Energy Star Score Prediction for NY Buildings

Data Importing

```
# Importing the required libraries
library(Hmisc)
library(caret)
library(corrplot)
library(Metrics)

memory.limit()
memory.limit(size=16000)
```

```
#Importing the dataset
data <- read.csv(file = 'Usecase1_Dataset.csv')
#summary (data)
dim(data)
```

```
## [1] 11746    60
```

Data Cleaning and Preprocessing

```
#There are 60 variables including a mix of discrete and continuous variables

#We can remove the unwanted columns like additional location details (Street number,
 street address) which is taken care by Postal Code variable

#Assumption 1 - Property id and other property details do not affect the energy score
as these are only identification details for record purposes

#Assumption 2 - Duplicate entries like 'Primary Property Type' can be omitted since t
he data is covered well in 'Largest property use type'

data_clean <- data[,11:60]
data_clean$Street.Number <- NULL
data_clean$Street.Name <- NULL
data_clean$List.of.All.Property.Use.Types.at.Property <- NULL
data_clean$Primary.Property.Type...Self.Selected <- NULL
```

```
#Postal code and borough seem to be correlated
#There are more missing values in Borough than in postal code,hence dropping Borough
 column
data_clean$Borough<- NULL
data_clean$NTA <- NULL
data_clean$DOF.Gross.Floor.Area <- NULL
data_clean$Release.Date <- NULL
```

```
#Removing latitude & longitude columns as we have postal code to group according to a
rea
data_clean <- data_clean[,1:(ncol(data_clean)-5)]

#Tackling rows with 'Not available' data for energy score (target variable)
#Assumption 3 - Removing data points with NaN values of target variable will not affe
ct the accuracy of model as the removed data points are much less
sum(data_clean$ENERGY.STAR.Score == "Not Available")
```

```
## [1] 2104
```

```
data_clean <- data_clean[-which(data_clean$ENERGY.STAR.Score == "Not Available"),]
```

```
#Dropping variables not normalized for weather and as we are taking (/ftA2) values in
stead of total value
data_clean$`Site.EUI..kBtu.ftÂ².` <- NULL
data_clean$`Source.EUI..kBtu.ftÂ².` <- NULL
data_clean$Total.GHG.Emissions..Metric.Tons.CO2e. <- NULL
data_clean$Weather.Normalized.Site.Electricity..kWh. <- NULL
data_clean$Water.Required. <- NULL
data_clean$Water.Use..All.Water.Sources...kgal. <- NULL
```

Remove data columns with more than 40% of 'Not Available' entries

```
cnt_NA <- rep(0,31)

for (i in 1:31){
  if (length(grep("Not Available", data_clean[,i])) > 0){
    cnt_NA[i] <- length(which(data_clean[,i] == "Not Available"))
  }
}
rm(i)

percent_NA <- (cnt_NA/nrow(data_clean))*100
df_NAcolumns <- cbind(colnames(data_clean), percent_NA)
rm(cnt_NA)

data_clean <- data_clean[,which(percent_NA <= 40)]
rm(percent_NA)
```

```
#Arrange columns per factored, continuous and target variable/s
data_clean <- subset(data_clean,select = c(1,2,7,8,21,3:6,10:20,9))
```

```
#Converting Postal code data to valid data, truncating first five digits (as is the s
tandard format)
data_clean$Postal.Code <- substr(data_clean$Postal.Code, 1,5)
```

```
#Convert the variables to factors and numericals
for (i in 1:5){
  data_clean[,i] <- as.factor(data_clean[,i])
}

for (i in 6:21){
  data_clean[,i] <- as.numeric(data_clean[,i])
}
rm(i)

summary(data_clean)
```

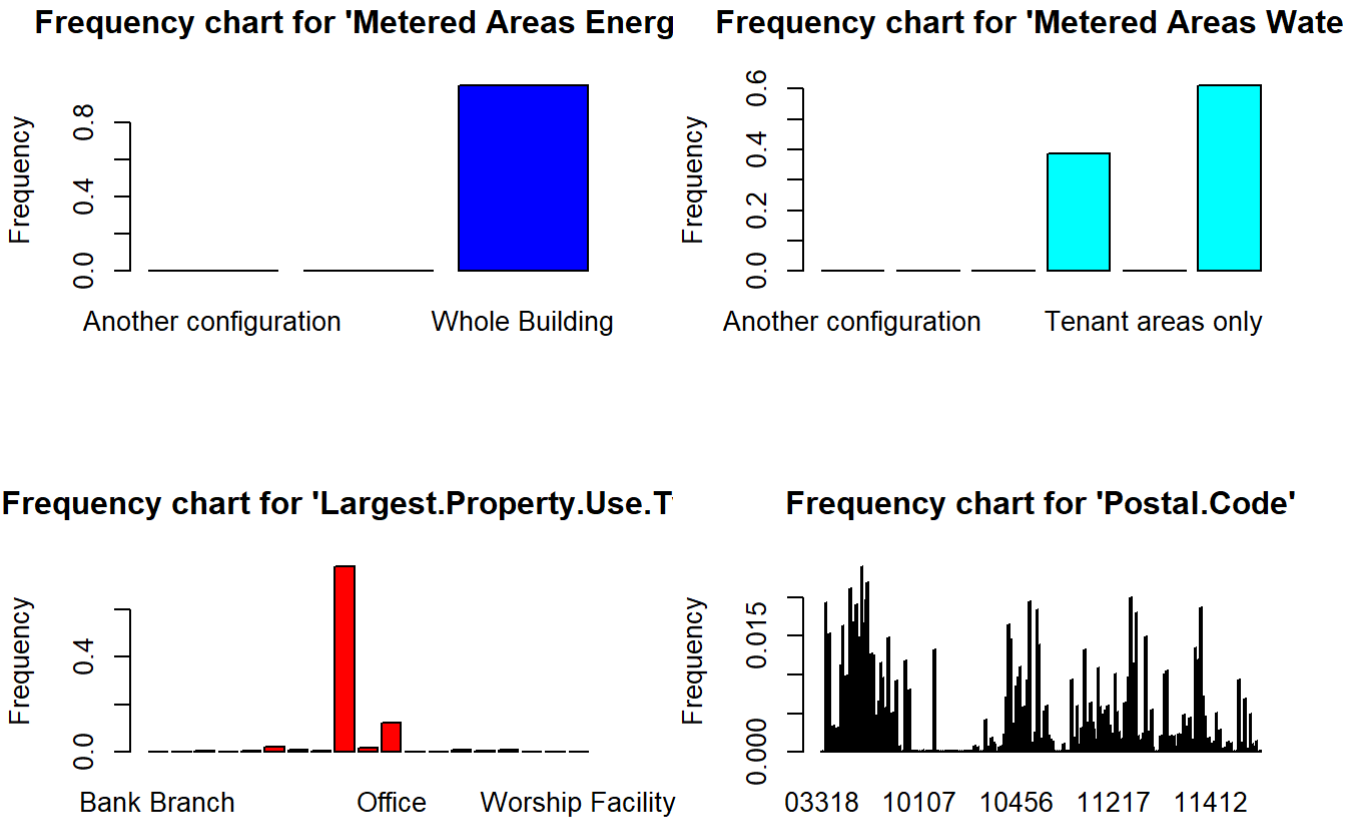
```
##   Postal.Code                Largest.Property.Use.Type
##  10022  : 231   Multifamily Housing       :7529
##  10025  : 211   Office                    :1193
##  10016  : 204   Hotel                     : 215
##  11226  : 192   Non-Refrigerated Warehouse: 159
##  10024  : 190   K-12 School               :  98
##  10463  : 187   Residence Hall/Dormitory  :  97
##  (Other):8427   (Other)                   : 351
##            Metered.Areas..Energy.
##  Another configuration:  16
##  Not Available        :   4
##  Whole Building       :9622
##
##
##
##
##                             Metered.Areas...Water.
##  Another configuration                :   1
##  Combination of common and tenant areas:   8
##  Common areas only                    :   4
##  Not Available                        :3741
##  Tenant areas only                    :   2
##  Whole Building                       :5886
##
##  DOF.Benchmarking.Submission.Status
##               :  26
##  In Compliance:9616
##
##
##
##
##
##  Largest.Property.Use.Type...Gross.Floor.Area..ftÂ².   Year.Built
##  Min.   :    3190                                     Min.   :1600
##  1st Qu.:   65780                                     1st Qu.:1927
##  Median :   90642                                     Median :1941
##  Mean   :  157391                                     Mean   :1948
##  3rd Qu.:  150195                                     3rd Qu.:1965
##  Max.   :14217119                                     Max.   :2019
##
##  Number.of.Buildings...Self.reported   Occupancy
##  Min.   :  0.000                     Min.   :  0.00
##  1st Qu.:  1.000                     1st Qu.:100.00
##  Median :  1.000                     Median :100.00
##  Mean   :  1.284                     Mean   : 99.03
##  3rd Qu.:  1.000                     3rd Qu.:100.00
##  Max.   :155.000                     Max.   :100.00
##
##  Weather.Normalized.Site.EUI..kBtu.ftÂ².
##  Min.   :     0.0
##  1st Qu.:    66.3
##  Median :    82.9
##  Mean   :   260.7
##  3rd Qu.:   102.2
##  Max.   :939329.0
```

```
##  NA's   :1098
##  Weather.Normalized.Site.Electricity.Intensity..kWh.ftÂ².
##  Min.   :   0.000
##  1st Qu.:   3.800
##  Median :   5.100
##  Mean   :   9.524
##  3rd Qu.:   8.700
##  Max.   :5304.400
##  NA's   :492
##  Weather.Normalized.Site.Natural.Gas.Intensity..therms.ftÂ².
##  Min.   :   0.000
##  1st Qu.:   0.100
##  Median :   0.500
##  Mean   :   1.655
##  3rd Qu.:   0.700
##  Max.   :9393.000
##  NA's   :1413
##  Weather.Normalized.Source.EUI..kBtu.ftÂ². Natural.Gas.Use..kBtu.
##  Min.   :     0.0                         Min.   :0.000e+00
##  1st Qu.:   104.1                         1st Qu.:1.237e+06
##  Median :   128.8                         Median :4.207e+06
##  Mean   :   351.5                         Mean   :1.266e+07
##  3rd Qu.:   165.0                         3rd Qu.:6.841e+06
##  Max.   :986366.0                         Max.   :4.946e+10
##  NA's   :1098                             NA's   :1002
##  Weather.Normalized.Site.Natural.Gas.Use..therms.
##  Min.   :        0
##  1st Qu.:    13041
##  Median :    45381
##  Mean   :   138309
##  3rd Qu.:    73422
##  Max.   :534458969
##  NA's   :1413
##  Electricity.Use...Grid.Purchase..kBtu. Direct.GHG.Emissions..Metric.Tons.CO2e.
##  Min.   :0.000e+00                      Min.   :      0.0
##  1st Qu.:1.021e+06                      1st Qu.:    169.3
##  Median :1.771e+06                      Median :    280.5
##  Mean   :5.714e+06                      Mean   :    714.0
##  3rd Qu.:4.111e+06                      3rd Qu.:    447.9
##  Max.   :1.692e+09                      Max.   :2627015.0
##  NA's   :77                             NA's   :13
##  Indirect.GHG.Emissions..Metric.Tons.CO2e. Property.GFA...Self.Reported..ftÂ².
##  Min.   : -23134                           Min.   :     120
##  1st Qu.:     94                           1st Qu.:   66917
##  Median :    165                           Median :   92584
##  Mean   :   1572                           Mean   :  162367
##  3rd Qu.:    400                           3rd Qu.:  154000
##  Max.   :4764375                           Max.   :14217119
##  NA's   :9
##  Water.Intensity..All.Water.Sources...gal.ftÂ². ENERGY.STAR.Score
##  Min.   :    0.00                              Min.   :  1.00
##  1st Qu.:   29.37                              1st Qu.: 37.00
##  Median :   47.19                              Median : 65.00
##  Mean   :  108.21                              Mean   : 59.85
##  3rd Qu.:   73.17                              3rd Qu.: 85.00
```

```
##  Max.    :21689.36                       Max.    :100.00
##  NA's    :3114
```

Understanding class distribution of levels for discrete variables As can be seen in the proportion table, 'Metered areas Energy' column has skewed ratio of the factors and very less data points for 'another config' and 'not available', hence removing these rows (20 count) and the column Similarly with 'Metered areas water', removing the rows with factors of less data points and removing the column

```
#Bar Plot for Discrete Variables
par(mfrow=c(2,2))
barplot(prop.table(table(data_clean$Metered.Areas..Energy.)), ylab = "Frequency",
        main = "Frequency chart for 'Metered Areas Energy'", col = "blue")
barplot(prop.table(table(data_clean$Metered.Areas...Water.)), ylab = "Frequency",
        main = "Frequency chart for 'Metered Areas Water'", col = "Cyan")
barplot(prop.table(table(data_clean$Largest.Property.Use.Type)), ylab = "Frequency",
        main = "Frequency chart for 'Largest.Property.Use.Type'", col = "Red")
barplot(prop.table(table(data_clean$Postal.Code)), ylab = "Frequency",
        main = "Frequency chart for 'Postal.Code'", col = "Black")
```




```
par(mfrow=c(1,1))
```

```
#Cleaning Metered Areas Energy column
data_clean <- data_clean[which(data_clean$Metered.Areas..Energy. == "Whole Building"
),]
data_clean$Metered.Areas..Energy. <- NULL
```

```
#Cleaning 'Metered Area Water' column
data_clean <- data_clean[which(data_clean$Metered.Areas...Water. == "Not Available" |
data_clean$Metered.Areas...Water. == "Whole Building"), ]
data_clean$Metered.Areas...Water. <- NULL
```

```
#Cleaning 'Largest Property Use Type' column
table(data_clean$Largest.Property.Use.Type )
```

```
##
##                          Bank Branch                          Courthouse
##                                    2                                   2
##                   Distribution Center                     Financial Office
##                                   61                                  14
## Hospital (General Medical & Surgical)                               Hotel
##                                   40                                 212
##                           K-12 School                       Medical Office
##                                   98                                  29
##                   Multifamily Housing           Non-Refrigerated Warehouse
##                                 7503                                 159
##                               Office                             Parking
##                                 1191                                   3
##               Refrigerated Warehouse             Residence Hall/Dormitory
##                                    9                                  96
##                         Retail Store                 Senior Care Community
##                                   64                                  92
##             Supermarket/Grocery Store           Wholesale Club/Supercenter
##                                   19                                   4
##                      Worship Facility
##                                    9
```

```
data_clean <- data_clean[which(data_clean$Largest.Property.Use.Type == "Multifamily H
ousing" |
                                 data_clean$Largest.Property.Use.Type == "Office" |
                                 data_clean$Largest.Property.Use.Type == "Hotel" |
                                 data_clean$Largest.Property.Use.Type == "Non-Refrige
rated Warehouse"),]
data_clean$Largest.Property.Use.Type <- factor(data_clean$Largest.Property.Use.Type)

str(data_clean)
```

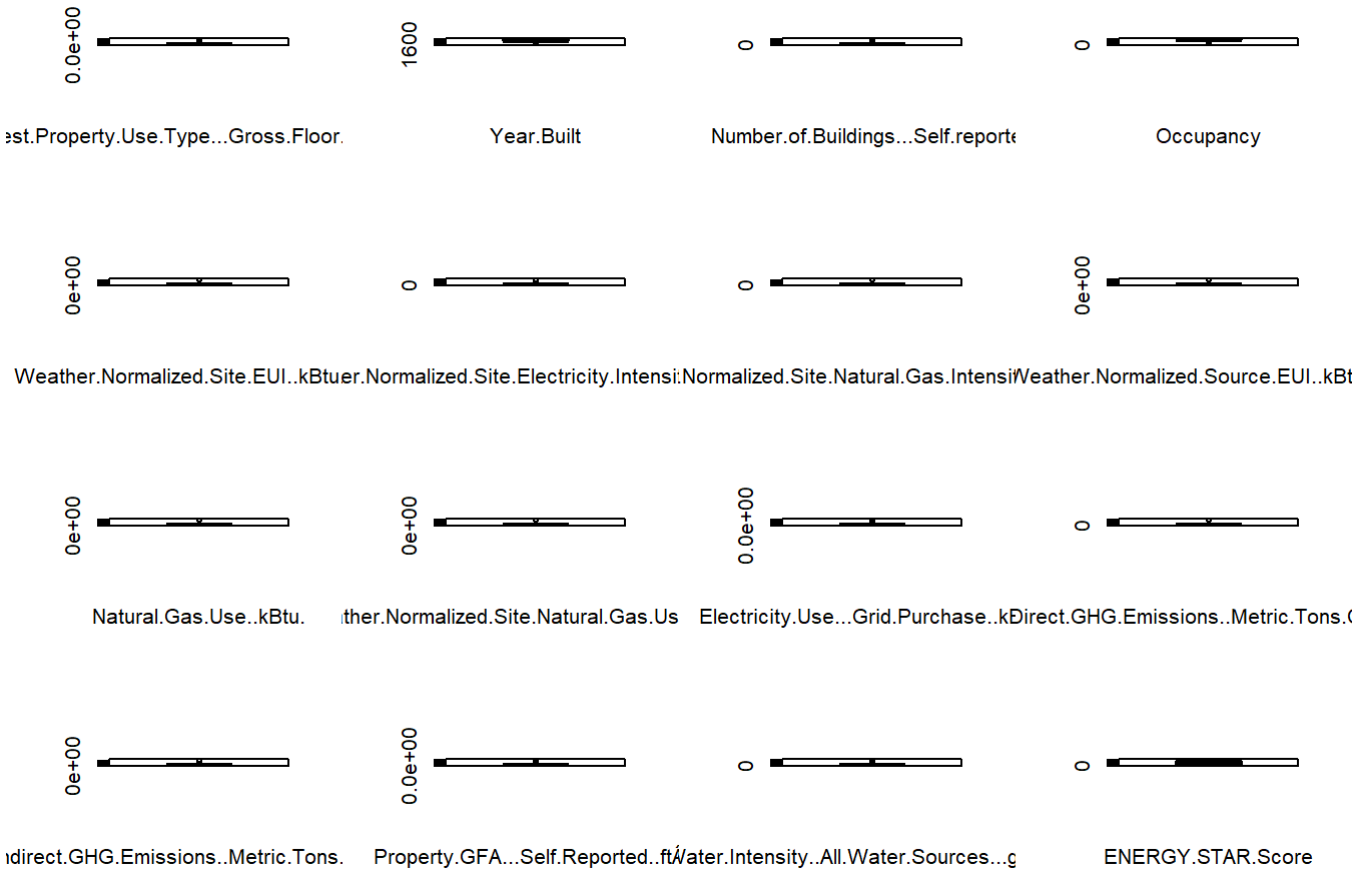
```
## 'data.frame':    9065 obs. of  19 variables:
##  $ Postal.Code                                          : Factor w/ 231 leve
ls "03318","10000",..: 139 139 40 40 40 21 29 216 216 216 ...
##  $ Largest.Property.Use.Type                            : Factor w/ 4 levels
"Hotel","Multifamily Housing",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ DOF.Benchmarking.Submission.Status                   : Factor w/ 2 levels
"","In Compliance": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Largest.Property.Use.Type...Gross.Floor.Area..ftÂ².   : num  412503 162656
321464 204720 116744 ...
##  $ Year.Built                                           : num  1903 1903 193
9 1939 1939 ...
##  $ Number.of.Buildings...Self.reported                  : num  2 1 2 1 1 2 2
4 6 4 ...
##  $ Occupancy                                            : num  100 100 100 1
00 100 100 100 100 100 100 ...
##  $ Weather.Normalized.Site.EUI..kBtu.ftÂ².              : num  NA 316.8 79.9
79 82.7 ...
##  $ Weather.Normalized.Site.Electricity.Intensity..kWh.ftÂ².   : num  7.5 5.8 3.5 4
2.6 6.1 10.7 7.4 4.9 12.4 ...
##  $ Weather.Normalized.Site.Natural.Gas.Intensity..therms.ftÂ².: num  0 3 0 0 0 0.5
0.2 0.3 0.1 0.2 ...
##  $ Weather.Normalized.Source.EUI..kBtu.ftÂ².            : num  NA 374 106 10
9 103 ...
##  $ Natural.Gas.Use..kBtu.                               : num  2047200 47522
901 880200 553439 326761 ...
##  $ Weather.Normalized.Site.Natural.Gas.Use..therms.     : num  20472 512711
8906 5604 3302 ...
##  $ Electricity.Use...Grid.Purchase..kBtu.               : num  11241926 3477
065 3968642 2886852 1081789 ...
##  $ Direct.GHG.Emissions..Metric.Tons.CO2e.              : num  151 2524 1525
936 589 ...
##  $ Indirect.GHG.Emissions..Metric.Tons.CO2e.            : num  1043 323 368
268 100 ...
##  $ Property.GFA...Self.Reported..ftÂ².                  : num  432503 172656
321464 204720 116744 ...
##  $ Water.Intensity..All.Water.Sources...gal.ftÂ².       : num  51 NA 18.4 NA
NA ...
##  $ ENERGY.STAR.Score                                    : num  93 1 72 67 80
100 40 67 99 12 ...
```

Removing out lier from continuous variables and replacing missing values with average values

```
#Plot box plot for each continuous variable
par(mfrow=c(4,4))
ncol(data_clean)
```

```
## [1] 19
```

```
for (i in 4:19){
  boxplot(data_clean[,i], xlab = colnames(data_clean)[i])
}
```



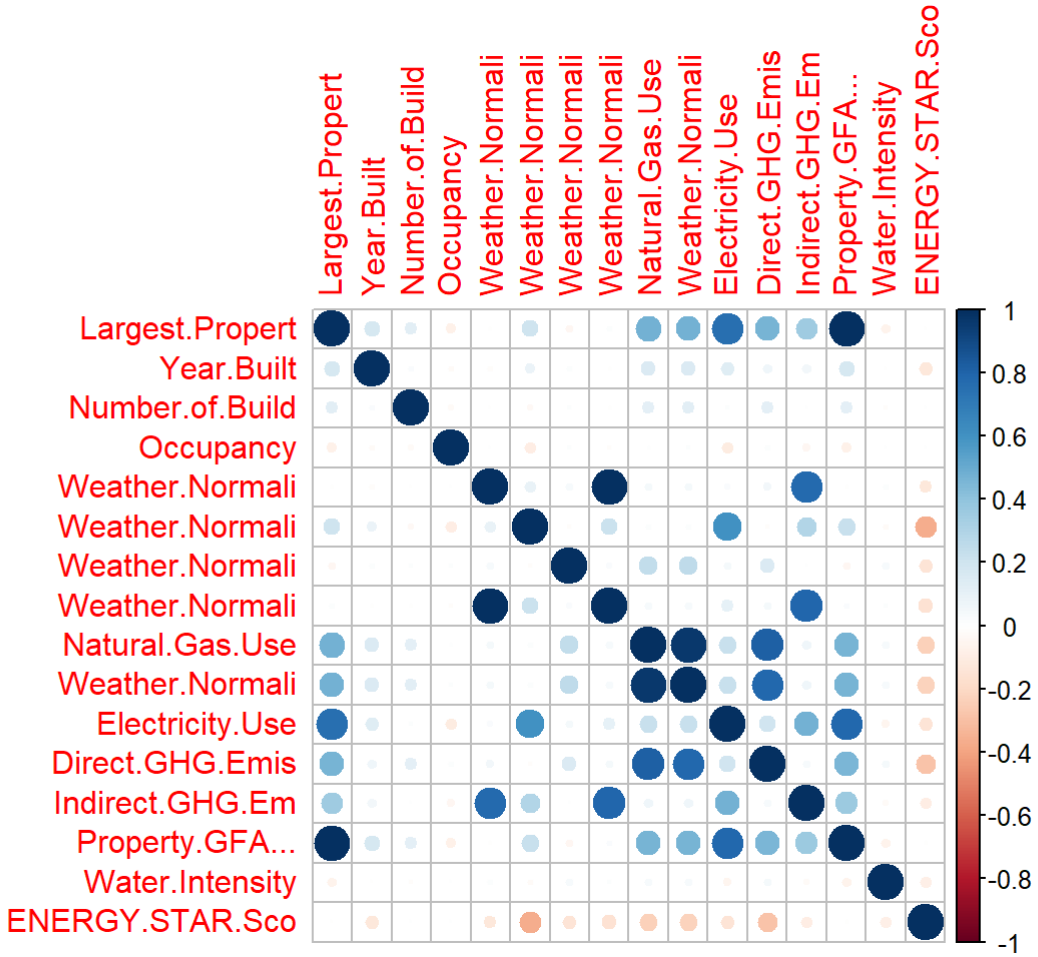

```
par(mfrow=c(1,1))
```

```
#Find absolute value of z-score for each value in each column
df <- data_clean[,4:19]
z_scores <- as.data.frame(sapply(df, function(df) (abs(df-mean(df, na.rm = TRUE))/sd
(df, na.rm = TRUE))))
data_clean <- data_clean[!rowSums(z_scores>3, na.rm = TRUE), ]

for (i in 1:19){
  if(length(which(is.na(data_clean[,i]))=="TRUE")>0){
  data_clean[which(is.na(data_clean[,i])),i] <- median(data_clean[,i], na.rm = TRUE)
  }
}
rm(i)
rm(df)
```

Correlation Matrix for feature selection

```
cor_matrix <- data_clean
data_clean.cor <- cor(cor_matrix[4:19])
data_clean.cor.plot <- cor(cor_matrix[4:19])
colnames(data_clean.cor.plot) <- substring(colnames(data_clean[,4:19]),1,15)
rownames(data_clean.cor.plot) <- substring(colnames(data_clean[,4:19]),1,15)
corrplot(data_clean.cor.plot)
```




```
highlyCorrelated <- findCorrelation(data_clean.cor, cutoff=0.75, names = TRUE)

#Removing highly correlated columns
data_clean <- data_clean[,-which(colnames(data_clean) %in% highlyCorrelated)]
```

Renaming columns

```
length(colnames(data_clean))
```

```
## [1] 13
```

```
names_col <- c("PostalCode","LargeUse", "SubmissionStatus",
                           "BuiltYear","CountofBuildings", "Occupancy",
                           "WeatherNormSiteEUI", "WeatherNormElecEUI", "WeatherNormNGE
UI",
                           "ElecUse","DirectGHG","WaterIntensity","EnergyScore")
length(names_col)
```

```
## [1] 13
```

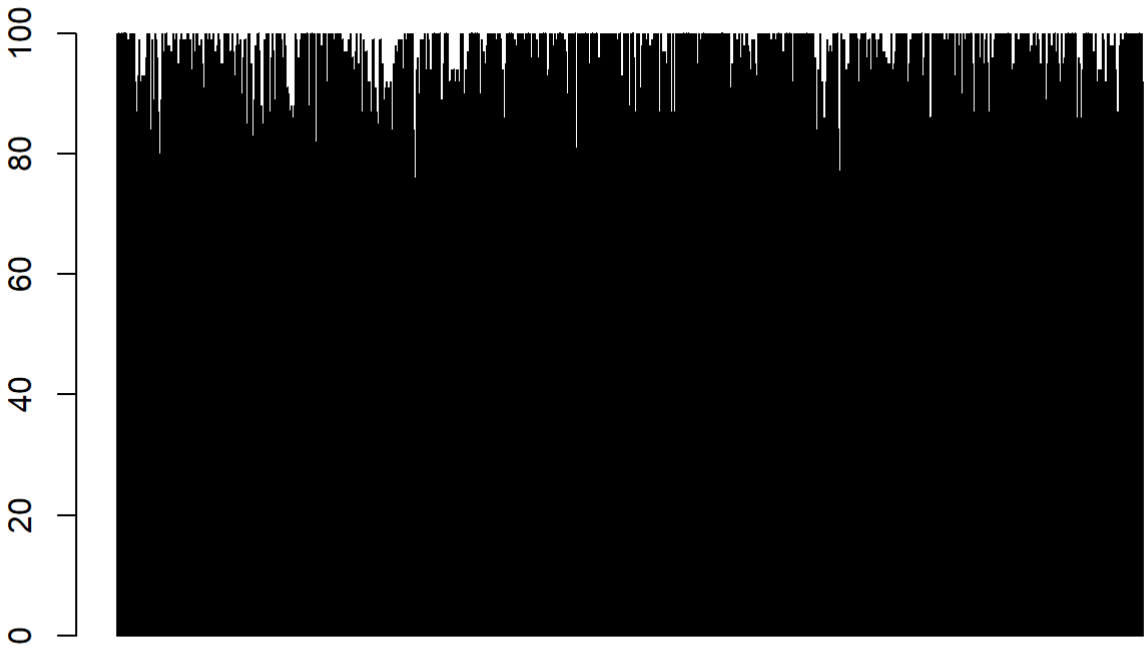
```
colnames(data_clean) <- names_col
rm(names_col)
```

```
#Remove duplicate entries
data_clean <- unique(data_clean)

str(data_clean)
```

```
## 'data.frame':    8729 obs. of  13 variables:
##  $ PostalCode        : Factor w/ 231 levels "03318","10000",..: 139 139 40 40 40 2
1 29 216 216 216 ...
##  $ LargeUse          : Factor w/ 4 levels "Hotel","Multifamily Housing",..: 2 2 2
2 2 2 2 2 2 2 ...
##  $ SubmissionStatus  : Factor w/ 2 levels "","In Compliance": 2 2 2 2 2 2 2 2 2 2
...
##  $ BuiltYear         : num  1903 1903 1939 1939 1939 ...
##  $ CountofBuildings  : num  2 1 2 1 1 2 2 4 6 4 ...
##  $ Occupancy         : num  100 100 100 100 100 100 100 100 100 100 ...
##  $ WeatherNormSiteEUI: num  82.7 316.8 79.9 79 82.7 ...
##  $ WeatherNormElecEUI: num  7.5 5.8 3.5 4 2.6 6.1 10.7 7.4 4.9 12.4 ...
##  $ WeatherNormNGEUI  : num  0 3 0 0 0 0.5 0.2 0.3 0.1 0.2 ...
##  $ ElecUse           : num  11241926 3477065 3968642 2886852 1081789 ...
##  $ DirectGHG         : num  151 2524 1525 936 589 ...
##  $ WaterIntensity    : num  51 48 18.4 48 48 ...
##  $ EnergyScore       : num  93 1 72 67 80 100 40 67 99 12 ...
```

```
#Visualizing the distribution of Target variable for any skewed distribution
barplot(data_clean$EnergyScore)
```



Split data in train and test in 70-30

```
dt <- sort(sample(nrow(data_clean), nrow(data_clean)*.7))

train_data <- data_clean[dt,]
test_data <- data_clean[-dt,]
rm(dt)
```

scaling variables for uniform range. As the range for EnergyScore is not much, we will scale only the X variables and not target variable

```
train_data[,4:12] <- scale(train_data[,4:12])
test_data[,4:12] <- scale(test_data[,4:12])
```

```
#Removing the Submission Status column in train data. This column can later be used t
o predict only 'Not Submitted' Entries in test data

train_data_submission_col <- train_data$SubmissionStatus
train_data$SubmissionStatus <- NULL

test_data_submission_col <- test_data$SubmissionStatus
test_data$SubmissionStatus <- NULL

str(train_data)
```

```
## 'data.frame':    6110 obs. of  12 variables:
##  $ PostalCode        : Factor w/ 231 levels "03318","10000",..: 139 40 40 40 21 29
216 216 216 27 ...
##  $ LargeUse          : Factor w/ 4 levels "Hotel","Multifamily Housing",..: 2 2 2
2 2 2 2 2 2 4 ...
##  $ BuiltYear         : num  -1.541 -0.294 -0.294 -0.294 -1.264 ...
##  $ CountofBuildings  : num  -0.128 1.413 -0.128 -0.128 1.413 ...
##  $ Occupancy         : num  0.228 0.228 0.228 0.228 0.228 ...
##  $ WeatherNormSiteEUI: num  5.7229 -0.1468 -0.1691 -0.0774 -0.0774 ...
##  $ WeatherNormElecEUI: num  -0.1492 -0.5563 -0.4678 -0.7156 -0.0961 ...
##  $ WeatherNormNGEUI  : num  6.306761 -1.260312 -1.260312 -1.260312 0.000867 ...
##  $ ElecUse           : num  -0.0462 0.0265 -0.1336 -0.4006 0.0332 ...
##  $ DirectGHG         : num  4.58 2.434 1.17 0.423 0.179 ...
##  $ WaterIntensity    : num  -0.127 -0.55 -0.127 -0.127 -0.127 ...
##  $ EnergyScore       : num  1 72 67 80 100 40 99 12 88 100 ...
```

```
str(test_data)
```

```
## 'data.frame':    2619 obs. of  12 variables:
##  $ PostalCode        : Factor w/ 231 levels "03318","10000",..: 139 216 32 27 39 3
9 39 191 191 191 ...
##  $ LargeUse          : Factor w/ 4 levels "Hotel","Multifamily Housing",..: 2 2 4
2 2 2 2 2 2 ...
##  $ BuiltYear         : num  -1.535 0.12 0.728 0.526 -0.691 ...
##  $ CountofBuildings  : num  1.423 4.497 -0.113 -0.113 -0.113 ...
##  $ Occupancy         : num  0.227 0.227 0.227 0.227 0.227 ...
##  $ WeatherNormSiteEUI: num  -0.02578 -0.0993 -0.00947 0.01581 0.0142 ...
##  $ WeatherNormElecEUI: num  0.128 0.109 0.544 -1.05 -0.524 ...
##  $ WeatherNormNGEUI  : num  -0.311 -0.134 -0.311 0.277 0.218 ...
##  $ ElecUse           : num  1.1263 0.0126 -0.1395 -0.5352 -0.4281 ...
##  $ DirectGHG         : num  -0.518 -0.424 -0.142 0.146 -0.148 ...
##  $ WaterIntensity    : num  -0.0923 -0.1427 -0.5926 0.3889 -0.1427 ...
##  $ EnergyScore       : num  93 67 81 58 34 43 1 96 100 100 ...
```

Modeling with all variable and understanding each variable's significance

```
#10 fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "RMSE"
```

```
#Simple rpart - decision tree
set.seed(5)
fit.rpart <- train(EnergyScore~., data=train_data, method="rpart",metric = "RMSE",
                   trControl=control, tuneLength = 15)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
fit.rpart
```

```
## CART
##
## 6110 samples
##   11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5500, 5499, 5499, 5499, 5499, 5499, ...
## Resampling results across tuning parameters:
##
##   cp           RMSE      Rsquared   MAE
##   0.005405593  17.01398  0.6724382  12.81309
##   0.005420461  17.02005  0.6721771  12.81705
##   0.006116909  17.27968  0.6619349  13.07038
##   0.006731918  17.51184  0.6529023  13.35942
##   0.009554129  17.62442  0.6480408  13.45712
##   0.011815389  17.84454  0.6390578  13.63731
##   0.013023533  18.48504  0.6130145  14.11862
##   0.013300969  18.48504  0.6130145  14.11862
##   0.016835322  18.89166  0.5957877  14.53839
##   0.024893900  20.03178  0.5450510  15.56711
##   0.029730893  20.49968  0.5237451  16.00393
##   0.042827866  21.31285  0.4848316  16.74388
##   0.044315596  21.71114  0.4657043  17.14159
##   0.069681338  23.10159  0.3950501  18.57441
##   0.364503036  26.96402  0.3366081  22.49094
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.005405593.
```

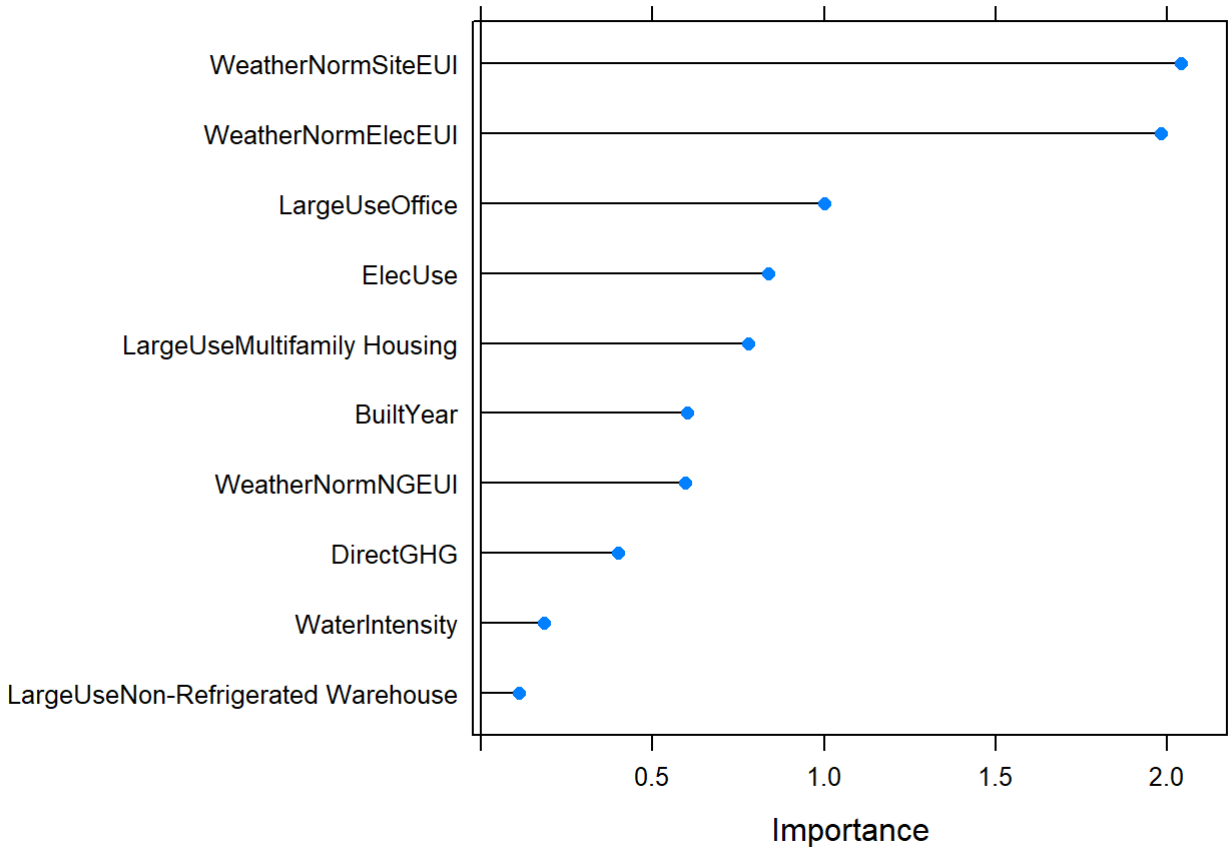
```
#MAE with decision tree = 13
```

```
#xgboost
#set.seed(7)
#fit.boost <- train(EnergyScore~., data=train_data, method="xgbTree")
#fit.boost

#knn
#This algorithm is not run due to resource limitations
#set.seed(5)
#fit.knn <- train(EnergyScore~., data=train_data, method="knn")

#random forest
#This algorithm is not run due to resource limitations
#set.seed(5)
##fit.rf <- train(EnergyScore~., data=train_data, method="rf")
```

```
#Based on the comparative results of 2 algorithms (MAE and r-squared), we can go ahea
d with xGBoost
#Further the accuracy of the selected model can be improved with Cross validation and
tuning the hyper parameters for XGBoost
#It is not completed in this exercise due to resource limitations on the personal com
puter
#The model can be further improved by removing the non-important variables as follow
s. This assignment does not cover it due to time constraints.
Importance <- varImp(fit.rpart, scale=FALSE)
plot(Importance, top = 10)
```



Test the data with selected model

```
predictions <- predict(fit.rpart, test_data[,1:11])
MAE(test_data$EnergyScore,predictions)
```

```
## [1] 19.08936
```

Observations

```
#MAE for train data = 13
#MAE for test data = 19
#As test model MAE is about 2 times the MAE of train model, we can understand the mod
el is overfit
#We can perform k-fold cv to tune the parameters for xGBoost and run random forest fo
r further improvement in the trained model
#We will skip this step due to time limitation and resource limitations
```

Improving the model accuracy (difference between the MAE of train and test and MAE of test itself)

```
#Build dataset with important variable contributing more than 95% (drop postal code v
ariable)
train_data_mod <- train_data[,2:12]
test_data_mod <- test_data[,2:12]

train_data_mod$SubmissionStatus <- NULL
test_data_mod$SubmissionStatus <- NULL

set.seed(7)
fit.rpart.mod <- train(EnergyScore~., data=train_data_mod, method="rpart")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
fit.rpart.mod
```

```
## CART
##
## 6110 samples
##   10 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 6110, 6110, 6110, 6110, 6110, 6110, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE      Rsquared   MAE
##   0.04431560  21.66234  0.4681818  17.12230
##   0.06968134  23.17899  0.3909414  18.68961
##   0.36450304  26.46898  0.3520300  21.98760
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.0443156.
```

```
#set.seed(7)
#fit.boost.mod <- train(EnergyScore~., data=train_data_mod, method="xgbTree")
#fit.boost.mod

predictions.mod <- predict(fit.rpart.mod, test_data_mod[,1:10])
MAE(test_data_mod$EnergyScore,predictions)
```

```
## [1] 19.08936
```

Conclusion - We can derive a better regression model by running xGboost and tuning the parameters further to reduce overfitting.

We can explore clustering approach for possible accuracy improvement by creating similar subsets by location (weather) and building demography

The End