# Commercial Building Energy Consumption modeling

**Team 4**

Ashish Dass, Anamika Jha, Shruti Narain

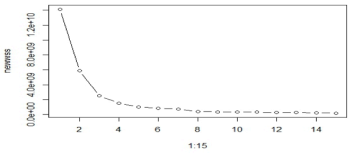
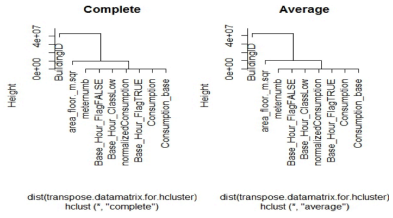
November 18, 2016

## Abstract

*This document demonstrates the modeling of building energy consumption by Tarja and Pasi Inc. Tarja and Pasi Inc. is an energy modeling consultancy and is monitoring energy consumption for 33 buildings owned by Vokia Inc. The models of energy consumption are using classification and prediction algorithms to predict the normalized energy consumption by these buildings and reduce energy usage in order to make their consumption efficient. The models are based on Linear Regression, KNN, Random Forest and Neural Network algorithms and have been optimized through feature selection and engineering.*

***An external configuration file has also been created (KMEANS.xml inside Clustering folder) where the user can change the input parameters based on his requirements without manual intervention in the code.***

*Final Statistics:*

| CLUSTERING TYPE | DATA SPLIT | ACCURACY ON SELECTED FEATURES | IMAGES | SELECTED FEATURES |
|---|---|---|---|---|
| **K Means** | 75% train<br>25% train | 78.00% |  | BuildingID","meternumb","Consumption","area_floor._m.sqr","normalizedConsumption","Base_Hour_Flag","Consumption_base","Base_Hour_Class" |
| **Heirarchical** | 65% train<br>35% test | 82.00% |  | BuildingID","meternumb","Consumption","area_floor._m.sqr","normalizedConsumption","Base_Hour_Flag","Consumption_base","Base_Hour_Class" |

| PREDICTION TYPE | DATA SPLIT | ACCURACY ON SELECTED FEATURES | ACCURACY ON ALL FEATURES | SELECTED FEATURES |
|---|---|---|---|---|
| **Regression** | 75% train<br>25% train | 99.50% | | Hour<br>Consumption<br>Month of Year<br>Holiday<br>Base Hour Flag<br>Base Consumption<br>Visibility |
| **Neural Networks** | 65% train<br>35% test | 75.63% | | Temperature<br>area_floor._m.sqr<br>normalizedConsumption<br>hour<br>Consumption<br>monthofYear<br>holiday<br>Base_Hour_Flag<br>Consumption_base<br>Base_Hour_Class |

| CLASSIFICATION TYPE | DATA SPLIT | ACCURACY ON SELECTED FEATURES | ACCURACY ON ALL FEATURES | SELECTED FEATURES |
|---|---|---|---|---|
| **Regression** | 75% train<br>25% train | 91.80% | | Hour<br>Consumption<br>Month of Year<br>Holiday<br>Base Hour Flag<br>Base Consumption |
| **K Nearest Neighbor** | 65% train<br>35% test | 100% | 68.91% | area_floor._m.sqr<br>Consumption<br>Temperature<br>hour<br>normalizedConsumption |
| **Random Forest** | 65% train<br>35% test | 97.56% | 77.54% | Temperature<br>area_floor._m.sqr<br>normalizedConsumption<br>hour<br>Consumption<br>monthofYear<br>holiday<br>Base_Hour_Flag |
| **Neural Networks** | 65% train<br>35% test | 75.63% | | Temperature<br>area_floor._m.sqr<br>normalizedConsumption<br>hour<br>Consumption<br>monthofYear |

# 1. Data Ingestion and Wrangling:

A part of the sample data comes from the csv files provided to us as Finland_masked_csv.csv which has the hourly detailed energy consumption data for all the 33 buildings for close to a year (2013). There is one other csv file is Finland_address_area.csv which contains the address of these 33 buildings. Other half of the data (i.e., the weather related) has been extracted online after the geolocation and nearest airport station for all the buildings were retrieved. The geolocation data was obtained using the address of the buildings provided to us. The data from all these three sources has been combined using R script and the below modifications has been done.

## 1.1 Data from csv files

- **Building Name:**
  The vac column (Building name) of the Finland_masked_csv is missing 2 values. Based on the Building_Id and their energy consumption the missing values has been replaced.

- **Type of consumption:**
  Only heating and electricity consumption for these 33 building have been taken into consideration for further analysis.

- Now that we have a column for power consumption of every day for every hour, we have added separate columns for day of week, month of year, weekday/weekend, Holiday and base hour flag (for 0,1,2,3,4,22,23) all of them derived from the date and hour column using the below logic.
- Day of Week: Assigned values 0-6 from Sun-Sat respectively to values taken from date column.
- Month of Year: Has the value 1-12 based on what month of the year it is.
- Weekday: value 1 if the day is a weekday else 0 (with use of chron library).
- Holiday: Holiday data for Finland has been extracted from the website http://www.timeanddate.com/calendar/?year=2013&country=24 and the dates mentioned here were assigned a flag of TRUE else FALSE.
- Base hour: for 0,1,2,3,4,22,23-hour value TRUE has been assigned else FALSE.

## 1.2 Data from wunderground website

The rest of the columns for geolocation data has been retrieved using "geoshpere" package. The data that is obtained through this package is the latitude and longitude information of all the address provided in the csv. These geolocations are further used to obtain the nearest airport location for all the addresses. RCurl, httr and XML packages are used to get these airport stations along with Ram-N/weatherdata github install.

- **Geolocation of the buildings:**
  The geolocations of all the buildings are extracted from the website https://developers.google.com/maps/documentation/geocoding/start .This gives us the latitude and the longitude information for all the buildings.

- **Airport Stations:**
  The nearest airport stations have then been extracted based on the geolocation from http://api.wunderground.com/auto/wui/geo/GeoLookupXML/index.xml?query=lat, long If an empty value was obtained from the list while parsing the xml, the next airport station closest to the given geolocation was

considered.

Now that we have the airport stations, same station codes were used to get the weather information for the year 2013.
The following cleansing operations are performed on the weather data that we scrape:

- Precipitation columns had N/A in the records which were replaced by NA.
- Erroneous/missing values for some days were replaced by an average value of next day data.
- Columns like Wind direction, Gust Speed, Conditions, Temperature, Dew Point, Humidity, Sea Level Pressure, Visibility, Wind Speed and Wind direction degrees had NA values which were replaced with the next value obtained i.e., value for next hour.

The two data sets are then merged based on the Date, Building name, BuildingID and meter number columns.

## 2. Modeling:

- A column named Normalized consumption is created by dividing the power consumed by a building per hour of every day by the area occupied by the building in square meter.
- Another column named Consumption_base has been calculated using the average of energy consumption for every building in their base hour and the same values is repeated for the entire day.
- A flag of Low/High has been assigned in the column base_hour_class where Consumption is lower/higher than the Consumption_base value respectively

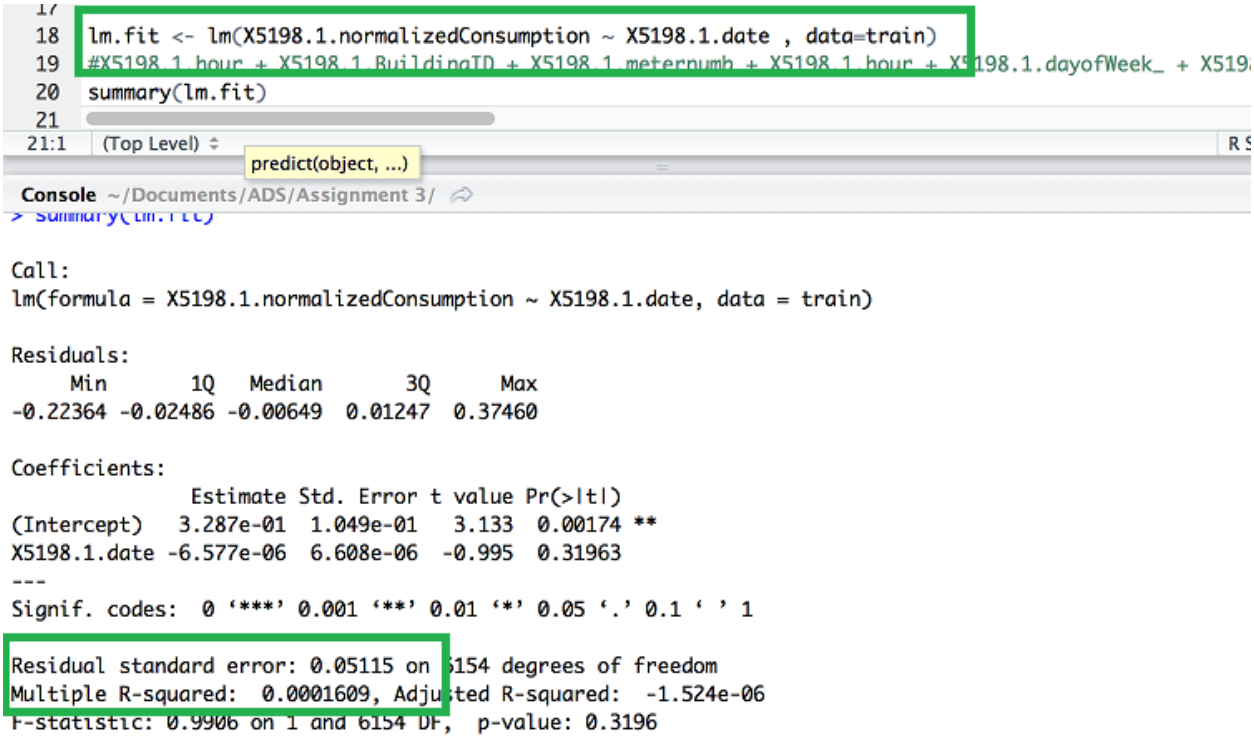
## PREDICTION:

- In order to do prediction on the data we have for all the buildings, the data has been spit in 78 data frames based on unique combination of Building Id and meter numbers.

- The screenshots pasted below shows the results of modeling for one such sample(combination of one Building Id and one meter number)
- Same script was run in a loop to obtain similar results for all 78 unique combinations.

# Multiple Linear Regression:

- The variable selection for multiple linear regressions was done using forward selection method.
- Started modelling with date alone as an independent variable.



```
18  lm.fit <- lm(X5198.1.normalizedConsumption ~ X5198.1.date , data=train)
19  #X5198.1.hour + X5198.1.BuildingID + X5198.1.meternumb + X5198.1.hour + X5198.1.dayofWeek_ + X519
20  summary(lm.fit)
21
21:1  (Top Level)
```

predict(object, ...)

Console ~/Documents/ADS/Assignment 3/

```
> summary(lm.fit)

Call:
lm(formula = X5198.1.normalizedConsumption ~ X5198.1.date, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.22364 -0.02486 -0.00649  0.01247  0.37460

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.287e-01  1.049e-01   3.133  0.00174 **
X5198.1.date -6.577e-06  6.608e-06  -0.995  0.31963
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05115 on 6154 degrees of freedom
Multiple R-squared:  0.0001609, Adjusted R-squared:  -1.524e-06
F-statistic: 0.9906 on 1 and 6154 DF,  p-value: 0.3196
```

- Repeated the process for other dependent variables.
- We kept adding one more variables at the time to check any improvements in the residual error and multiple R squared values.

```
> lm.fit <- lm(X5198.1.normalizedConsumption ~ X5198.1.date + X5198.1.BuildingID + X5198.1.meternumb, data=
train)
> summary(lm.fit)

Call:
lm(formula = X5198.1.normalizedConsumption ~ X5198.1.date + X5198.1.BuildingID +
    X5198.1.meternumb, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.22364 -0.02486 -0.00649  0.01247  0.37460

Coefficients: (2 not defined because of singularities)
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.287e-01  1.049e-01   3.133  0.00174 **
X5198.1.date       -6.577e-06  6.608e-06  -0.995  0.31963
X5198.1.BuildingID         NA         NA      NA       NA
X5198.1.meternumb          NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05115 on 6154 degrees of freedom
Multiple R-squared:  0.0001609, Adjusted R-squared:  -1.524e-06
F-statistic: 0.9906 on 1 and 6154 DF,  p-value: 0.3196
```

- The regression model for run including all the dependent variables in the end.
- And we observed that many of the variables were insignificant for the regression model (with 0 stars/. mapped to it).

```
22
23  lm.fit <- lm(X5198.1.normalizedConsumption ~ X5198.1.date + X5198.1.BuildingID + X5198.1.meternumb
24               + X5198.1.hour + X5198.1.Consumption + X5198.1.Temperature + X5198.1.Dew_PointF
25               +X5198.1.dayofWeek_ +X5198.1.monthofYear +X5198.1.isWeekend +X5198.1.holiday
26               +X5198.1.Wind_Direction + X5198.1.Gust_SpeedMPH +X5198.1.Conditions
27               +X5198.1.Base_Hour_Flag +X5198.1.Base_Hr_Usage + X5198.1.VisibilityMPH
28               +X5198.1.Humidity + X5198.1.Sea_Level_PressureIn + X5198.1.Wind_SpeedMPH
29               +X5198.1.WindDirDegrees +X5198.1.area_floor._m.sqr
30               +X5198.1.Consumption_base +X5198.1.Base_Hour_Class, data=train)
31  summary(lm.fit)
32
33
35:1  (Top Level)                                                              R Script

Console ~/Documents/ADS/Assignment 3/
X5198.1.WindDirDegreesWNW                    NA         NA      NA       NA
X5198.1.WindDirDegreesWSW                    NA         NA      NA       NA
X5198.1.area_floor._m.sqr                    NA         NA      NA       NA
X5198.1.Consumption_base               4.623e-05  1.796e-05   2.574 0.010087 *
X5198.1.Base_Hour_ClassLow             1.850e-04  1.197e-04   1.545 0.122308
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.003699 on 6059 degrees of freedom
Multiple R-squared:  0.9949,    Adjusted R-squared:  0.9948
F-statistic: 1.22e+04 on 96 and 6059 DF,  p-value: < 2.2e-16
```

- The insignificant variables were removed and finally we reached our optimum model with Residual standard error = 0.0039 and Multiple R squared value of 0.9942.

```
22
23  lm.fit <- lm(X5198.1.normalizedConsumption ~ X5198.1.hour + X5198.1.Consumption
24               +X5198.1.monthofYear +X5198.1.holiday
25               +X5198.1.Base_Hour_Flag + X5198.1.VisibilityMPH
26               +X5198.1.Consumption_base, data=train)
27  summary(lm.fit)
28
29
37:1   Getting 1 coefficient

Console ~/Documents/ADS/Assignment 3/

X5198.1.monthofYearNov     -2.142e-04  2.604e-04  -0.823 0.410650
X5198.1.monthofYearOct      3.033e-05  2.390e-04   0.127 0.899004
X5198.1.monthofYearSep     -5.986e-06  2.370e-04  -0.025 0.979853
X5198.1.holidayTRUE         2.205e-03  2.586e-04   8.528  < 2e-16 ***
X5198.1.Base_Hour_FlagTRUE  8.014e-05  1.178e-04   0.680 0.496507
X5198.1.VisibilityMPH       2.843e-08  1.147e-08   2.479 0.013189 *
X5198.1.Consumption_base    6.368e-05  1.605e-05   3.967 7.37e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.003902 on 6138 degrees of freedom
Multiple R-squared:  0.9942,    Adjusted R-squared:  0.9942
F-statistic: 6.186e+04 on 17 and 6138 DF,  p-value: < 2.2e-16
```

- Now we check for the collinearity between the selected variables for regression model using the vif function.
- If the GVIF value for any variable exceeds 5 it means that the variables have collinear relationship.
- In our model we see that none of the GVIF values are more than 5 so we are good to go.

```
> vif(lm.fit)
                            GVIF Df GVIF^(1/(2*Df))
X5198.1.hour            1.038885  1        1.019257
X5198.1.Consumption     2.516972  1        1.586497
X5198.1.monthofYear     1.640720 11        1.022761
X5198.1.holiday         1.069546  1        1.034188
X5198.1.Base_Hour_Flag  1.063512  1        1.031267
X5198.1.VisibilityMPH   1.173108  1        1.083101
X5198.1.Consumption_base 2.664598  1        1.632360
>
```

- Regression Output:

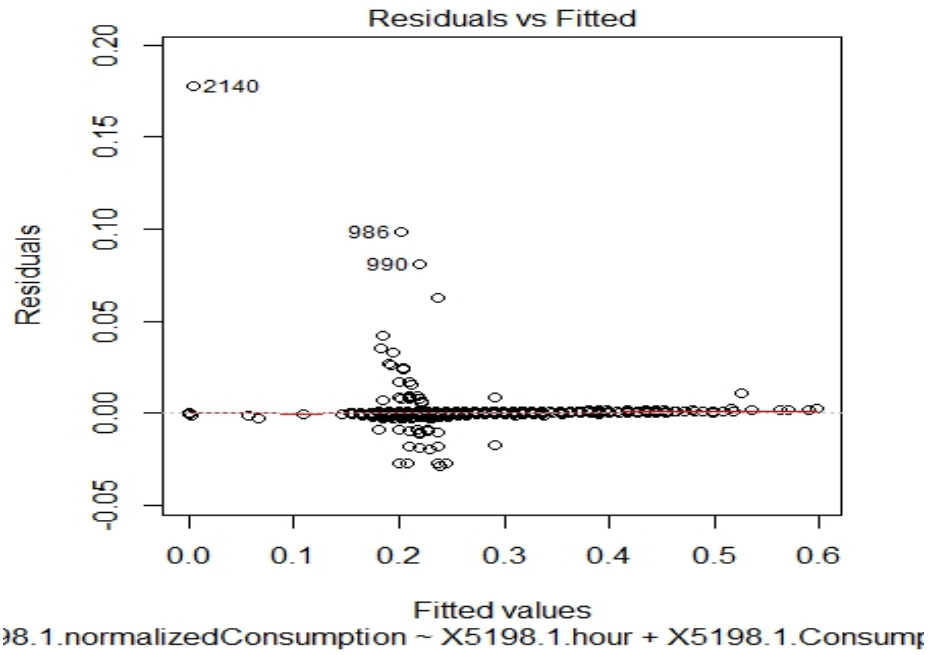
```
> RegressionOutput <- summary(lm.fit)$coefficients[,1]
> RegressionOutput
                 (Intercept)                X5198.1.hour         X5198.1.Consumption
                5.934407e-04               -2.133696e-05                9.025523e-03
     X5198.1.monthofYearAug      X5198.1.monthofYearDec      X5198.1.monthofYearFeb
                1.968825e-05               -3.912231e-04                6.414173e-04
     X5198.1.monthofYearJan      X5198.1.monthofYearJul      X5198.1.monthofYearJun
               -1.708614e-04               -4.172301e-05                8.553049e-04
     X5198.1.monthofYearMar      X5198.1.monthofYearMay      X5198.1.monthofYearNov
                3.048313e-04               -2.682476e-04               -2.035901e-04
     X5198.1.monthofYearOct      X5198.1.monthofYearSep         X5198.1.holidayTRUE
               -3.290943e-06               -3.234071e-05                2.076234e-03
X5198.1.Base_Hour_FlagTRUE       X5198.1.VisibilityMPH    X5198.1.Consumption_base
               -1.044775e-06                2.324147e-08                5.451019e-05
> |
```

- Error value of our best fit regression model:

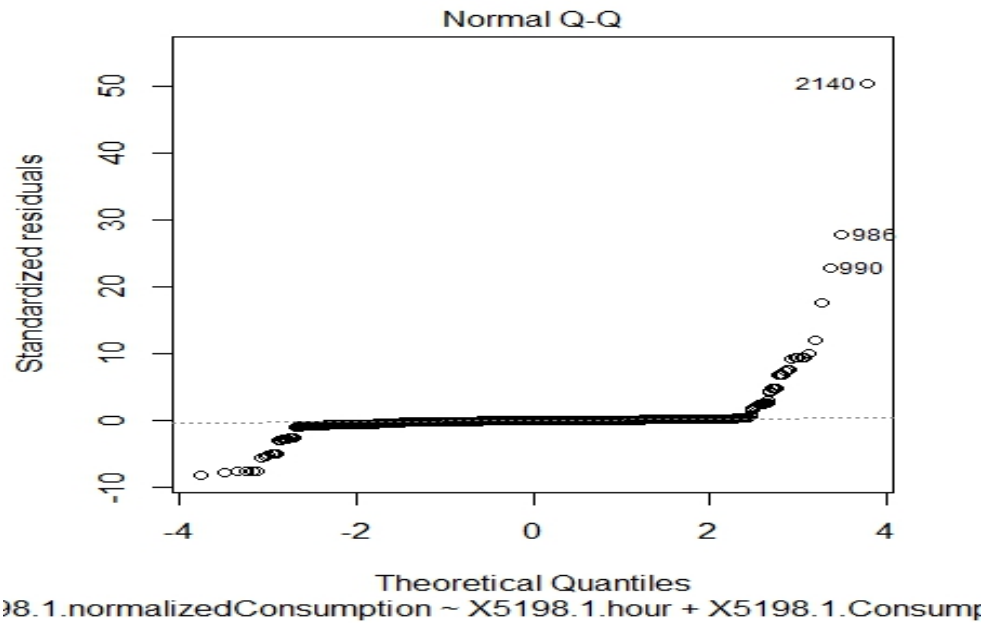
```
47  accuracyMatrix <- accuracy(pred,train$X5198.1.normalizedConsumption)
48  t(accuracyMatrix)
49
50  ################## CHECKING FIT OF ALL THE VARIABLES #########
51
50:1  CHECKING FIT OF ALL THE VARIABLES

Console C:/Users/AJha/Desktop/ADS Ass/Assignment 3/to be worked in office/

> accuracyMatrix <- accuracy(pred,train$X5198.1.normalizedConsumption)
> t(accuracyMatrix)
          Test set
ME    -0.0005792607
RMSE   0.0713606445
MAE    0.0466012771
MPE            -Inf
MAPE            Inf
```

Plots based on the regression model:

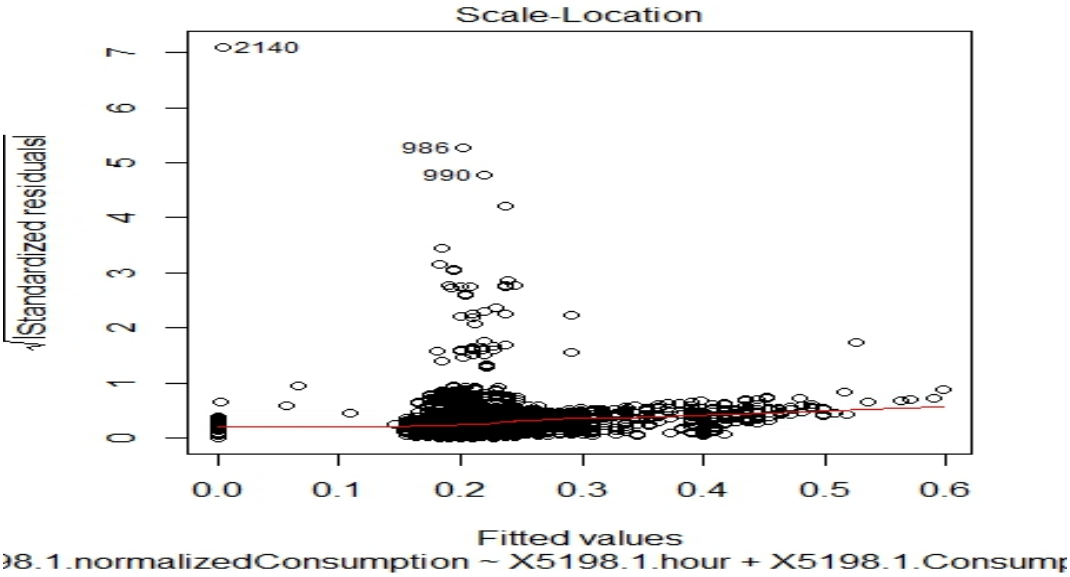
- The **Residual vs Fitted** plot shows us that most of the residual values for power consumption are close to the predicted response (Fitted values). Since it is along a straight line it shows that most of nonlinear relationship was not explained in the model and it was left out in the residuals.



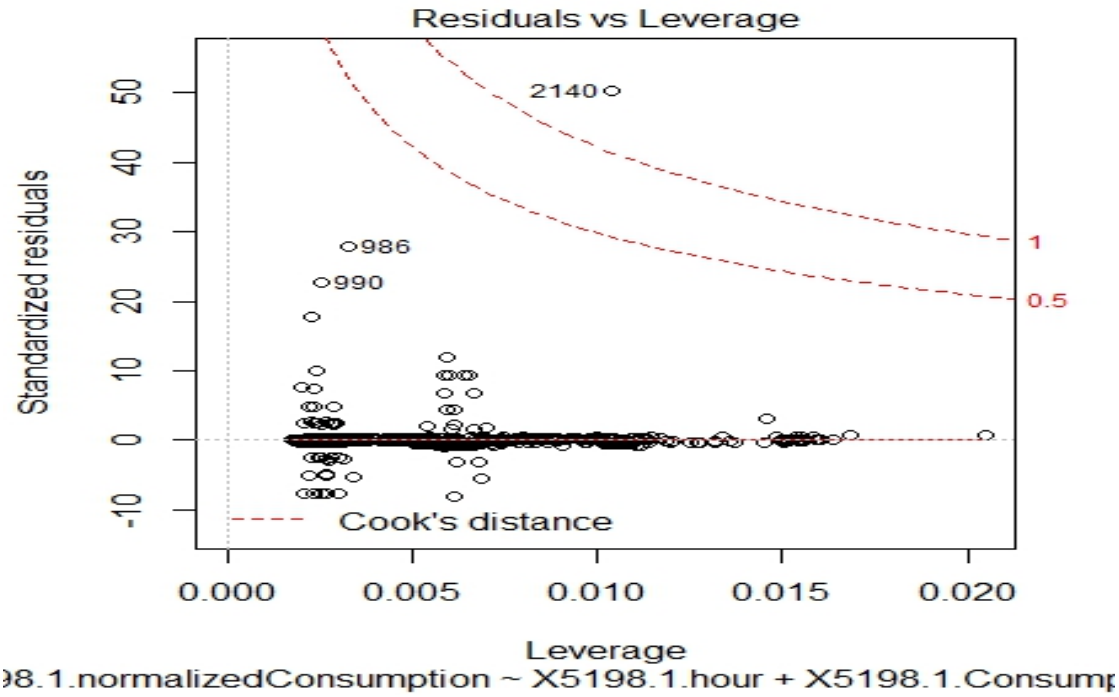

- The **Normal Q-Q plot** takes the sample data, sort it in ascending order, and then plot their quantiles against each other. In our plot we see that most of the data is parallel to theoretical Quantiles and are lined on the dashed line and are hence normally distributed. This plot provides us the probability distribution for our data.
- There are few data points like 2140, 986 and 990 that are way off the distribution line and can be a potential problem in our model.




- The **Scale-Location plot** shows if residuals are spread equally along the range of predictors which in our case holds true for most of the data. Of course there are some points that are a little off (same points we detected as a problem in our Normal Q-Q plot to be a potential problem)



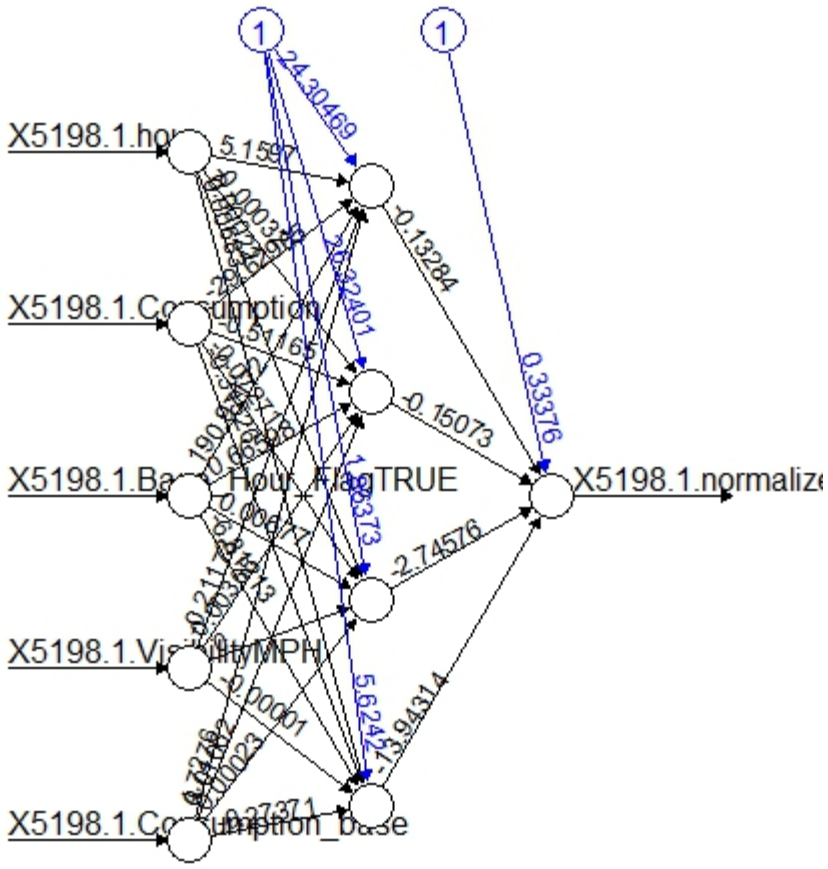
- The **Residual vs Leverage** plot helps us to find the influential factors in our model. Not all the variables selected for a regression model have same influence on the result of the model.
- When cases are outside of the Cooks distance, the cases are influential to the regression result which in case of our model is the 2140 record.
- If the record no 2140 is removed from the model, the model will provide better results. (This potential problem could be happening because of data quality issues).
- Record no 986 and 990 could also be potential problems records.




## Neural networks:

Our goal here is to predict the normalized energy consumption for every combination of Building Id and meter number using all other continuous variables. The first step towards doing this is to convert the 78 data frames that we have obtained by splitting the data set based on the combination of meter no and Building Id into matrix so that neural network computation is efficient. Now the matrix is split into testing and train data set. The train data set is used to train the neural network model. This model is then tested on the test data set and the accuracy of the prediction is calculated.

```
> net_consumption_NN <- neuralnet(X5198.1.normalizedConsumption ~ X5198.1.hour + X5198.1.Consumption
+                         + X5198.1.Base_Hour_FlagTRUE + X5198.1.VisibilityMPH
+                         + X5198.1.Consumption_base,train_NN, hidden = 4,
+                         lifesign = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:   12253  error: 0.07266  time: 23.35 secs
```




The Summary of the prediction result using test data is discussed below. We see that the actual and the predicted values are pretty close and hence the model seems to be a good fit.

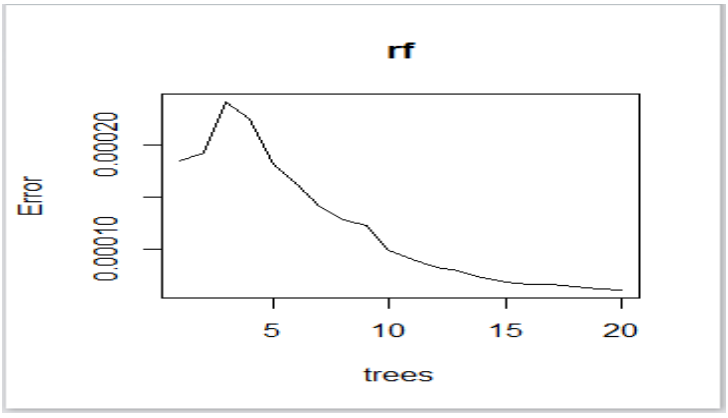
```
> summary(results)
     actual          prediction
 Min.   :0.0000000   Min.   :0.0000000
 1st Qu.:0.2000000   1st Qu.:0.1995511
 Median :0.2181818   Median :0.2167567
 Mean   :0.2262095   Mean   :0.2261418
 3rd Qu.:0.2363636   3rd Qu.:0.2360451
 Max.   :0.6000000   Max.   :0.6463376
```

- A comparison of the actual and the predicted value is presented

```
> head(results)
         actual   prediction
2  0.2272727273 0.2270197907
7  0.2272727273 0.2263560674
17 0.2181818182 0.2171460271
24 0.2090909091 0.2258759951
31 0.2363636364 0.2356626804
35 0.4818181818 0.4761367520
>
```

## Random Forest:

Random Forests improve the performance of decision trees. The algorithms build trees similar to a normal decision tree, however whenever a split is made it uses only a small random subset of features to decide the split instead of all the features. It builds multiple trees using the same process, and then takes the average of all the trees to arrive at the final model. This works by reducing the amount of correlation between trees, and thus helping reduce the variance of the final tree.

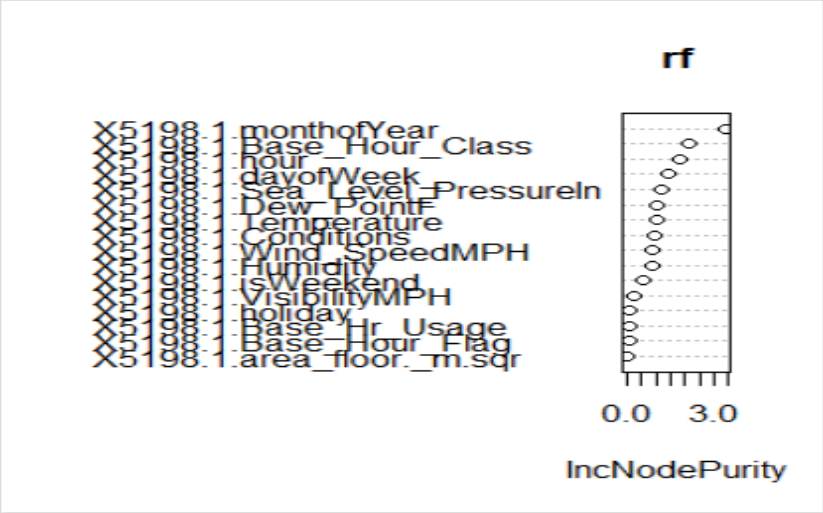



- According to the graph as the no of trees increases, the error decreases and there is improvement in the performance.

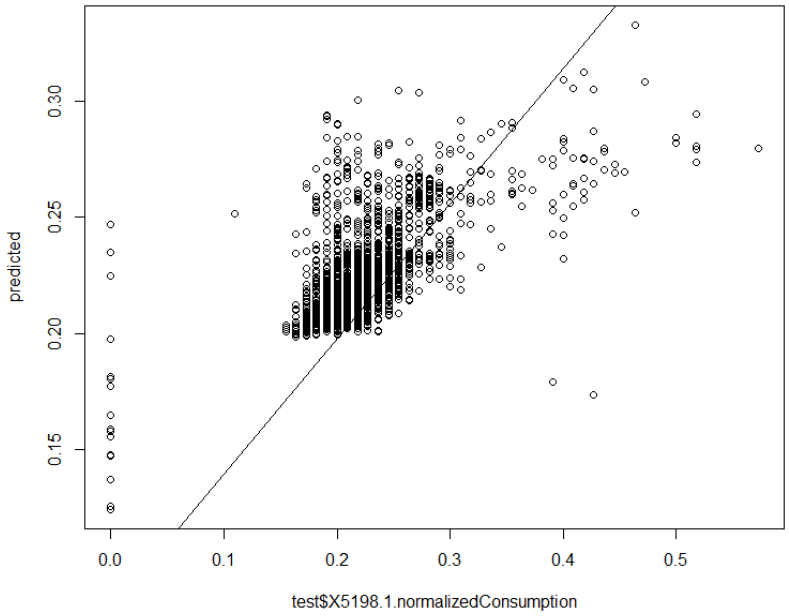
  However, the benefit in prediction performance by adding more trees, lowers than the cost in computation time for these additional trees.
  Hence, we picked the number of trees as 20, after which the if the number of trees increases, the computation time increases.

- A random forest can handle UNSCALED variables and CATEGORICAL variables, which reduces the need for cleaning and transforming variables which are steps that can be subject to overfitting and noise.

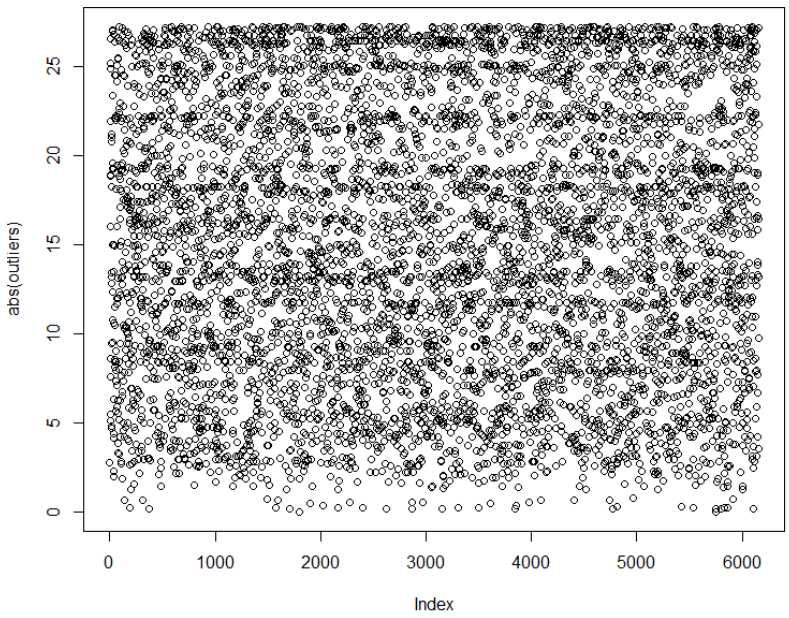
- According to the below plot, top 20% to 25% of the variables dominate the importance and should be considered for Random Forest Prediction.




- Therefore, while selecting variables in Random Forest Prediction, we consider the out-of-sample error rate
- On using varImpPlot() like in the above image, we can use top 20% to 25% of the variables that dominate the prediction. Hence, on taking top 5 out of 1 6 variables, the out-of-sample error rate is 0.02%
- Thus we pick top 5 variables for Random Forest Prediction and we get the below error values:
  RMSE: 0.0418
  MAE: 0.0253
- The graph plotted is as follows




- Residual Analysis

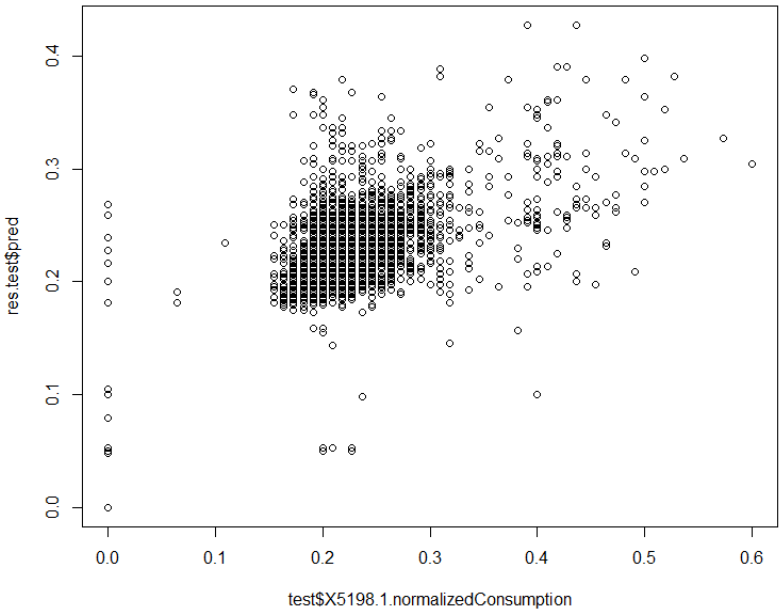



# KNN:

KNN algorithm is a non-parametric algorithm that is used for regression here Which means that it makes no assumption about the underlying data or its distribution. The data that we have is homogenous in nature and we need to classify the predicted column to one of the labeled groups. Like other algorithms

we divide our dataset into training and test data and then calculate the accuracy of the prediction. The result that we obtained are mentioned below:

- KNN handles only numeric variables.
- However, the factor variables can be converted into dummy variables using t he below line of
  code.
  predcheck <- model.matrix(~.+0, data=predcheck)
- We picked top 5 variables that we found in Random Forest Prediction. Few columns that were categorical were converted into dummy variables.
- On applying KNN on these selected features, the model couldn't predict bec ause there were a lot ties.
- Hence, we had to get rid of these specific factor variables and we selected th e variables that dominated the prediction.
- Graph representing Normalized Consumption vs Predicted Normalized Cons umption.

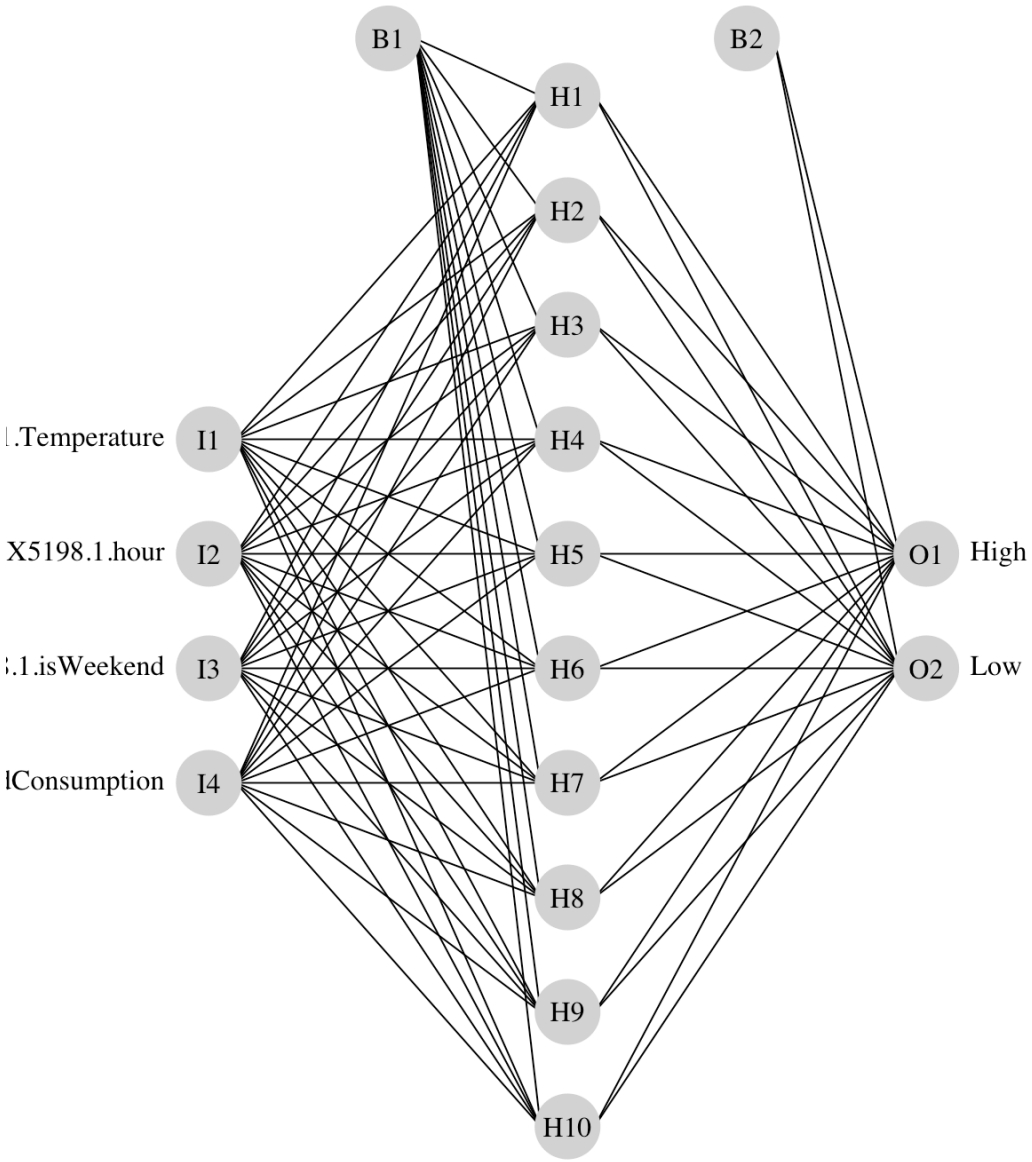


# CLASSIFICATION:

## Neural networks:

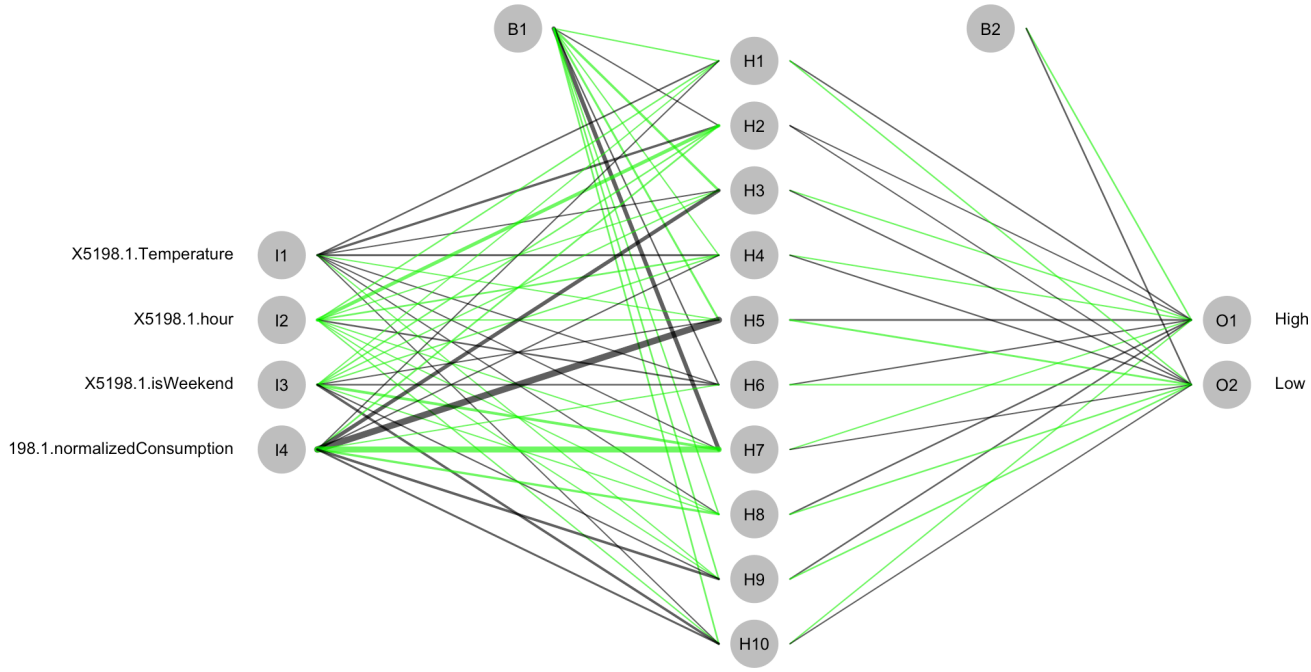
Classification using neural networks is using nnet package to classify the consumption of energy in high and low. Like other models the data is split into training and testing set to first train the model and test the result later.

```
install.packages("nnet")
library(nnet)
ideal <- class.ind(newesttrain$X5198.1.Base_Hour_Class)
ideal
consumptionANN = nnet(newesttrain[,-5], ideal, size = 10,softmax = TRUE)
predictANN <-predict(consumptionANN, newesttest[-5], type="class")
predictANNTable <- table(predictANN,newesttest$X5198.1.Base_Hour_Class)
predictANNTable
accuracyAnnPrediction <- (predictANNTable[1]+predictANNTable[4])/nrow(newesttest)
accuracyAnnPrediction

##75.63 % accuracy in PREDICTION OF BASE_HOUR_CLASS
```




The image above is a standard illustration of a neural network model. Each of the four random variables are shown in the first layer and the response variable is shown in the far right layer (labelled as High and Low). The hidden layer is labelled as H1 through H10, which was specified using the size argument in the nnet function. B1 and B2 are bias layers that apply constant values to the nodes, similar to intercept terms in a regression model. The image below is the same model illustrated as a neural interpretation diagram (default plot). The black

lines are negative weights and the green lines are positive weights. Line thickness is in proportion to magnitude of the weight relative to all others.




The function has several arguments that affect the plotting method:

| | |
|---|---|
| `consumption` | ANN model object for input created from `nnet`function |
| `nid` | logical value indicating if neural interpretation diagram is plotted, default T |
| `circle.cex` | numeric value indicating size of nodes, passed to cex argument, default 5 |
| `node.labs` | logical value indicating if text labels are plotted, default T |
| `line.stag` | numeric value that specifies distance of connection weights from nodes |
| `circle.col` | text value indicating color of nodes, default 'lightgrey' |
| `pos.col` | text value indicating color of positive connection weights, default 'black' |
| `neg.col` | text value indicating color of negative connection weights, default 'grey' |

```
#############################################################
#import function from Github
require(RCurl)

root.url<-'https://gist.githubusercontent.com/fawda123'
raw.fun<-paste(
  root.url,
  '5086859/raw/cc1544804d5027d82b70e74b83b3941cd2184354/nnet_plot_fun.r',
  sep='/'
)
script<-getURL(raw.fun, ssl.verifypeer = FALSE)
eval(parse(text = script))
rm('script','raw.fun')


par(mar=numeric(4),mfrow=c(1,2),family='serif')
plot(consumptionANN,nid=F)
plot(consumptionANN, pos.col='green',neg.col='black',alpha.val=0.7,rel.rsc=5,
     circle.cex=5,cex=0.8,
     circle.col='grey')
```

## Logistic Regression:

The model used for logistic regression is not very different from what we do for linear regression. The function used for Logistic Regression is glm() and it does the classification of whether the energy consumption is going to be high or low based on the binary model.

- So first we create the model using glm() function on our training data set including all the dependent variables.

```
23  model <- glm(X5198.1.Base_Hour_Class ~.,family=binomial(link='logit'),data=train)
24  summary(model)
39:2  (Top Level)
```

Console ~/Documents/ADS/Assignment 3/

```
                          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                                      6565       9073
X5198.1.date               1   101.38      6564       8971 < 2.2e-16 ***
X5198.1.BuildingID         0     0.00      6564       8971
X5198.1.meternumb          0     0.00      6564       8971
X5198.1.hour               1     1.30      6563       8970 0.2543492
X5198.1.Consumption        1  1268.81      6562       7701 < 2.2e-16 ***
X5198.1.dayofWeek_         1     1.42      6561       7700 0.2337252
X5198.1.monthofYear       11   723.77      6550       6976 < 2.2e-16 ***
X5198.1.isWeekend          0     0.00      6550       6976
X5198.1.Base_Hour_Flag     1     5.83      6549       6970 0.0157695 *
X5198.1.holiday            1     2.77      6548       6967 0.0962278 .
X5198.1.Base_Hr_Usage      0     0.00      6548       6967
X5198.1.Wind_Direction    17    30.23      6531       6937 0.0247663 *
X5198.1.Temperature        1     4.58      6530       6933 0.0323648 *
X5198.1.Dew_PointF         1     5.77      6529       6927 0.0163187 *
X5198.1.Humidity           1     4.63      6528       6922 0.0314746 *
X5198.1.Sea_Level_PressureIn 1   3.54      6527       6919 0.0600230 .
X5198.1.VisibilityMPH      1     0.70      6526       6918 0.4037968
X5198.1.Wind_SpeedMPH      1     1.49      6525       6916 0.2219371
X5198.1.WindDirDegrees     0     0.00      6525       6916
X5198.1.area_floor._m.sqr  0     0.00      6525       6916
X5198.1.normalizedConsumption 1  14.94      6524       6902 0.0001109 ***
X5198.1.Consumption_base   1     0.00      6523      56084 1.0000000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

- Here we see that there are few variables which does not have any stars rated to it meaning they do not contribute in the efficiency of the model. So we will go ahead and remove these variables from the model.

```
model1 <- glm(X5198.1.Base_Hour_Class ~ X5198.1.normalizedConsumption +X5198.1.monthofYear
              +X5198.1.Base_Hour_Flag +X5198.1.VisibilityMPH ,
              family=binomial(link='logit'),data=train)


summary(model1)
```

- Now the accuracy and the confusion matrix provides a much better results
- The collinearity between the variables are also checked and the value produced are below 5 (as expected).

| Prediction | Reference | Freq | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| High | High | 1680 | | | | | | | | | | |
| Low | High | 79 | | | | | | | | | | |
| High | Low | 240 | | | | | | | | | | |
| Low | Low | 1518 | | | | | | | | | | |
| Accuracy | Kappa | AccuracyLower | AccuracyUpper | AccuracyNull | AccuracyPValue | | | | | | | |
| 0.9092977 | 0.81859066 | 0.899320765 | 0.918587972 | 0.500142167 | 0 | | | | | | | |
| McnemarPV | Sensitivity | Specificity | Pos.Pred.Value | Neg.Pred.Value | Precision | Recall | F1 | Prevalence | Detection.Ra | Detection.Pr | Balanced.Accuracy | |
| 3.2978E-19 | 0.95508812 | 0.863481229 | 0.875 | 0.950532248 | 0.875 | 0.95508812 | 0.91329166 | 0.50014217 | 0.47767984 | 0.54591982 | 0.90928467 | |

## KNN:

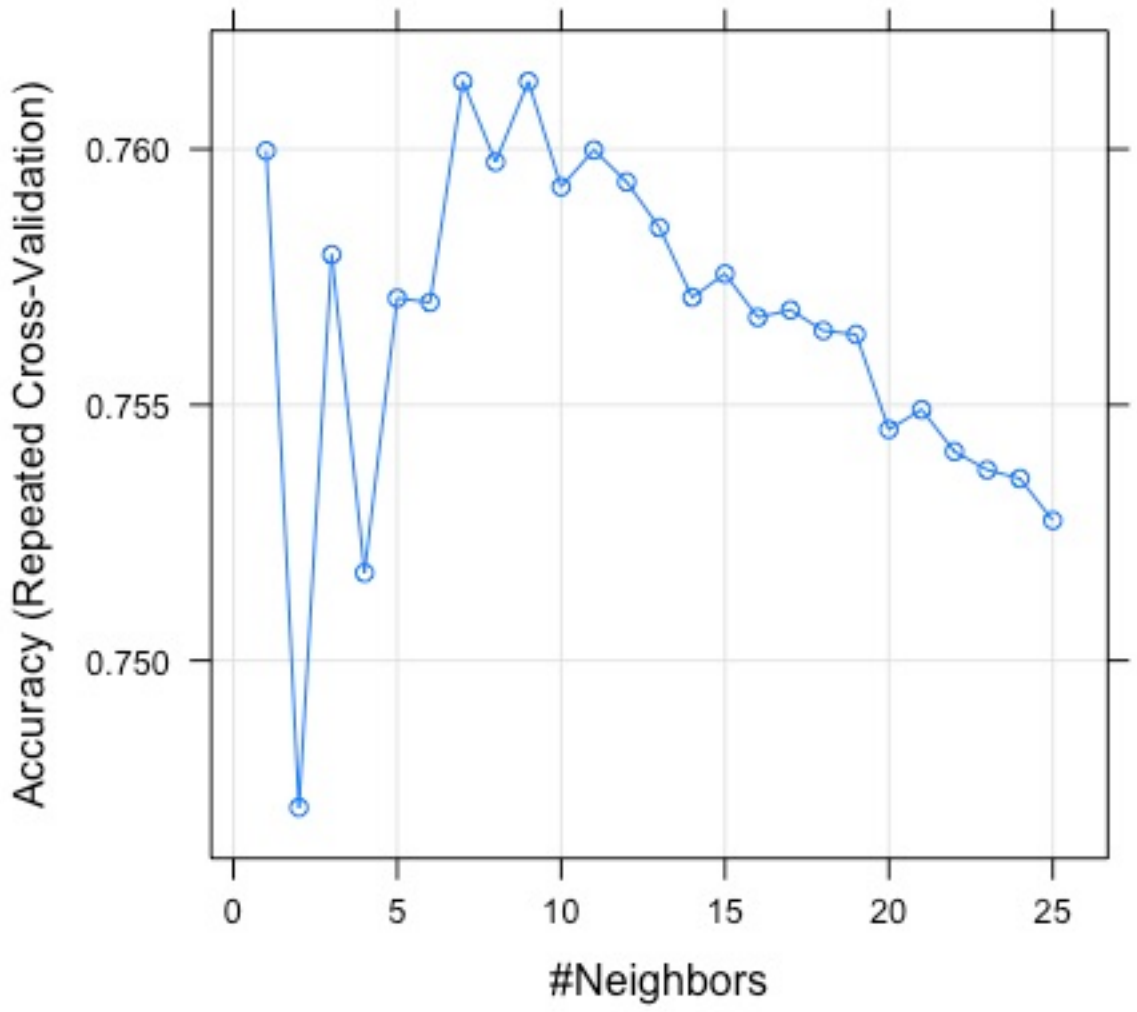
When KNN is used for classification in works in a way as for each data point the algorithm finds the k closest observations, and then classifies the data point to the majority. Usually, the k closest observations are defined as the ones with the smallest Euclidean distance to the data point under consideration. For example, if k = 3, and the three nearest observations to a specific data point belong to the classes A, B, and A respectively, the algorithm will classify the data point into class A. If k is even, there might be ties. To avoid this, usually weights are given to the observations, so that nearer observations are more influential in determining which class the data point belongs to. Our script uses k =7 to build the model for classification

The KNN model used testing, training dataset and the classification vector in the same step.

```
15
16  model <- train(
17    Base_Hour_Class~.,
18    data=split1,
19    method='knn',
20    tuneGrid=expand.grid(.k=1:25),
21    metric='Accuracy',
22    trControl=trainControl(
23      method='repeatedcv',
24      number=10,
25      repeats=15))
26
27
30:1  (Top Level)

Console ~/Documents/ADS/Assignment 3/

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 9.
> plot(model)
> confusionMatrix(model)
Cross-Validated (10 fold, repeated 15 times) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction High  Low
      High 42.2 12.4
      Low  11.5 33.9

Accuracy (average) : 0.7613
```

- The confusion matrix obtained for classification gives proper classification with an accuracy of 76%.

- The plot below shows the repeated cross validation (that is dividing the test and train data multiple times as required and cross validate it for every time) based on the value ok k. We see that for  k= 7 and 9 we get the maximum accuracy.




## Random Forest:

The random forest algorithm is like a normal decision tree except it has multiple decision trees and are built during the process. For a building a decision tree, samples of a data frame are selected with replacement along with selecting a subset of variables for each of the decision tree. Both sampling of data frame and

selection of subset of the variables are done randomly. Random forest algorithm produces high accuracy with reduced chances of over fitting.
Random Forest uses Gini Index based impurity measures for building decision tree.

```
rf.Model <- randomForest(Base_Hour_Class ~Temperature+area_floor._m.sqr+normalizedConsumption+hour+
                           Consumption+monthofYear+holiday
                         +Base_Hour_Flag+Consumption_base, ntree=400, data=train)

rf.Model.List[[ii]] <- rf.Model

test$pred <- predict(rf.Model, newdata = test)

predtable <- table(test$pred,test$Base_Hour_Class)

test$accuracy <- (predtable[1]+predtable[4])/nrow(test)
accuracy.data.frame$Accuracy[ii] <- test$accuracy
accuracy.data.frame$Model[ii] <- paste("Model",ii,sep ="")
#write.csv(test,paste("Model",ii,".csv",sep =""))

confusion.from.rf <- confusionMatrix(test$pred,test$Base_Hour_Class)
```

Parameters used in the code:

**x** : Random Forest Formula
**train**: Input data frame
**ntree**: Number of decision trees to be grown

# CLUSTERING:

## K means:

K mean clustering aims to partition n observations into k clusters in which each observation belongs to a particular cluster and results in the partition of the data space.

- The way it functions is it first select K centroids and then assign each data set to its closest centroids.
- Recalculates the centroids as the average of all data points in each cluster.
- Assign the data sets to their closest centroids
- Keep repeating these steps until the observations are not reaasigned or the maximum number of iterations are reached.

The resultant graph of K means clustering is shown below:

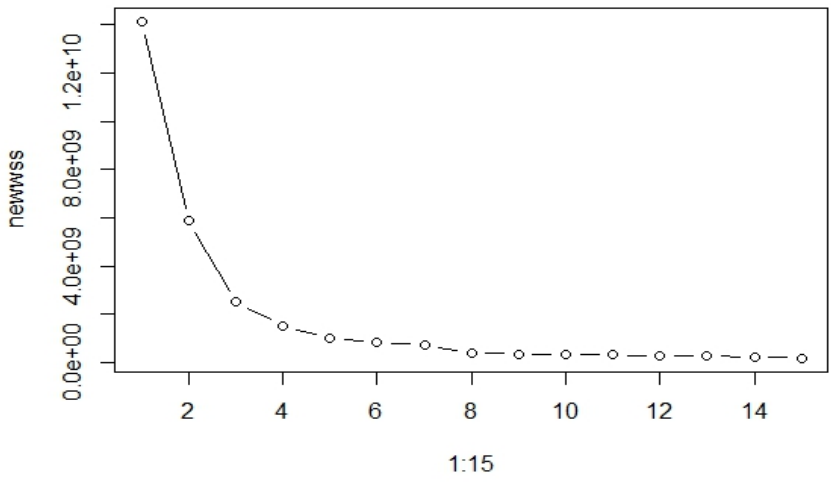
```
### KMEANS ON ENTIRE DATA SET#############################################################################
nrow(data.for.cluster)
str(data.for.cluster)
#View(data.for.cluster)
ncol(data.for.cluster)


## CONVERTING FACTORIAL COLUMNS TO MATRICES/ 1 MATRIX
datamatrix.for.cluster <- model.matrix(~.+0, data=data.for.cluster)
nrow(datamatrix.for.cluster)

## APPLYING KMEANS
data.for.kmeans <- kmeans(datamatrix.for.cluster, 2, nstart = 10)

##  MISC CHECKS
names(data.for.kmeans)
data.for.kmeans$betweenss
data.for.kmeans$size
data.for.kmeans
#plot(data.for.kmeans, col=(data.for.kmeans$cluster))

clustering.accuracy <- table(data.for.kmeans$cluster, data.for.cluster$Base_Hour_Class)
(clustering.accuracy[1]+clustering.accuracy[4])/nrow(data.for.cluster)
```

31:1 KMEANS ON ENTIRE DATA SET R Scr

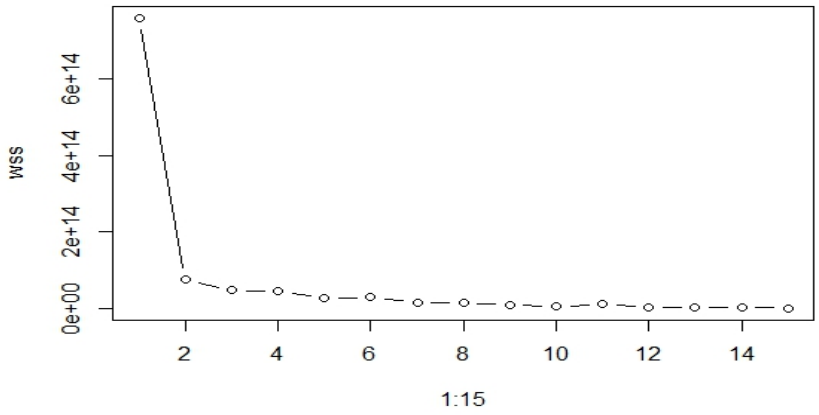
Console ~/Documents/ADS/Assignment 3/

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> clustering.accuracy <- table(data.for.kmeans$cluster, data.for.cluster$Base_Hour_Class)
> (clustering.accuracy[1]+clustering.accuracy[4])/nrow(data.for.cluster)
[1] 0.5449876
```

Bend graph plot for entire dataset:




For one specific model (out of 78):

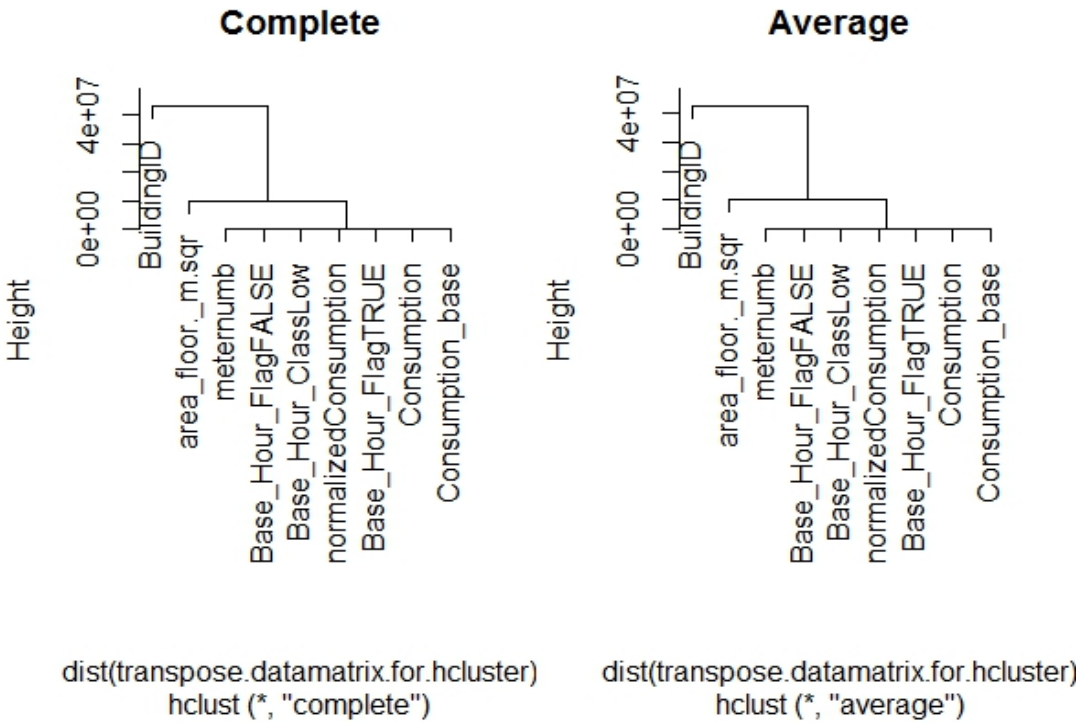



# Hierarchical clustering:

Hierarchical clustering does cluster analysis by building a hierarchy of clusters. The results of the clusters are presented in a dendrogram. The method used for hierarchical clustering is Agglomerative where we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps until there is only a single cluster left. The related algorithm is shown below.

Hierarchical clustering plot for complete data set

The first plot shows the dendrogram for both the complete method where at each step, the two clusters separated by the shortest distance are combined and Average method clustering where the distance between two clusters is defined as the average of distances between all pairs of objects, where each pair is made up of one object from each group.




For one specific model (out of 78):