix(trainingData)
testData1<-as.matrix(testData)
m1<-knn(train = trainingData,test=testData,cl=train_target,k=5)


table(m1,trainingData$Normalised)
mean(test_target==m1)

summary(m1)
```

Random Forest Regression

```
####################Random forest########################

install.packages("h2o")
library(h2o)
localH2O <- h2o.init(nthreads = -1)
h2o.init()
train.h2o <- as.h2o(trainingData)
test.h2o <- as.h2o(testData)
y.dep <- 21
x.indep <- c(1:20)

system.time(
  rforest.model <- h2o.randomForest(y=y.dep, x=x.indep, training_frame =
                   train.h2o, ntrees = 1000, mtries = 3, max_depth = 4, seed = 1122)
)

td1_normalise<-as.data.frame(testData[[21]])
x2<-data.frame(sub_rf,td1_normalise)
plot(x2)

#check variable importance
h2o.varimp(rforest.model)

#making predictions on unseen data
system.time(predict.rforest <- as.data.frame(h2o.predict(rforest.model, test.h2o)))
sub_rf <- data.frame(BaseHourRate =  predict.rforest$predict)
```

- Running random forest model in R threw an error a "memory heap error"
- To optimize it , we used the library(h2o) which uses the JVM to execute the code ,hence removing the memory issue.

Variable Importances:

	variable	relative_importance	scaled_importance	percentage
1	sta_code	139496.281250	1.000000	0.450595
2	BuildingID	95717.132812	0.686163	0.309182
3	Gust.SpeedMPH	27245.988281	0.195317	0.088009
4	VisibilityMPH	25553.988281	0.183188	0.082543
5	TemperatureF	3666.728027	0.026285	0.011844
6	meternumb	3629.106934	0.026016	0.011723
7	Dew.PointF	3500.954834	0.025097	0.011309
8	type	3336.192871	0.023916	0.010776
9	month	2840.497803	0.020363	0.009175
10	Events	1261.291748	0.009042	0.004074
11	Wind.SpeedMPH	874.869751	0.006272	0.002826
12	Sea.Level.PressureIn	768.606689	0.005510	0.002483
13	DayofWeek	332.196564	0.002381	0.001073
14	Weekday	321.923981	0.002308	0.001040
15	Peakhour	318.687805	0.002285	0.001029
16	Humidity	251.027344	0.001800	0.000811
17	WindDirDegrees	202.399017	0.001451	0.000654
18	Time	112.111389	0.000804	0.000362
19	day	91.192726	0.000654	0.000295
20	mont	61.033161	0.000438	0.000197

>

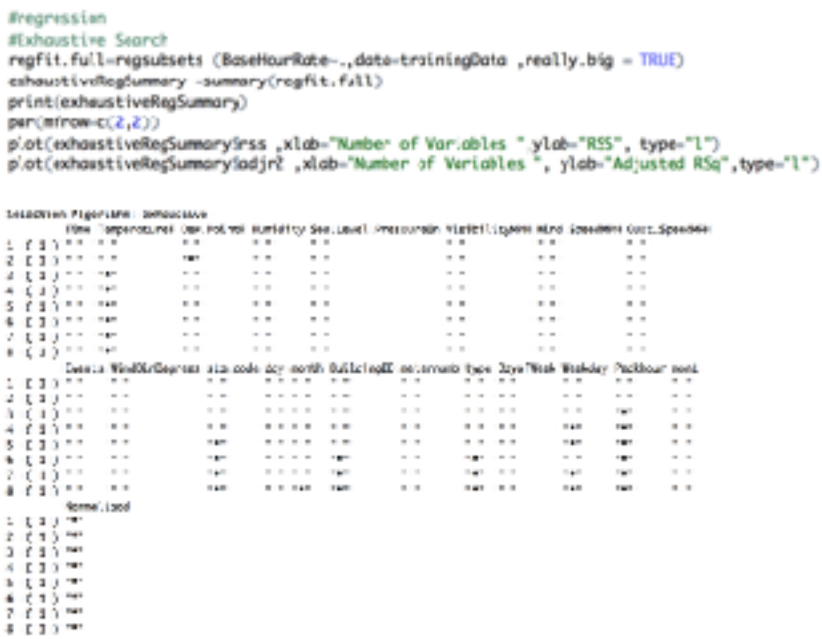
- From the linear regression model , we know that we need 5 variables for the best results. So we consider the top 5 variables from the above figure for our model.
- It is constant with the variables deemed fit by linear regression model.

Conclusion for Prediction

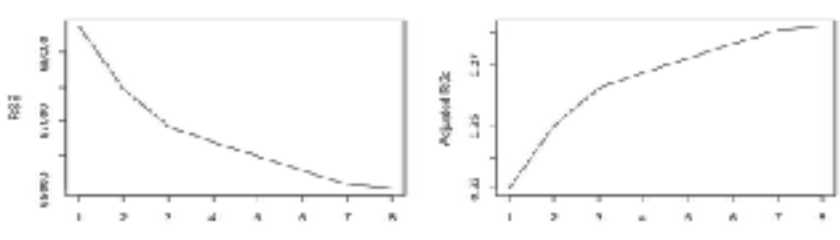
For regression model , linear regression using lm.fit and random forest will give accurate models as compared to k nearest neighbor . The neural network as seen below can be difficult to interpret but with forecast it might yield better results.

Classification

Exhaustive Search Regression


```
#regression
#Exhaustive Search
regfit.full=regsubsets (BaseHourRate~.,data=trainingData ,really.big = TRUE)
exhaustiveRegSummary =summary(regfit.full)
print(exhaustiveRegSummary)
par(mfrow=c(2,2))
plot(exhaustiveRegSummary$rss ,xlab="Number of Variables ",ylab="RSS", type="l")
plot(exhaustiveRegSummary$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
```

- Looking at the selection algorithm we can come to a conclusion that 8 independent variables have major influence on the Base Hour Class (Variable name : BaseHourRate)
- The below image confirms our conclusion.



- Forward Search Classification rendered the same result.

Logistic Regression

```
#Logistic Regression

glm.fit = glm(BaseHourRate ~., data = trainingData,family = "binomial")
summary(glm.fit)
accuracy(glm.fit)
L<-predict(glm.fit,testData)
accuracy(L,trainingData$Normalised)
```

```
Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          3.767e-06  9.968e-04   3.017 8.24e-06 ***
Time                 2.438e-03  1.543e+04   0.158    0.875
TemperatureF        -1.679e-04  1.677e+05  -0.156    0.876
Dew PointF           1.609e-04  1.148e+05   0.140    0.889
Humidity            -4.095e-03  4.288e+04  -0.096    0.924
Sea.Level.PressureIn -9.849e-04  6.377e+04  -1.544    0.123
VisibilityMPH       -3.345e-03  7.513e+04  -0.045    0.964
Wind.SpeedMPH        7.772e-03  3.129e+04   0.248    0.804
Gust.SpeedMPH       -2.575e-03  2.720e+04  -0.095    0.925
Events              -3.127e-03  6.752e+04  -0.046    0.963
WindDirDegrees      -2.477e-02  1.088e+03  -0.228    0.820
sta_code            -1.187e-09  2.535e+08  -4.045 5.23e-05 ***
day                 -2.741e-03  1.181e+04  -0.232    0.816
month                6.150e-03  3.589e+04   0.172    0.864
Building.ID         -1.479e-01  1.223e+01  -1.204    0.228
meternumb            1.215e-07  1.709e+07   0.711    0.477
type                -7.330e-05  6.204e+05  -0.890    0.373
DayofWeek           -4.118e-03  5.231e+04  -0.078    0.937
Weekday             -2.027e-04  2.320e+05  -0.104    0.917
Peakhour             2.898e-04  2.618e+05   0.110    0.912
mont                 1.314e-06  8.510e+06   0.154    0.877
Normalised           6.703e-05  1.234e+05   5.432 5.59e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5.2843e+05  on 435270  degrees of freedom
Residual deviance: 6.1757e-06  on 435249  degrees of freedom
AIC: 44.618

Number of Fisher Scoring iterations: 25
```

- Looking at the 'p' values of the logistic regression , we can infer that 'sta_code' & 'Normalised' variables influence the dependent variable (Base Hour Rate) highly.

```
> accuracy(glm.fit)
                        ME         RMSE          MAE  MPE MAPE         MASE
Training set -1.856771e-08 5.386075e-05 7.079525e-07 -Inf  Inf 1.700283e-06
> L<-predict(lm.fit,testData)
> L<-predict(glm.fit,testData)
> accuracy(L,trainingData$Normalised)
               ME     RMSE      MAE     MPE      MAPE
Test set 439.3712 29042.76 13095.64 5945148 162620900
>
```

- From the predictions RMSE , we have come to the following conclusion

  - our division of test and train data with respect to quantity might be wrong.

  - looking at the sta_code , we have observed that there are two distinct station codes(airport code) and one airport code is very small with respect to the other . This could possibly influence the prediction as the division of data might not be even.

  - use of multiple numeric independent variable to obtain regression of factor dependent variable might have resulted in bad model.

KNN Classification

```
#KNN Classification
sqrt(24)
m2<-knn(train = trainingData,test=testData,cl=train_target,k=5)
table(test_target,m2)
```

```
> table(test_target,m2)
           m2
test_target      0      1
          0 131664      0
          1      0  54881
```

```
> mean(test_target==m2)
[1] 1
```

- it looks like the algorithm successfully predicted base hour rate for our test data with an accuracy of 100%.
- this results could be because we have normalized our input variable between a constant range of values (0 - 1)

Random Forest

```
#Random Forest classification


y.dep1 <- 22
x.indep1 <- c(1:21)

system.time(
  rforest.model1 <- h2o.randomForest(y=y.dep1, x=x.indep1, training_frame = train.h2o,
                                     ntrees = 1000, mtries = 3, max_depth = 4, seed = 1122)
)
#check variable importance
h2o.varimp(rforest.model1)

#making predictions on unseen data
system.time(predict.rforest1 <- as.data.frame(h2o.predict(rforest.model1, test.h2o)))
sub_rf <- data.frame(BaseHourRate =  predict.rforest1$predict)
```

```
Variable Importances:
     variable relative_importance scaled_importance percentage
1  Normalised     29537094.000000          1.000000   0.840784
2    sta_code      1431060.125000          0.048450   0.040736
3  BuildingID       928483.750000          0.031434   0.026430
4       month       694171.687500          0.023502   0.019760
5 TemperatureF      528416.625000          0.017890   0.015042

---
              variable relative_importance scaled_importance percentage
16 Sea.Level.PressureIn        14193.941406          0.000481   0.000404
17             Humidity         9323.667969          0.000316   0.000265
18        Wind.SpeedMPH         7883.343262          0.000267   0.000224
19                  day         2361.557617          0.000080   0.000067
20       WindDirDegrees         1974.001343          0.000067   0.000056
21                 mont         1511.108887          0.000051   0.000043
>

  BaseHourRate testDataA..22..
1            0               0
2            0               0
3            1               1
4            1               1
5            0               0
6            1               1
```

- Look at our predictions for random forest for classification , we are getting 100% accuracy for predicting the test data with train data for Base Hour rate

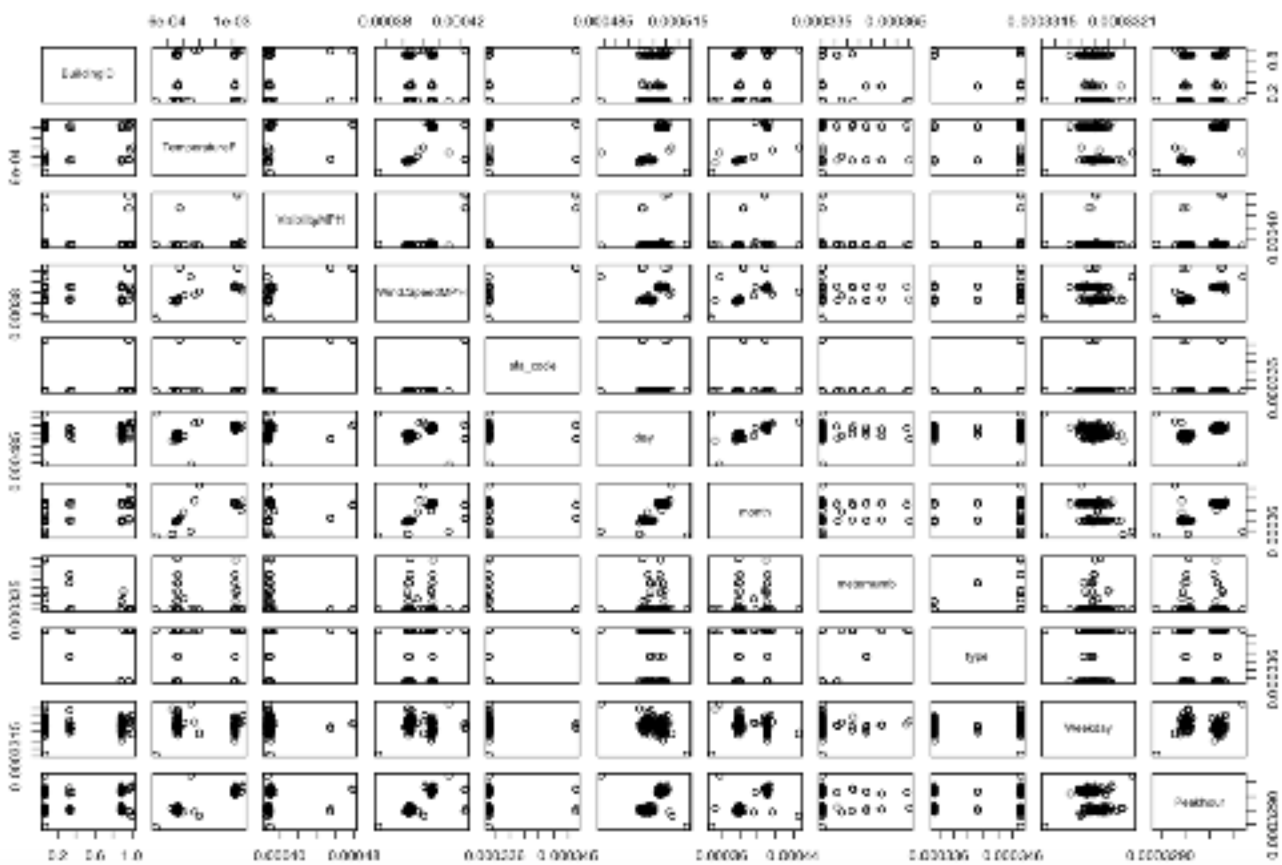
Conclusion for Classification :

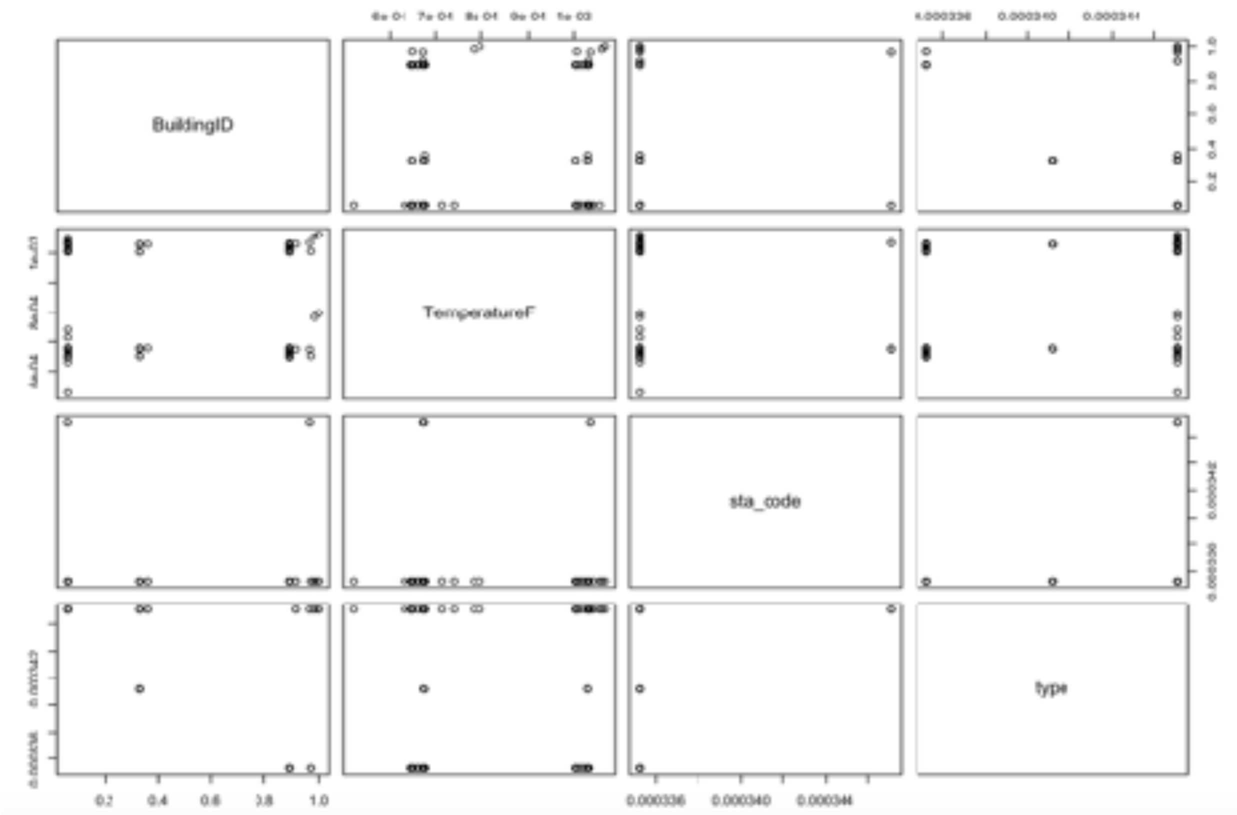
For the regression model , K nearest neighbor and Random Forest yielded high accuracy as compared to logistic model.

Clustering

K-means Clustering

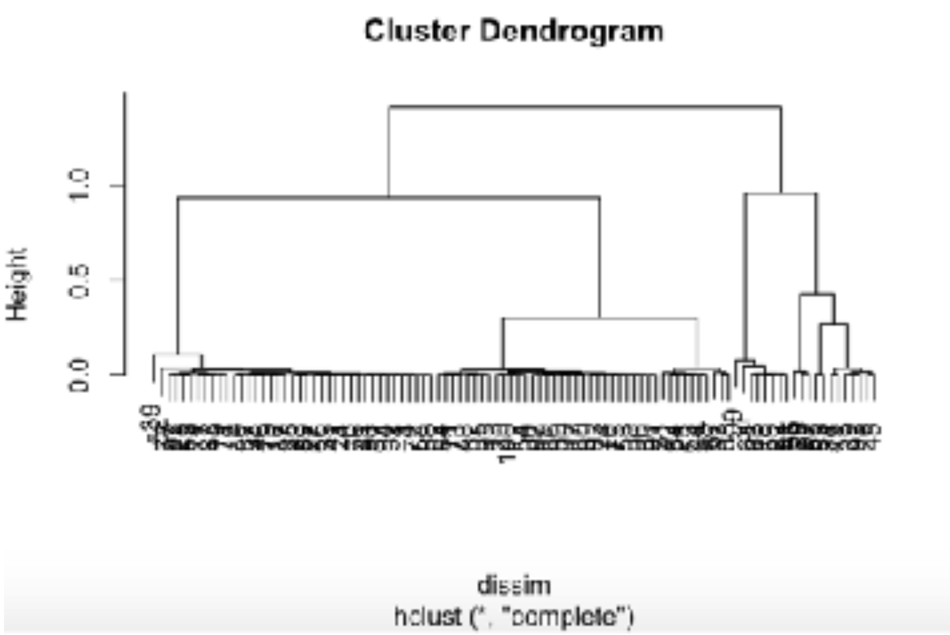
```
#Kmeans clustering
fx <- trainingData[,data.table(kmeans(cbind(TemperatureF,sta_code,type),centers=2)$centers),by=BuildingID]
fx1 <- trainingData[,data.table(kmeans(cbind(TemperatureF,VisibilityMPH,Wind.SpeedMPH,sta_code,day,
                                month,meternumb,type,Weekday,Peakhour),
                            centers=2)$centers),by=BuildingID]

plot(fx1)
plot(fx)
```




BuildingID
TemperatureF
sta_code
type

Hierarchical Clustering

```
#Hierarchial clustering
install.packages("fastcluster")
library('fastcluster')
df<-as.data.frame(trainingData)
dissim <- dist(df[1:50,1:10])
hr <- hclust(dissim)
plot(hr)

clusterCut <- cutree(hr, 4)
table(clusterCut, df$BuildingID[1:50])
```



NEURAL NETWORK

Artificial Neural Networks(ANN) are implemented by the following process,

1. Do the train-test split and fit the model to the train set
2. Normalize the data
3. Categorize the input parameters based on regression
4. Implement the neuralnet() function
5. Carry out multiple iterations.

1.After getting the input from part 1, we start by randomly splitting the data into a train and a test set, then we fit a linear regression model and test it on the test set.

```
index <- sample(1:nrow(DT_2),round(0.75*nrow(DT_2)))

train <- DT_2[index,]

test <- DT_2[-index,]

lm.fit <- glm(Consumption~., data=train)

summary(lm.fit)

pr.lm <- predict(lm.fit,test)

MSE.lm <- sum((pr.lm - test$Consumption)^2)/nrow(test)
```

2. Normalizing the data using the min-max method.

```
maxs <- apply(DT_2, 2, max)

mins <- apply(DT_2, 2, min)

scaled <- as.data.frame(scale(DT_2, center = mins, scale = maxs - mins))

train_ <- scaled[index,]

test_ <- scaled[-index,]
```

3.Input parameters based on the regression

- Weekday
- Peakhour
- Mont
- Sta_code

- Day
- Month
- BuildingID
- Meternumb
- Type
- Area_floor _m.sqr
- TemperatureF
- Wind.SpeedMPH
- Gust.SpeedMPH

4 Implementing the function neuralnet() for the train and test data.

```
nn <- neuralnet(
  Consumption ~Weekday+Peakhour+mont+sta_code+day+month+
    BuildingID+meternumb+type+TemperatureF+Wind.SpeedMPH+Gust.SpeedMPH,
  data=train_, hidden=5, err.fct="ce",
  linear.output=FALSE)

nn <- neuralnet(
  Consumption ~Weekday+Peakhour+mont+sta_code+
    day+month+BuildingID+meternumb+type+TemperatureF+Wind.SpeedMPH+Gust.SpeedMPH,
  data=test_, hidden=5, err.fct="ce",
  linear.output=FALSE)
```

5a. When we used one neuron in the hidden layer, we found the error to be very high, 318 to be precise.

The result and the pictorial representation is as shown below.

error	318.643978590810
reached.threshold	0.009923574821
steps	11343.000000000000
Intercept.to.1layhid1	-16.199244005901
Weekday.to.1layhid1	-0.797269070692
Peakhour.to.1layhid1	-1.035939020437
mont.to.1layhid1	0.227693918439
sta_code.to.1layhid1	38.499532668095
day.to.1layhid1	-0.237598592023
month.to.1layhid1	0.762361266469
BuildingID.to.1layhid1	13.556366162686
meternumb.to.1layhid1	16.813724278667
type.to.1layhid1	15.446192809758
TemperatureF.to.1layhid1	10.469788211412
Wind.SpeedMPH.to.1layhid1	-1.549429791468

Gust.SpeedMPH.to.1layhid1 0.727833782168
Intercept.to.Consumption -1.560359741601
1layhid.1.to.Consumption -1.897422456452

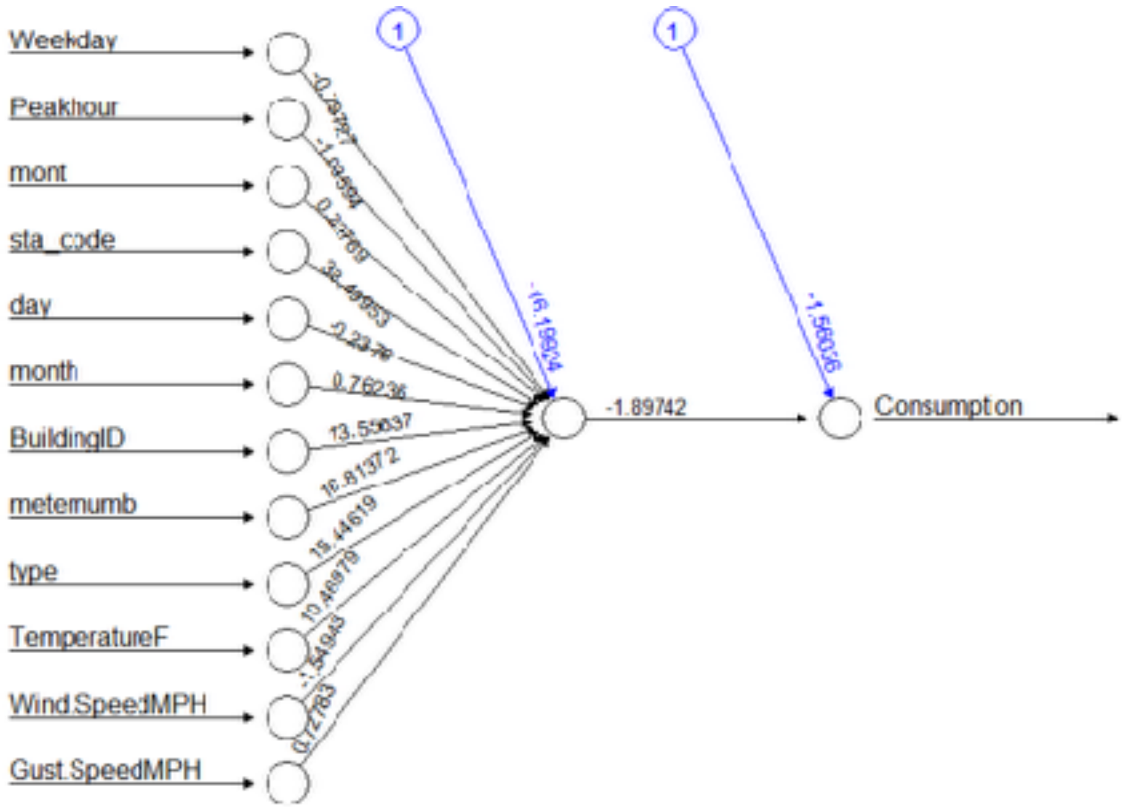
5b. We used 5 neurons in the hidden layer and we find the error to be 1.31, a very significant improvement from our previous iteration. Our model now is way more effective and precise.

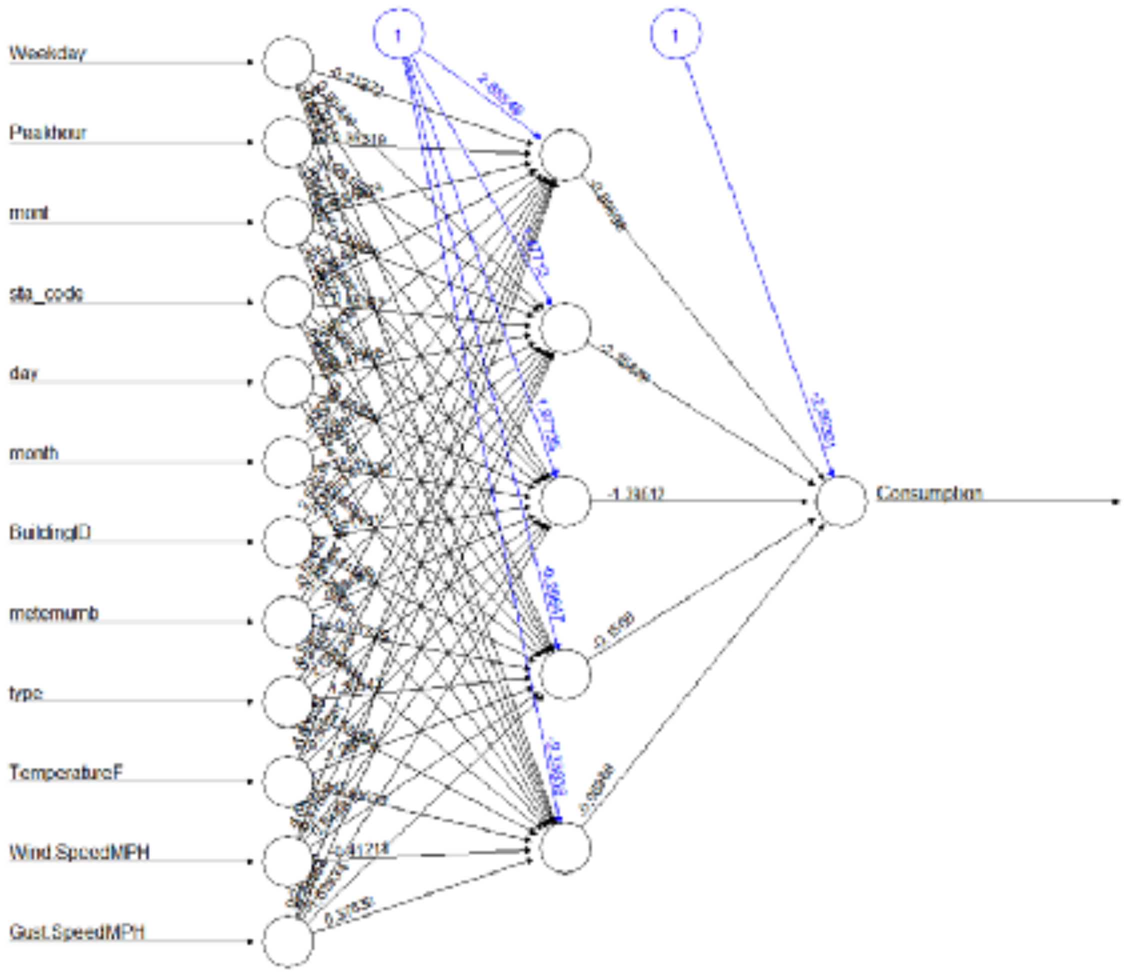
error	1.318237568931
reached.threshold	0.004516627377
steps	19.000000000000
Intercept.to.1layhid1	2.855457390283
Weekday.to.1layhid1	-0.212721260154
Peakhour.to.1layhid1	0.363186786673
mont.to.1layhid1	-0.956285792057
sta_code.to.1layhid1	0.365870527727
day.to.1layhid1	-1.369086711318
month.to.1layhid1	-0.720694728398
BuildingID.to.1layhid1	2.642342343749
meternumb.to.1layhid1	-0.536251391682
type.to.1layhid1	-0.928107850618
TemperatureF.to.1layhid1	-1.469454309811
Wind.SpeedMPH.to.1layhid1	-1.561321086177
Gust.SpeedMPH.to.1layhid1	0.230634231519
Intercept.to.1layhid2	1.477126460176
Weekday.to.1layhid2	0.904356922957
Peakhour.to.1layhid2	3.031855605898
mont.to.1layhid2	2.583836778537
sta_code.to.1layhid2	0.683971789602
day.to.1layhid2	-0.477954798895
month.to.1layhid2	-0.581897516734
BuildingID.to.1layhid2	-0.820341771254
meternumb.to.1layhid2	0.725141469330
type.to.1layhid2	0.991061798303
TemperatureF.to.1layhid2	0.905089119727
Wind.SpeedMPH.to.1layhid2	2.053847914534
Gust.SpeedMPH.to.1layhid2	-0.170235618395
Intercept.to.1layhid3	1.977353993978
Weekday.to.1layhid3	0.732303865565
Peakhour.to.1layhid3	0.777574029014
mont.to.1layhid3	-0.803784782927
sta_code.to.1layhid3	-0.669269731634
day.to.1layhid3	-0.663888073167
month.to.1layhid3	-0.206355884779
BuildingID.to.1layhid3	1.111310851212

```
meternumb.to.1layhid3     -0.050778871185
type.to.1layhid3          2.051254070581
TemperatureF.to.1layhid3  -0.003813760523
Wind.SpeedMPH.to.1layhid3  0.378222389006
Gust.SpeedMPH.to.1layhid3  0.144818069673
Intercept.to.1layhid4     -0.296466875766
Weekday.to.1layhid4        0.448246274196
Peakhour.to.1layhid4      -0.443340483840
mont.to.1layhid4           0.190705890739
sta_code.to.1layhid4      -0.825902717874
day.to.1layhid4           -0.355175507222
month.to.1layhid4         -0.085313400164
BuildingID.to.1layhid4     0.415807489276
meternumb.to.1layhid4     -0.617353197524
type.to.1layhid4           1.355422512525
TemperatureF.to.1layhid4  -1.298482875283
Wind.SpeedMPH.to.1layhid4  0.544811243808
Gust.SpeedMPH.to.1layhid4 -1.639336655807
Intercept.to.1layhid5     -2.338363117352
Weekday.to.1layhid5        0.537142876512
Peakhour.to.1layhid5       0.819459306233
mont.to.1layhid5          -0.395904640687
sta_code.to.1layhid5       0.819848519567
day.to.1layhid5           -2.703404111654
month.to.1layhid5         -1.637540210180
BuildingID.to.1layhid5    -0.268324558076
meternumb.to.1layhid5      0.959948101243
type.to.1layhid5          -1.550136299243
TemperatureF.to.1layhid5  -0.490328732288
Wind.SpeedMPH.to.1layhid5 -0.412110686432
Gust.SpeedMPH.to.1layhid5  0.378315447403
Intercept.to.Consumption  -2.563012910893
1layhid.1.to.Consumption  -0.895376470240
1layhid.2.to.Consumption  -2.156791886898
1layhid.3.to.Consumption  -1.395115463591
1layhid.4.to.Consumption  -0.150803398059
1layhid.5.to.Consumption  -0.069682441638
```

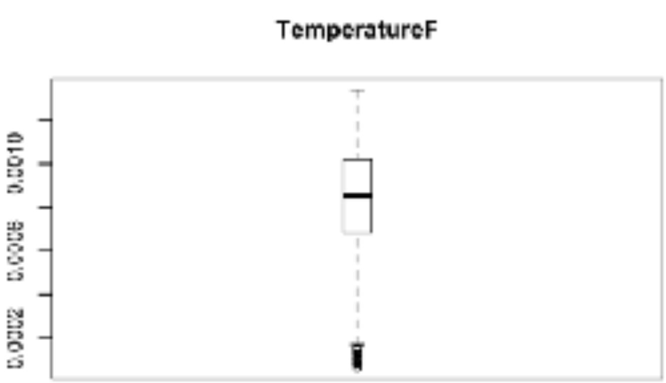
The result above displays the weights of the paths from the input to the hidden layer and from the hidden layer to the output layer. The plot below gives us a pictorial representation of the result.

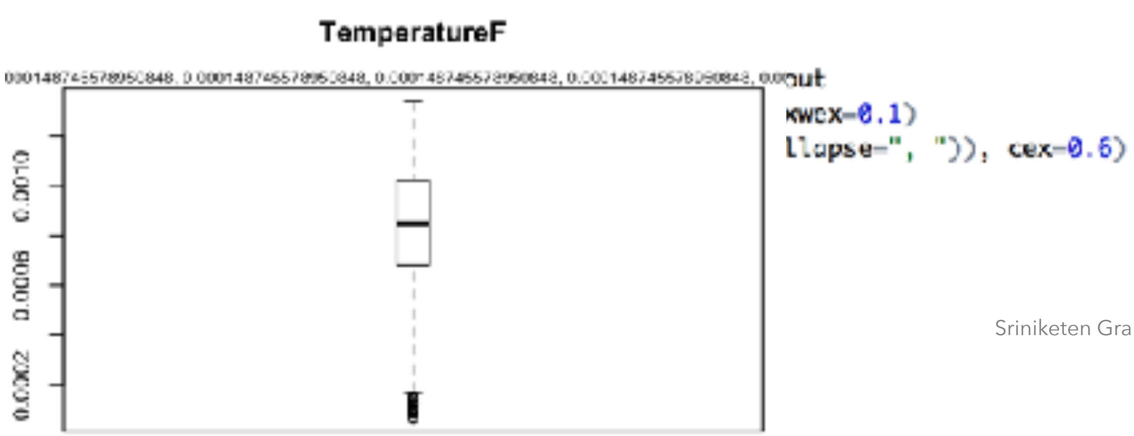
The result is plotted by using the script, plot(nn)


Weekday
Peakhour
mont
sta_code
day
month
BuildingID
meternumb
type
TemperatureF
Wind.SpeedMPH
Gust.SpeedMPH
1
1
-16.19924
-1.56036
-1.89742
0.76236
13.55837
16.81372
15.44619
Consumption


Weekday
Peakhour
mont
sta_code
day
month
BuildingID
meternumb
type
TemperatureF
Wind.SpeedMPH
Gust.SpeedMPH
1
1
Consumption

OUTLIER DETECTION





The outliers which are not too unrealistic are retained even though they are out of the box plot even though they are quite a lot in number.

On visual examination , we have removed -9999F as temperature value can never be in that value range.