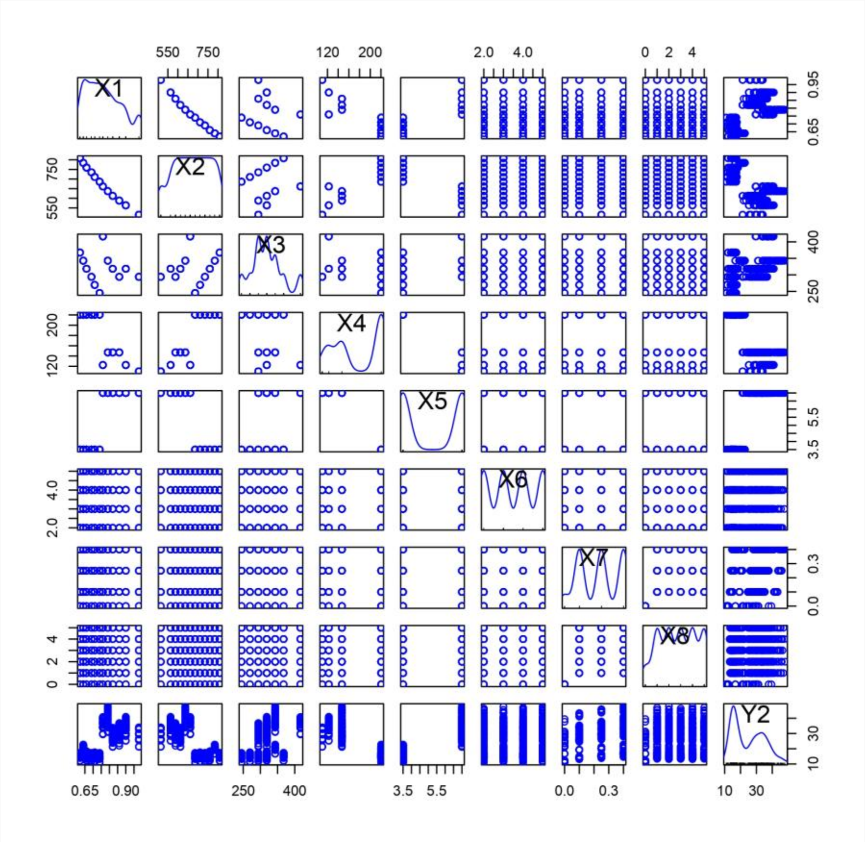
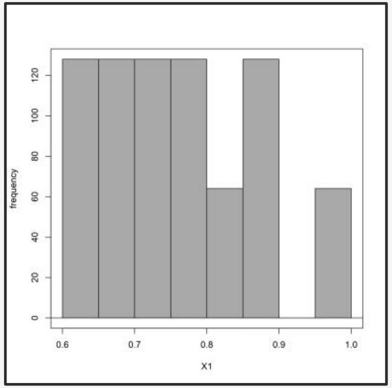
Energy Efficiency Analysis

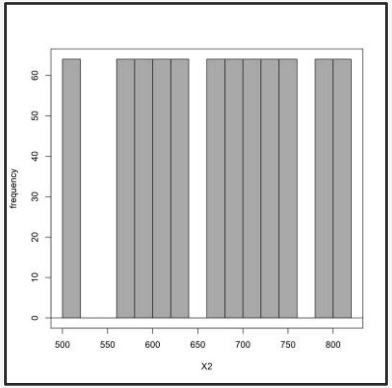
Group 6 : ChihHao Luca Yuan, Ching Yu Hsu

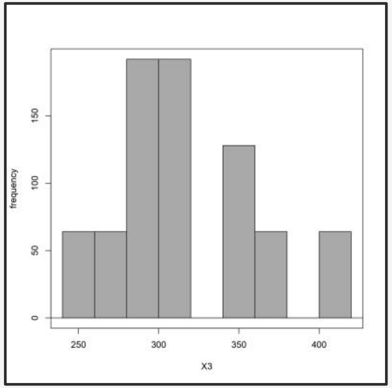
1a)- Scatterplots to display relationships

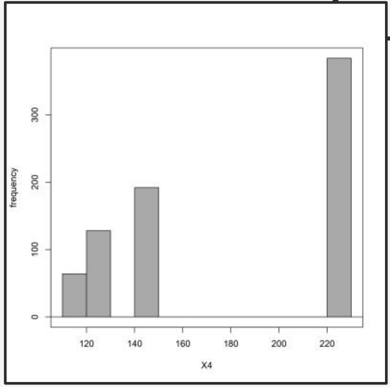


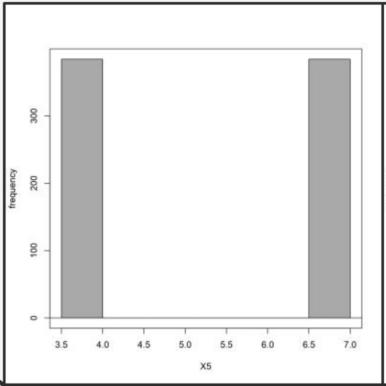
1b)- Histogram for each variable

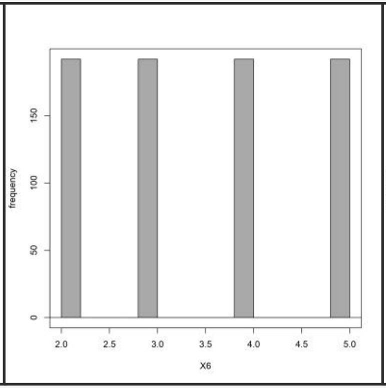


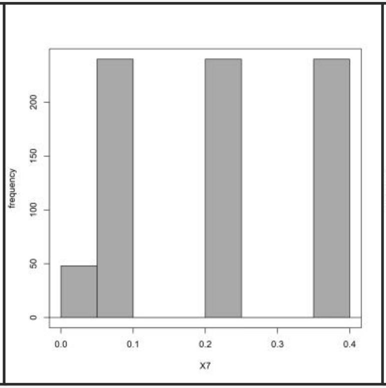


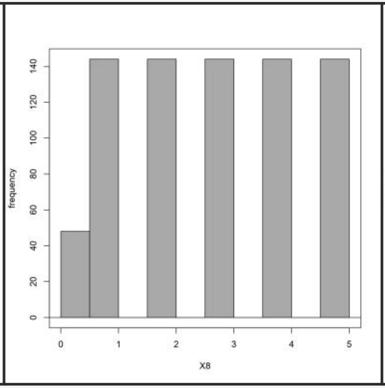


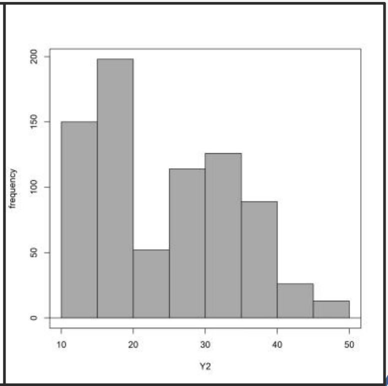


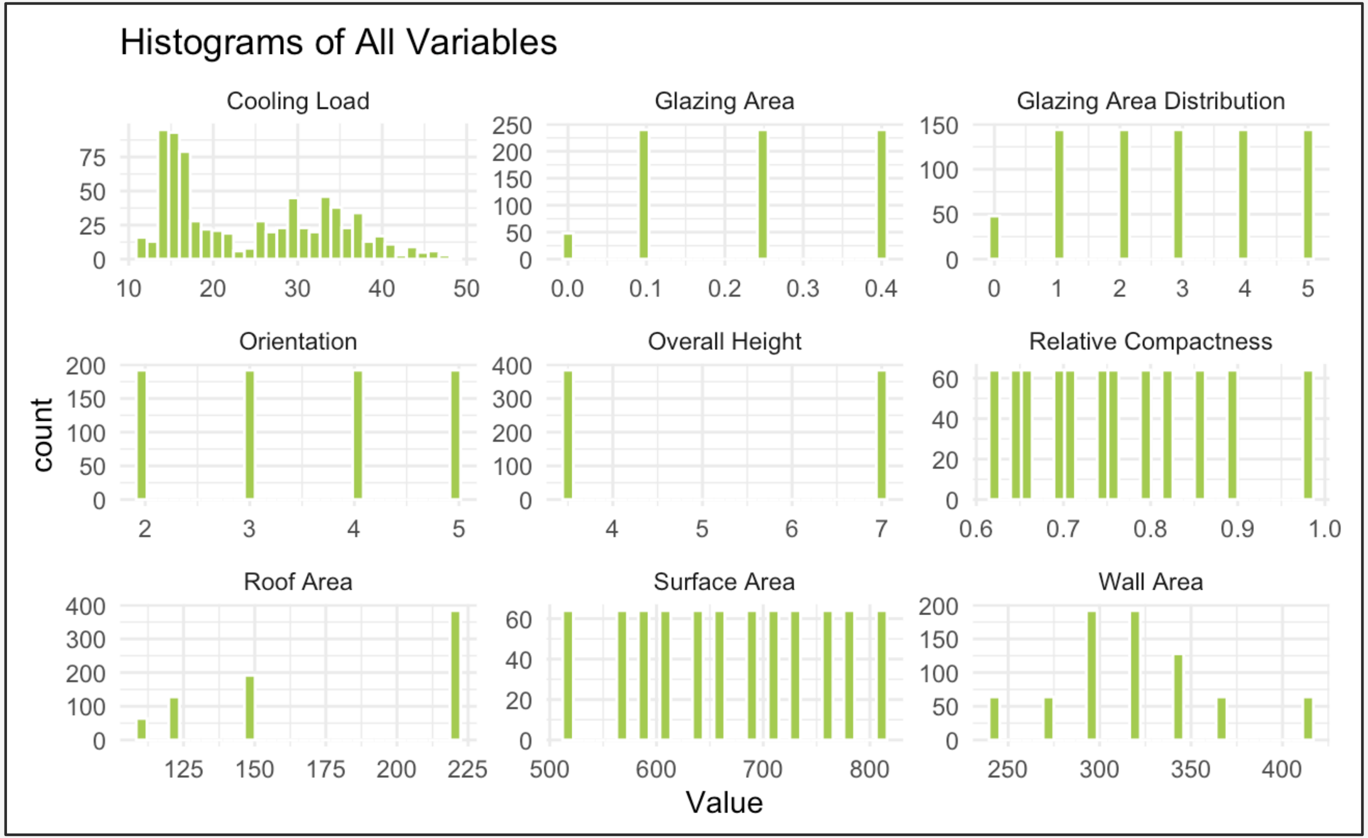










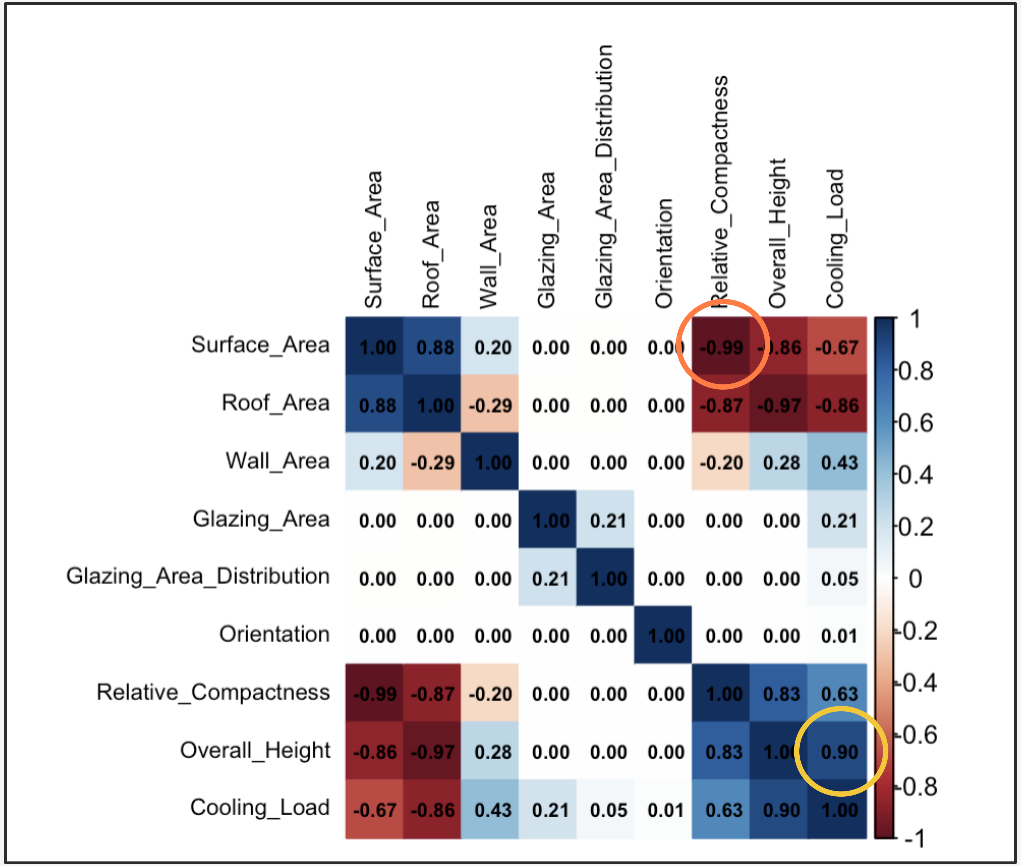


Histograms of All Variables
Cooling Load
Glazing Area
Glazing Area Distribution
Orientation
Overall Height
Relative Compactness
Roof Area
Surface Area
Wall Area
count
Value

2a)- Descriptive statistics

	mean	sd	IQR	0%	25%	50%	75%	100%	n
Cooling_Load	24.5877604	9.5133056	17.5125	10.90	15.6200	22.08	33.1325	48.03	768
Glazing_Area	0.2343750	0.1332206	0.3000	0.00	0.1000	0.25	0.4000	0.40	768
Glazing_Area_Distribution	2.8125000	1.5509597	2.2500	0.00	1.7500	3.00	4.0000	5.00	768
Orientation	3.5000000	1.1187626	1.5000	2.00	2.7500	3.50	4.2500	5.00	768
Overall_Height	5.2500000	1.7511404	3.5000	3.50	3.5000	5.25	7.0000	7.00	768
Relative_Compactness	0.7641667	0.1057775	0.1475	0.62	0.6825	0.75	0.8300	0.98	768
Roof_Area	176.6041667	45.1659502	79.6250	110.25	140.8750	183.75	220.5000	220.50	768
Surface_Area	671.7083333	88.0861161	134.7500	514.50	606.3750	673.75	741.1250	808.50	768
Wall_Area	318.5000000	43.6264814	49.0000	245.00	294.0000	318.50	343.0000	416.50	768

2b)- Correlation



Strongest Negative Relationship

- Surface Area & Relative Compactness

Strongest Positive Relationship

- Cooling Load & Overall Height

2c)- Linear Regression for Cooling Load

```
linearmodel<- lm(Cooling_Load ~ .,data = EnergyEfficiency_df)
```

```
Coefficients: (1 not defined because of singularities)
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)             97.245749  20.764711   4.683 3.34e-06 ***
Relative_Compactness   -70.787707  11.225269  -6.306 4.85e-10 ***
Surface_Area            -0.088245   0.018628  -4.737 2.59e-06 ***
Wall_Area                0.044682   0.007253   6.161 1.17e-09 ***
Roof_Area                      NA         NA      NA       NA    
Overall_Height           4.283843   0.368730  11.618  < 2e-16 ***
Orientation              0.121510   0.103318   1.176    0.240    
Glazing_Area            14.717068   0.888018  16.573  < 2e-16 ***
Glazing_Area_Distribution 0.040697   0.076277   0.534    0.594    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.201 on 760 degrees of freedom
Multiple R-squared:  0.8878,	Adjusted R-squared:  0.8868 
F-statistic: 859.1 on 7 and 760 DF,  p-value: < 2.2e-16
```

Roof Area shows NA , which means completely multicollinearity

2d)- VIF analysis

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)    
(Intercept)              97.245749  20.764711   4.683 3.34e-06 ***
Relative_Compactness    -70.787707  11.225269  -6.306 4.85e-10 ***
Surface_Area             -0.088245   0.018628  -4.737 2.59e-06 ***
Wall_Area                 0.044682   0.007253   6.161 1.17e-09 ***
Overall_Height            4.283843   0.368730  11.618  < 2e-16 ***
Orientation               0.121510   0.103318   1.176    0.240    
Glazing_Area             14.717068   0.888018  16.573  < 2e-16 ***
Glazing_Area_Distribution  0.040697   0.076277   0.534    0.594    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.201 on 760 degrees of freedom
Multiple R-squared:  0.8878,	Adjusted R-squared:  0.8868 
F-statistic: 859.1 on 7 and 760 DF,  p-value: < 2.2e-16
```

```
     Relative_Compactness              Surface_Area                 Wall_Area 
               105.524054                201.531134                  7.492984 
           Overall_Height               Orientation              Glazing_Area 
                31.205474                  1.000000                  1.047508 
Glazing_Area_Distribution 
                 1.047508 
```

Relative Compactness, Surface Area ad Overall Height > 10, so there are multicollinearity between these three variables

3a)- Assign categories for Cooling Load

(A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile)

```
EnergyEfficiency_df$Cooling_category <-
  ifelse(EnergyEfficiency_df$Cooling_Load > 33.1325, "A",
  ifelse(EnergyEfficiency_df$Cooling_Load > 22.08, "B",
  ifelse(EnergyEfficiency_df$Cooling_Load > 15.62, "C", "D")))
```

Worst Efficiency

Cooling_category <chr>	avg_loading <dbl>
A	38.00328
B	28.56725
C	17.76775
D	12.61354

Cooling_Load	Cooling_category
26.89	B
26.46	B
22.93	B
23.84	B
24.17	B
23.87	B
35.78	A
35.48	A
36.97	A
36.70	A
32.52	B
33.28	A
32.33	B
33.24	A
10.39	D
10.34	D

Convert “Orientation” and “Glazing Area Distribution” to dummy variables

```
df_EE <- fastDummies::dummy_cols(
  df_EE,
  select_columns = c("Orientation", "Glazing_Area_Distribution"),
  remove_first_dummy = TRUE,
  remove_selected_columns = TRUE
)
```

```
tibble [768 × 16] (S3: tbl_df/tbl/data.frame)
 $ Relative_Compactness            : num [1:768] 0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9
0.86 0.86 ...
 $ Surface_Area                    : num [1:768] 514 514 514 514 564 ...
 $ Wall_Area                       : num [1:768] 294 294 294 294 318 ...
 $ Roof_Area                       : num [1:768] 110 110 110 110 122 ...
 $ Overall_Height                  : num [1:768] 7 7 7 7 7 7 7 7 7 7 ...
 $ Glazing_Area                    : num [1:768] 0 0 0 0 0 0 0 0 0 0 ...
 $ Cooling_Load                    : num [1:768] 21.3 21.3 21.3 21.3 28.3 ...
 $ Cooling_cat                     : chr [1:768] "C" "C" "C" "C" ...
 $ Orientation_North               : int [1:768] 1 0 0 0 1 0 0 0 1 0 ...
 $ Orientation_South               : int [1:768] 0 0 1 0 0 0 1 0 0 0 ...
 $ Orientation_West                : int [1:768] 0 0 0 1 0 0 0 1 0 0 ...
 $ Glazing_Area_Distribution_North  : int [1:768] 0 0 0 0 0 0 0 0 0 0 ...
 $ Glazing_Area_Distribution_South  : int [1:768] 0 0 0 0 0 0 0 0 0 0 ...
 $ Glazing_Area_Distribution_Uniform: int [1:768] 0 0 0 0 0 0 0 0 0 0 ...
 $ Glazing_Area_Distribution_Unknown: int [1:768] 1 1 1 1 1 1 1 1 1 1 ...
 $ Glazing_Area_Distribution_West   : int [1:768] 0 0 0 0 0 0 0 0 0 0 ...
```

Dummy for orientation

Dummy for glazing area distribution

3b)- Create 5 perceptrons for Cooling Load

(assign 1 for categories A &B, -1 for categories C&D)

Assign categories

```r
df_EE_for_perceptrons$Cooling_cat <-ifelse(df_EE_for_perceptrons$Cooling_cat %in%
c("A","B"), 1 , -1)
```

Delete cooling load column

```r
df_EE_for_perceptrons <- df_EE_for_perceptrons %>% select(-Cooling_Load)
```

Create the training and testing data

```r
set.seed(42)
ml_index <- sample(nrow(df_EE),
                   0.7 * nrow(df_EE),
                   replace = FALSE)
```

```r
ptrons_train<- df_EE_for_perceptrons[ml_index,]
ptrons_test <- df_EE_for_perceptrons[-ml_index,]
```

```r
X <- ptrons_train%>% select(-Cooling_cat)
y <- ptrons_train$Cooling_cat
```

```r
X_test <- ptrons_test%>% select(-Cooling_cat)
y_test <- ptrons_test$Cooling_cat
```

```r
perceptron <- function(X, y, numEpochs) {
  set.seed(42)
results <- list()
w <- runif(ncol(X), -10, 10) #Initalize weights
# For loop - number of generations(epochs) - number of times dataset is ran through
for(j in 1:numEpochs) {
predictedResult <- numeric(length=100) # Initalize predictedResult vector
numIncorrect = 0 # Keeps track of # of missclassified points
# For loop - loop throught dataset
for(i in 1:length(y)) {
xi = as.numeric(unlist(X[i,])) # Convert dataframe to vector
predictedResult[i] = sign(w %*% xi) # Predict the point
# If predicted point is incorrect - change weight
if(predictedResult[i] != y[i]) {
numIncorrect = numIncorrect + 1 # Add one to # of missclassified points
w <- w + as.numeric(y[i]) * xi # Update the weight w <- w + WiXi
}
}
# Print results of this generation(epoch)
cat("\nEpoch #: ", j)
cat("\nNumber Incorrect: ", numIncorrect)
cat("\nFinal Weight: ", w)
}
}
```

3b)- Output and Accuracy

```
Epoch #:  1
Number Incorrect:  100
Final Weight:  21.64612 -946.7585 3033.723 -1990.141 188.3349 2.781919 5.731766
-0.3066681 5.139846 5.101296 1.154836 9.382245 4.693445 -3.891424
Epoch #:  2
Number Incorrect:  42
Final Weight:  24.88612 -1191.758 3793.223 -2492.391 237.3349 5.231919 9.731766 -1.306668
6.139846 7.101296 1.154836 12.38225 -1.306555 -3.891424
Epoch #:  3
Number Incorrect:  22
Final Weight:  24.91612 -1240.758 3915.723 -2578.141 240.8349 6.481919 8.731766 0.6933319
6.139846 9.101296 0.1548355 15.38225 -6.306555 -3.891424
Epoch #:  4
Number Incorrect:  29
Final Weight:  26.49612 -1314.258 4209.723 -2761.891 261.8349 8.081919 7.731766 1.693332
8.139846 12.1013 -2.845164 18.38225 -12.30656 -1.891424
Epoch #:  5
Number Incorrect:  36
Final Weight:  29.64612 -1461.258 4626.223 -3043.641 300.3349 10.08192 8.731766 3.693332
9.139846 15.1013 -5.845164 22.38225 -18.30656 -0.8914235NULL
```

```r
weight_list <- list(w1, w2, w3, w4, w5)

for (i in 1:length(weight_list)){
  w <- weight_list[[i]]
  score <- as.matrix(X_test) %*% w
  ptron_prediction <- ifelse(score > 0, 1, -1)
  cat("========This is w", i, "Accuracy======= \n")
  print(confusionMatrix(as.factor(ptron_prediction), as.factor(y_test)))
}
```

```
========This is w 1 Accuracy=======
Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 105   2
        1    2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

```
========This is w 2 Accuracy=======
Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 105   2
        1    2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

```
========This is w 3 Accuracy=======
Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 105   2
        1    2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

```
========This is w 4 Accuracy=======
Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 105   2
        1    2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

```
========This is w 5 Accuracy=======
Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 105   2
        1    2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

3c)- Create a SVM for Cooling Load(category)

```
df_EE_SVM <- df_EE %>% select(-Cooling_Load)
```

Convert category to factor

```
df_EE_SVM$Cooling_cat <- as.factor(df_EE_SVM$Cooling_cat)
```

Create the training and testing data

```
SVM_train <- df_EE_SVM[ml_index,]
SVM_test <- df_EE_SVM[-ml_index,]
```

Run SVM

```
EE_svm <- svm(Cooling_cat ~.,
              data = SVM_train)
```

Make the prediction

```
EE_predict <- predict(EE_svm,
                      SVM_test[,-7],
                      type = "response")
```

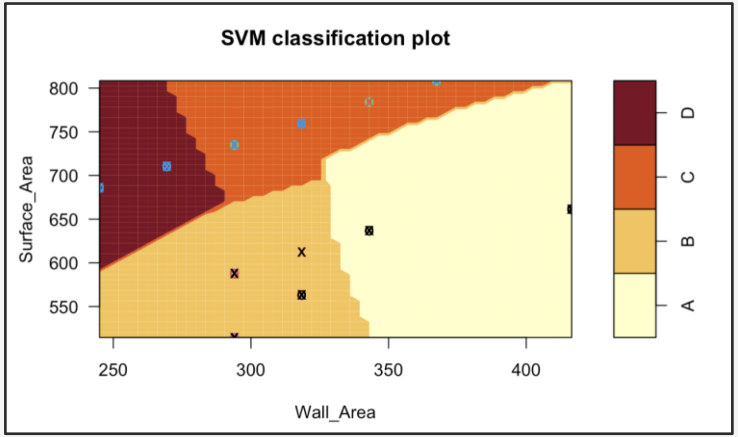
```
Confusion Matrix and Statistics

          Reference
Prediction  A  B  C  D
         A 48  7  0  0
         B 16 50  1  0
         C  0  3 46 13
         D  0  0 10 37

Overall Statistics

               Accuracy : 0.7835
                 95% CI : (0.7248, 0.8349)
    No Information Rate : 0.2771
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7108

 Mcnemar's Test P-Value : NA
```

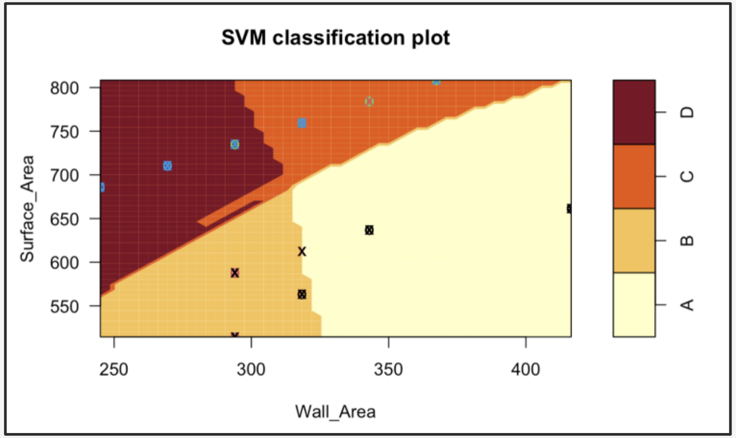
Accuracy : 0.7835

3c)- Graph for SVM

```r
plot(EE_svm,
     SVM_train,
     Surface_Area ~ Wall_Area, slice = list(Relative_Compactness = 0.7,
                                            Roof_Area = 110.25,
                                            Overall_Height = 3.5,
                                            Glazing_Area = 0.25,
                                            Orientation_North = 0,
                                            Orientation_South = 1,
                                            Orientation_West = 0,
                                            Glazing_Area_Distribution_North = 0,
                                            Glazing_Area_Distribution_South = 1,
                                            Glazing_Area_Distribution_Uniform = 0,
                                            Glazing_Area_Distribution_Unknown =0,
                                            Glazing_Area_Distribution_West =0))
```

```r
plot(EE_svm,
     SVM_train,
     Surface_Area ~ Wall_Area, slice = list(Relative_Compactness = 0.7,
                                            Roof_Area = 110.25,
                                            Overall_Height = 3.5,
                                            Glazing_Area = 0.25,
                                            Orientation_North = 0,
                                            Orientation_South = 0,
                                            Orientation_West = 0,
                                            Glazing_Area_Distribution_North = 0,
                                            Glazing_Area_Distribution_South = 0,
                                            Glazing_Area_Distribution_Uniform = 0,
                                            Glazing_Area_Distribution_Unknown =0,
                                            Glazing_Area_Distribution_West =0))
```





3d)- Create Neural Network from 1 to 5 hidden nodes

(cooling load, continuous variable)

Create normalize and denormalize

```
normalize <- function(x) {return((x-min(x))/(max(x)-min(x)))}
denormalize <- function(y,x){return(y*(max(x)-min(x))+min(x))}
df_EE_neural_network <- df_EE %>% select(-Cooling_cat)
df_EE_neural_network <- as.data.frame(lapply(df_EE_neural_network,normalize))
```

Create the training and testing data

```
NN_train <- df_EE_neural_network[ml_index,]
NN_test <- df_EE_neural_network[-ml_index,]
```

Run Neural Network (using loop for 1 to 5)

```
EE_neural_list <- list()

for (layer in 1:5) {
  set.seed(42)
  EE_net <- neuralnet(Cooling_Load ~ .,
                      NN_train,
                      hidden = layer,
                      lifesign = "minimal",
                      linear.output = TRUE,
                      threshold = 0.01)

  EE_net.results <- compute(EE_net, NN_test)

  NN_denorm <- denormalize(EE_net.results$net.result, df_EE$Cooling_Load)
  actual_NN_denorm <- denormalize(NN_test$Cooling_Load, df_EE$Cooling_Load)

  predictive_cor <- cor(NN_denorm, actual_NN_denorm)
  EE_neural_list[[as.character(layer)]] <- predictive_cor
}


print(EE_neural_list)
```

```
hidden: 1    thresh: 0.01    rep: 1/1    steps:      971 error: 1.89333  time: 0.13 secs
hidden: 2    thresh: 0.01    rep: 1/1    steps:      795 error: 1.72856  time: 0.1 secs
hidden: 3    thresh: 0.01    rep: 1/1    steps:    12384 error: 0.70008  time: 2.04 secs
hidden: 4    thresh: 0.01    rep: 1/1    steps:    42717 error: 0.41825  time: 9.28 secs
hidden: 5    thresh: 0.01    rep: 1/1    steps:    22379 error: 0.57512  time: 5.54 secs
```

Correlation :

```
$`1`
          [,1]
[1,] 0.9488919

$`2`
          [,1]
[1,] 0.9498522

$`3`
          [,1]
[1,] 0.9773979

$`4`
          [,1]
[1,] 0.9854045

$`5`
          [,1]
[1,] 0.9812274
```

3e)- Create two K-nearest neighbor analysis for Cooling Load (category)

Remove the Cooling_Load and assign to a new dataset

Extract Cooling_cat as label and normalize the dataset

```
df_EE_knn_pre <- df_EE %>% select(-Cooling_Load)
EE_knn_labels <- df_EE_knn_pre %>% select(Cooling_cat)
EE_knn_labels
```

```
df_EE_knn <- as.data.frame(lapply(df_EE_knn_pre[,-7],normalize))
```

```
knn_train <- df_EE_knn[ml_index,]
knn_test <- df_EE_knn[-ml_index,]
```

```
knn_train_labels <- EE_knn_labels[ml_index,]
knn_test_labels <- EE_knn_labels[-ml_index,]
```

3e)- Output and Accuracy of KNN

```r
set.seed(42)
EE_knn <- knn(train = knn_train,
              test = knn_test,
              cl = knn_train_labels$Cooling_cat,
              k = 21)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  A  B  C  D
         A 45 14  0  0
         B 19 44  2  0
         C  0  2 45 12
         D  0  0 10 38

Overall Statistics

               Accuracy : 0.7446
                 95% CI : (0.6833, 0.7995)
    No Information Rate : 0.2771
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6586

 Mcnemar's Test P-Value : NA
```

```r
set.seed(42)
EE_knn <- knn(train = knn_train,
              test = knn_test,
              cl = knn_train_labels$Cooling_cat,
              k = 51)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  A  B  C  D
         A 49 13  0  0
         B 15 45  2  0
         C  0  2 41 11
         D  0  0 14 39

Overall Statistics

               Accuracy : 0.7532
                 95% CI : (0.6924, 0.8074)
    No Information Rate : 0.2771
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6704

 Mcnemar's Test P-Value : NA
```

3f)- Create Naïve Bayes analysis for Cooling Load (category)

Delete the cooling load column

```
df_EE_nb <- df_EE[,-7]
nb_train <- df_EE_nb[ml_index,]
nb_test <- df_EE_nb[-ml_index,]
```

Run Naïve Bayes Model

```
NB_model <- naiveBayes(Cooling_cat ~.,
                       data = nb_train,
                       laplace = 1)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        A         B         C         D 
0.2402235 0.2439479 0.2513966 0.2644320 
```

3f)- Output and Accuracy of Naïve Bayes

```
Conditional probabilities:
     Relative_Compactness
Y          [,1]        [,2]
  A 0.8253488 0.06993352
  B 0.8658779 0.07688023
  C 0.6741481 0.05751554
  D 0.6877465 0.04146937

     Surface_Area
Y         [,1]     [,2]
  A 614.7791 47.28979
  B 587.0649 55.23730
  C 750.7889 51.71025
  D 736.2077 42.41793

     Wall_Area
Y         [,1]     [,2]
  A 350.5969 46.01894
  B 309.8969 21.68549
  C 316.3222 36.34759
  D 295.2077 42.41793

     Roof_Area
Y         [,1]     [,2]
  A 132.0911 13.52027
  B 138.5840 22.59217
  C 217.2333 17.07671
  D 220.5000  0.00000
```

```
   Overall_Height
Y      [,1]      [,2]
  A 7.000000 0.0000000
  B 6.866412 0.6731797
  C 3.629630 0.6634458
  D 3.500000 0.0000000

   Glazing_Area
Y        [,1]       [,2]
  A 0.2786822 0.12496487
  B 0.2053435 0.13246943
  C 0.3037037 0.12438350
  D 0.1461268 0.09603375

   Orientation_North
Y        [,1]      [,2]
  A 0.2403101 0.4289375
  B 0.2595420 0.4400662
  C 0.2592593 0.4398603
  D 0.2816901 0.4514154

   Orientation_South
Y        [,1]      [,2]
  A 0.2170543 0.4138470
  B 0.2748092 0.4481318
  C 0.2444444 0.4313579
  D 0.2323944 0.4238542

   Orientation_West
Y        [,1]      [,2]
  A 0.2790698 0.4502906
  B 0.2290076 0.4218072
  C 0.2296296 0.4221611
  D 0.2042254 0.4045614
```

```
   Glazing_Area_Distribution_North
Y      [,1]      [,2]
  A 0.2015504 0.4027221
  B 0.2061069 0.4060610
  C 0.2074074 0.4069599
  D 0.1760563 0.3822163

   Glazing_Area_Distribution_South
Y      [,1]      [,2]
  A 0.1627907 0.3706139
  B 0.1908397 0.3944715
  C 0.1777778 0.3837495
  D 0.1478873 0.3562449

   Glazing_Area_Distribution_Uniform
Y      [,1]      [,2]
  A 0.1782946 0.3842528
  B 0.1984733 0.4003815
  C 0.2148148 0.4122234
  D 0.1408451 0.3490930

   Glazing_Area_Distribution_Unknown
Y       [,1]       [,2]
  A 0.01550388 0.1240272
  B 0.06870229 0.2539182
  C 0.05185185 0.2225537
  D 0.09859155 0.2991681

   Glazing_Area_Distribution_West
Y      [,1]      [,2]
  A 0.2093023 0.4083966
  B 0.1679389 0.3752470
  C 0.1925926 0.3958044
  D 0.2112676 0.4096528
```

```
Confusion Matrix and Statistics

          Reference
Prediction  A  B  C  D
         A 64 52  0  0
         B  0  5  2  0
         C  0  0  0  0
         D  0  3 55 50

Overall Statistics

               Accuracy : 0.5152
                 95% CI : (0.4487, 0.5812)
    No Information Rate : 0.2771
    P-Value [Acc > NIR] : 1.937e-14

                  Kappa : 0.3551

 Mcnemar's Test P-Value : NA
```

Accuracy : 0.5152

3g)- Create Decision Tree for Cooling Load

(assign 1 for categories A &B, 0 for categories C&D)

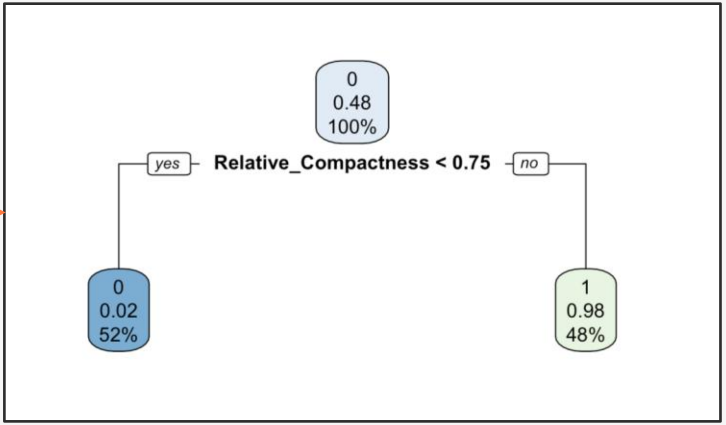
Assign categories

```r
dtree_train$Cooling_bi <- ifelse(dtree_train$Cooling_cat %in% c("A", "B"), 1, 0)
dtree_test$Cooling_bi <- ifelse(dtree_test$Cooling_cat %in% c("A", "B"), 1, 0)
```

```r
dtree_train <- dtree_train %>% select(-Cooling_cat)
dtree_test <- dtree_test %>% select(-Cooling_cat)
```

Run Decision Tree

```r
EE_tree <- rpart(Cooling_bi~ .,
                 data = dtree_train,
                 method = 'class')
rpart.plot(EE_tree)
```




```r
EE_dtree_predict <- predict(EE_tree, dtree_test[,-15], type = 'class')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 105   2
         1   2 122

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 1
```

Accuracy : 0.9827

3g)- Create Random Forest for Cooling Load

(assign 1 for categories A &B, 0 for categories C&D)

```
rforest_train$Cooling_bi <- as.factor(rforest_train$Cooling_bi)
```

Run Random Forest

```
set.seed(42)
EE_forest <- randomForest(Cooling_bi~ .,
                          data = rforest_train,
                          ntree=500,
                          proximity=TRUE,
                          importance=TRUE)
EE_forest
```

```
Call:
 randomForest(formula = Cooling_bi ~ ., data = rforest_train,      ntree = 500, proximity = 
TRUE, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.86%
Confusion matrix:
    0   1 class.error
0 273   4  0.01444043
1   6 254  0.02307692
```

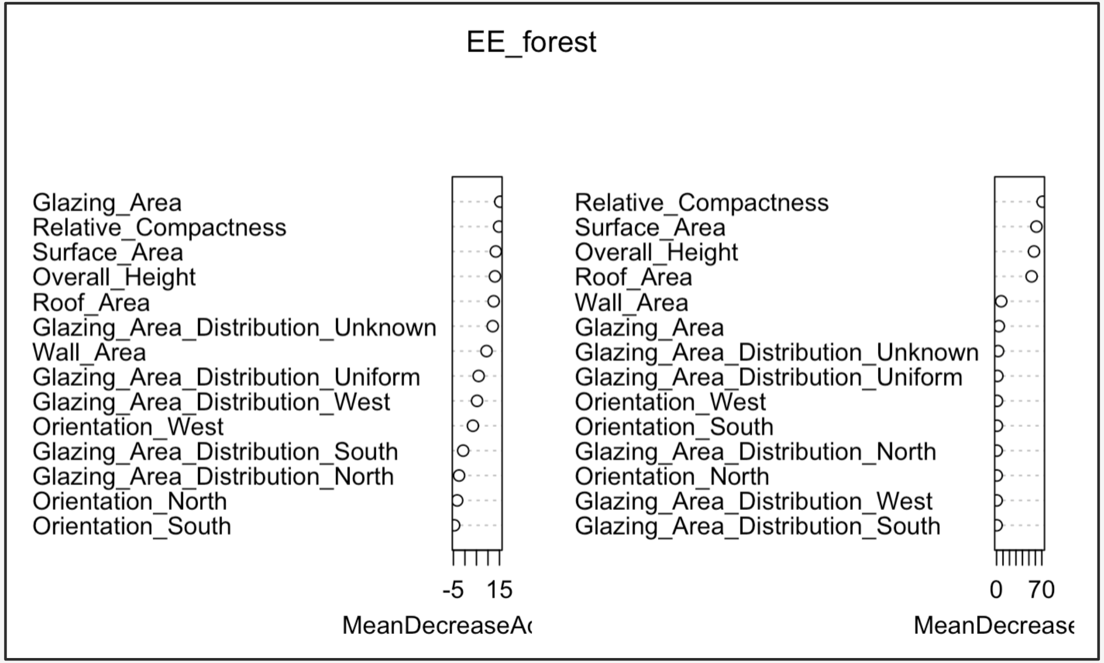
Accuracy : 1-1.86% = 98.14%

```
importance(EE_forest)
```

```
                                         0          1 MeanDecreaseAccuracy
Relative_Compactness            14.8068939 14.7042983           14.8110016
Surface_Area                    13.3564226 13.3153472           13.3892546
Wall_Area                        8.3300217  8.8846852            9.3668720
Roof_Area                       12.2696719 12.5283233           12.4677729
Overall_Height                  12.8959803 13.0017451           13.0152429
Glazing_Area                    11.7614052 12.4286280           15.3172749
Orientation_North               -1.0924686 -3.9506109           -3.3553858
Orientation_South               -4.5292794 -2.0534055           -4.6637967
Orientation_West                 0.8182706  3.9322485            3.4606739
Glazing_Area_Distribution_North -1.5178616 -2.6809275           -2.5294804
Glazing_Area_Distribution_South -1.1180599  0.1176387           -0.7657586
Glazing_Area_Distribution_Uniform 0.4662219  8.1619197           5.9627584
Glazing_Area_Distribution_Unknown 12.6844669  8.2490095          12.0610822
Glazing_Area_Distribution_West   5.0894779  2.5205898            5.2652767
```

3g)- Create Random Forest for Cooling Load

(assign 1 for categories A &B, 0 for categories C&D)




```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 106   3
         1   1 121

               Accuracy : 0.9827
                 95% CI : (0.9563, 0.9953)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9652

 Mcnemar's Test P-Value : 0.6171
```

Accuracy : 0.9827

3i)- Create Boosting model for Cooling Load

```
xgboost_train <- rforest_train
xgboost_test <- rforest_test
```

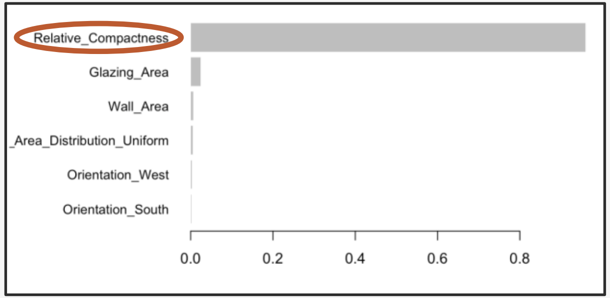
```
x_train <- as.matrix(xgboost_train[,-15])
y_train <- as.numeric(xgboost_train$Cooling_bi)-1
x_test <- as.matrix(xgboost_test[,-15])
y_test <- xgboost_test$Cooling_bi
```

```
dtrain <- xgb.DMatrix(data = x_train,
                      label = y_train)
dtrain
```

```
EE_xgb <- xgboost(data = dtrain,
                  max.depth = 5,
                  eta = 1,
                  nthread = 2,
                  nrounds = 1000,
                  objective = "binary:logistic",
                  verbose = 0)
```

Run Boosting Model

```
xgb.plot.importance(xgb.importance(model = EE_xgb),
                    measure = "Gain")
```



Relative Compactness is the most important variable

```
EE_xgboost_predict <- predict(EE_xgb, x_test)
head(EE_xgboost_predict)
```

```
[1] 0.1295246 0.5870160 0.8806500 0.8806500 0.2023590 0.1362648
```

```
EE_xgboost_predict <- as.numeric(EE_xgboost_predict > 0.5)
head(EE_xgboost_predict)
```

```
[1] 0 1 1 1 0 0
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 106   4
         1   1 120

               Accuracy : 0.9784
                 95% CI : (0.9502, 0.9929)
    No Information Rate : 0.5368
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9566

 Mcnemar's Test P-Value : 0.3711
```

Accuracy : 0.9784

3j)- Summarize the accuracy of all models

Technique	Target Variable	Accuracy	P-value
Perceptron 1-5	Cooling_cat (A/B: 1 vs C/D : -1)	0.9827	2e-16
SVM	Cooling_cat	0.7835	2.2e-16
KNN(K=21)	Cooling_cat	0.7446	2.2e-16
KNN(K=51)	Cooling_cat	0.7532	2.2e-16
Naïve Bayes	Cooling_cat	0.5152	1.937e-14
Decision Tree	Cooling_bi (A/B :1 vs C/D :0)	0.9827	2e-16
Random Forest	Cooling_bi	0.9827	2e-16
Boosting	Cooling_bi	0.9784	2e-16

3j)- Summarize the correlation of all models

Technique	Target Variable	Correlation
Neural Network Node = 1	Cooling_Load	0.9489
Neural Network Nodes =2	Cooling_Load	0.9499
Neural Network Nodes = 3	Cooling_Load	0.9774
Neural Network Nodes = 4	Cooling_Load	0.9854
Neural Network Nodes = 5	Cooling_Load	0.9812

4)- List of Lessons Learned

a)- What technique worked best?

Technique	Accuracy	P-value
Perceptron	0.9827	2e-16
Decision Tree	0.9827	2e-16
Random Forest	0.9827	2e-16

Technique	Target Variable	Correlation
Neural Network Nodes = 4	Cooling_Load	0.9854

4)- List of Lessons Learned

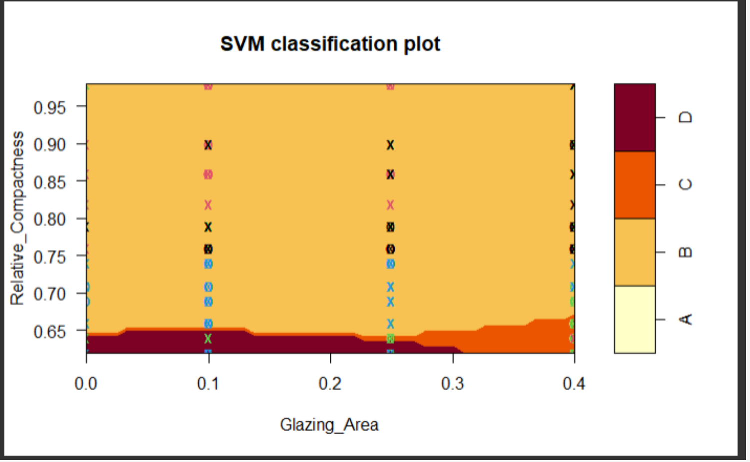
b)- What insights do you have for builders of energy efficient buildings?

- According to **linear regression analysis**, we found **Relative Compactness, Surface Area, Wall Area, Overall Height, and Glazing Area** are all significant, which means these variables will impact cooling load.
- In addition, according to **Decision Tree** and **XGboosting** analysis, it indicates **Relative Compactness** is the most important variables.
- Furthermore, based on the **Random Forest** variable importance analysis, we observe that **Glazing Area** emerges as the most critical feature, slightly ahead of **Relative Compactness**. Other variables such as **Surface Area**, **Overall Height**, and **Roof Area** also demonstrate importance scores, reinforcing the insights obtained from linear regression analysis except for “Roof Area”
- Even though Roof_Area was eliminated in the linear regression model due to perfect multicollinearity, it still contributes to the Random Forest model I believe it’s because Random Forest is not based on linear assumptions.
- Additionally, by applying **SVM slice** visualization, we can fix non-target variables and strategically adjust key features to explore how specific changes can lead to improved energy efficiency outcomes.

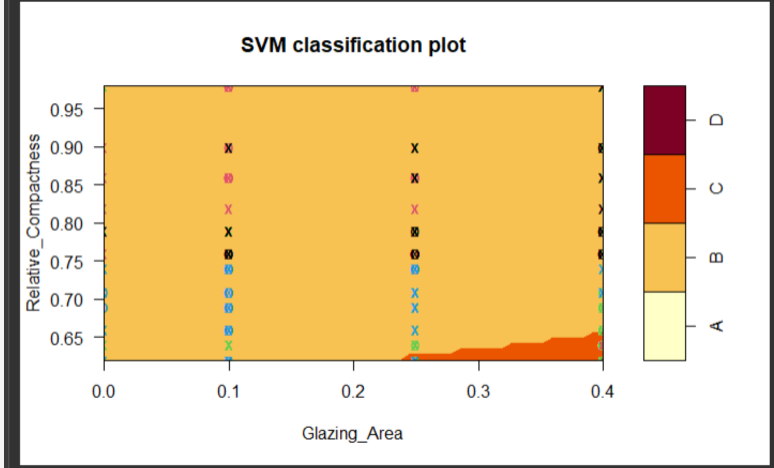
4)- List of Lessons Learned

b)- What insights do you have for builders of energy efficient buildings?

```{r}
plot(EE_svm,
     SVM_train,
     Relative_Compactness ~ Glazing_Area, slice = list(Wall_Area = 245,
                                                        Surface_Area = 514.50,
                                                        Roof_Area = 110.25,
                                                        Overall_Height = 3.5,
                                                        Orientation_North = 0,
                                                        Orientation_South = 0,
                                                        Orientation_West = 0,
                                                        Glazing_Area_Distribution_North = 0,
                                                        Glazing_Area_Distribution_South = 0,
                                                        Glazing_Area_Distribution_Uniform = 0,
                                                        Glazing_Area_Distribution_Unknown =0,
                                                        Glazing_Area_Distribution_West =0))
```




```{r}
plot(EE_svm,
     SVM_train,
     Relative_Compactness ~ Glazing_Area, slice = list(Wall_Area = 245,
                                                        Surface_Area = 514.50,
                                                        Roof_Area = 110.25,
                                                        Overall_Height = 3.5,
                                                        Orientation_North = 0,
                                                        Orientation_South = 1,
                                                        Orientation_West = 0,
                                                        Glazing_Area_Distribution_North = 0,
                                                        Glazing_Area_Distribution_South = 1,
                                                        Glazing_Area_Distribution_Uniform = 0,
                                                        Glazing_Area_Distribution_Unknown =0,
                                                        Glazing_Area_Distribution_West =0))
```



Thanks!

Group 6 :
ChihHao Luca Yuan
Ching Yu Hsu