BUA 751 - Machine Learning for Business

Energy Efficiency

Content

- **Business Problem**
- **Visualization** -
  - Scatterplot
  - Histograms
- **Statistical Analysis** -
  - Descriptive Statistics
  - Correlation Matrix
  - Linear Regression
  - VIF (Variance Inflation Factor)
- **Modeling Analysis** -
  - Perceptron
  - SVM (Support Vector Machine)
  - Neural Networks
  - k-Nearest Neighbor
  - Naive Bayes
  - Decision Trees
  - Random Forest

Business Problem

Alternative building designs area evaluated for energy efficiency – both heating and cooling. We perform energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. We simulate various settings as functions of the afore-mentioned characteristics to obtain 768 building shapes. The dataset comprises 768 samples and 8 features, aiming to predict two real valued responses. It can also be used as a multi-class classification problem if the response is rounded to the nearest integer.

The aim is to use the eight features to predict each of the two responses.

Source: *https://archive.ics.uci.edu/ml/datasets/Energy+efficiency*

Dataset Overview

Dataset contains **768** rows that represents **building shapes** and **10** columns -

X1 - Relative Compactness - it is the measure of compactness of the building

X2 - Surface Area - surface area of the building (**m²)**

X3 - Wall Area - area of the building covered by the width of the wall **(m²)**

X4 - Roof Area - area covered under roofs **(m²)**

X5 - Overall Height - total height of the building **(m)**

X6 - Orientation - orientation of the building based on direction **(2=North, 3=East,4=South,5=West)**

X7 - Glazing Area - total area of the wall that is glass **(0%,10%,25%,40% of the floor area)**

X8 - Glazing Area Distribution - how glazing area is distributed within the building **(1=Uniform,2=North, 3=East,4=South,5=West)**

y1 - Heating Load - how much load is required to heat the building **(kWh/m²)**

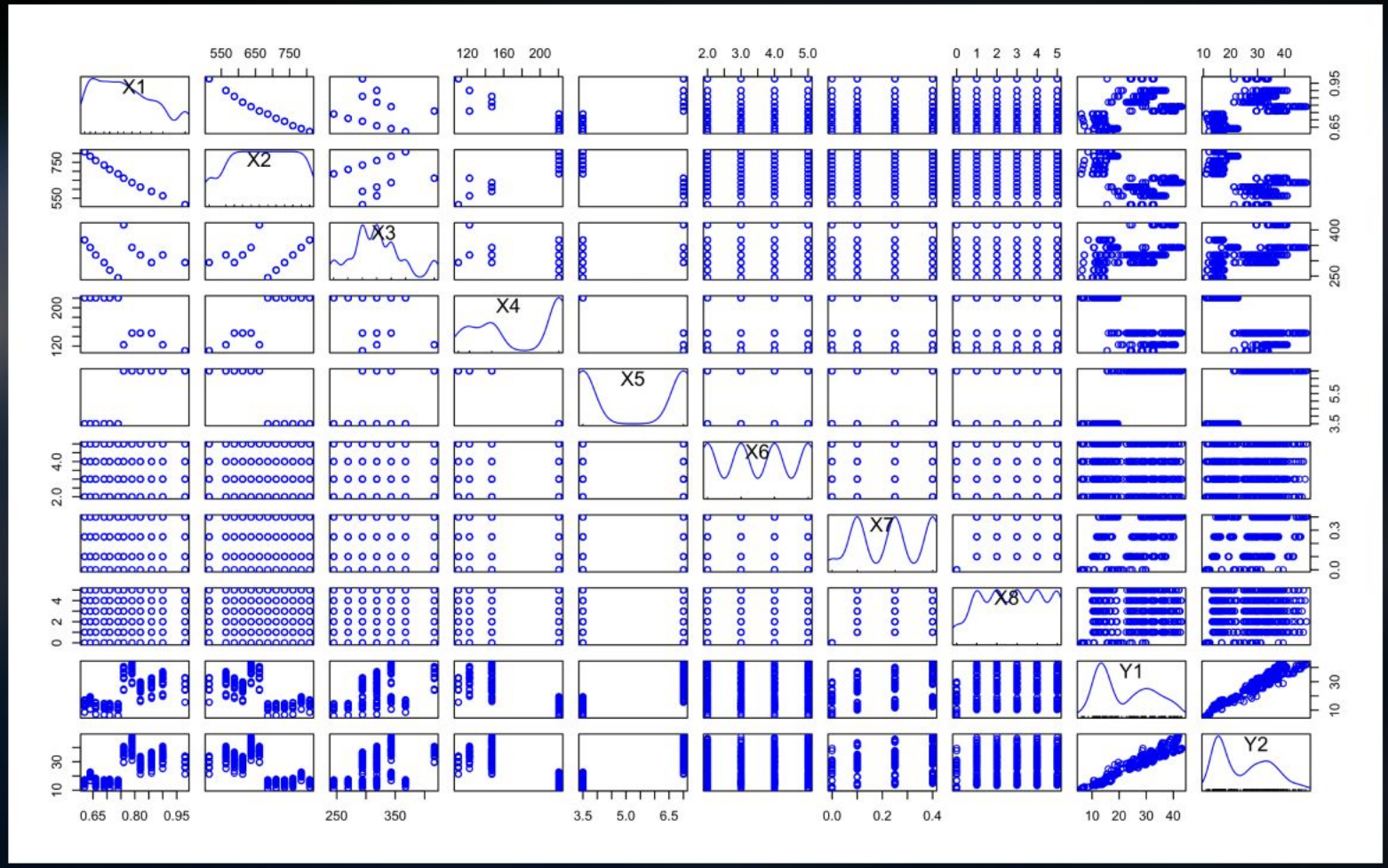
y2 - Cooling Load - how much load is required to cool the building **(kWh/m²)**

Visualizations

Overall view of relationship between dependent variables (y1 and y2) with all other continuous variables.

Steps -

1. Plotting ‘Scatterplot Matrix’ using Rcmdr in R.
2. Analyzing plots and infering trends between individual variables and dependent variables.
3. Good linear trends between X1 and X2, y1 and y2.



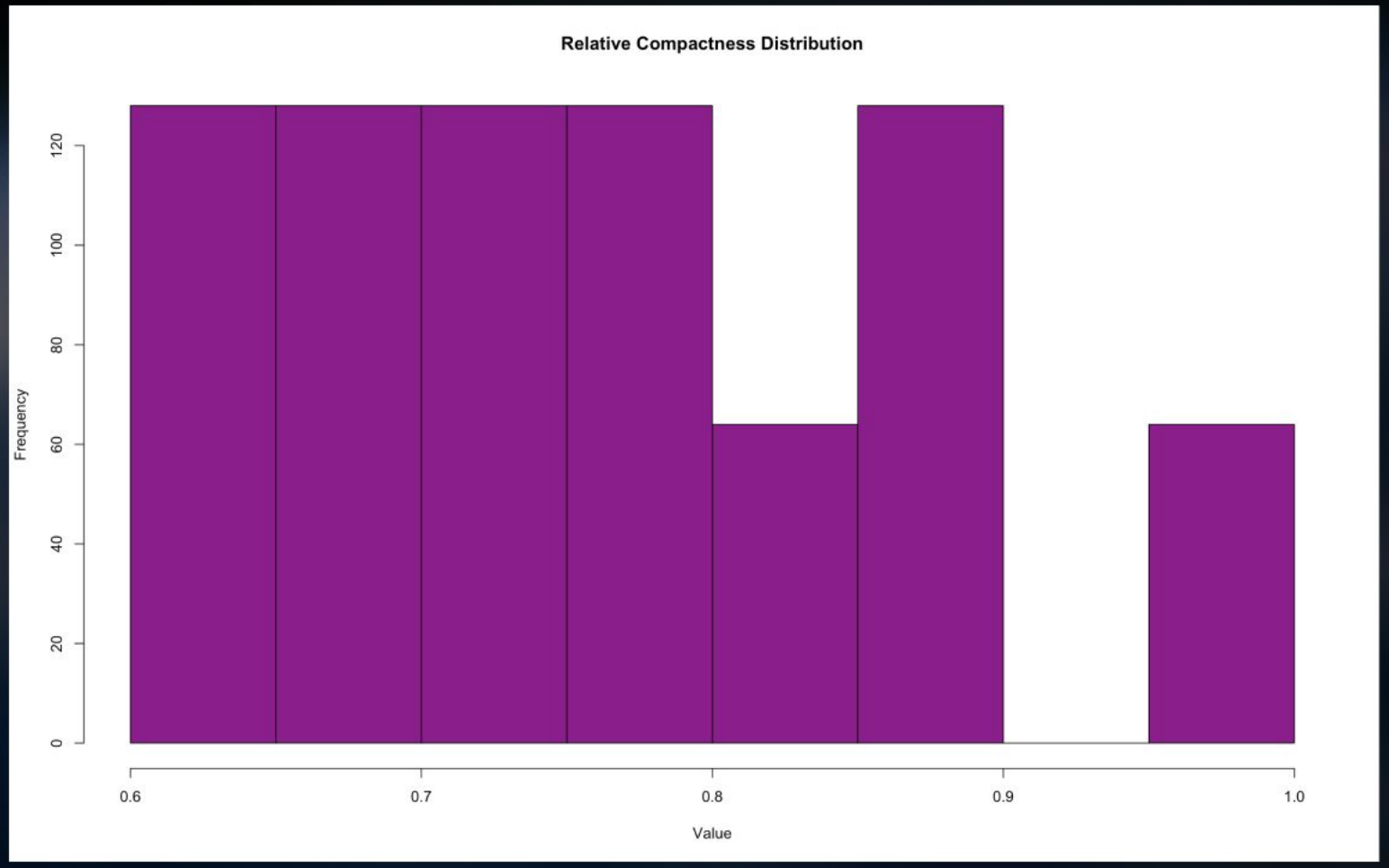
Histograms

Histogram of
Relative Compactness

Steps -

1. Using **hist()** function -

```
hist(main_energy$X1,
     main="Relative Compactness Distribution",
     xlab="Value",
     xlim=c(0.6,1),
     col="darkmagenta"
)
```



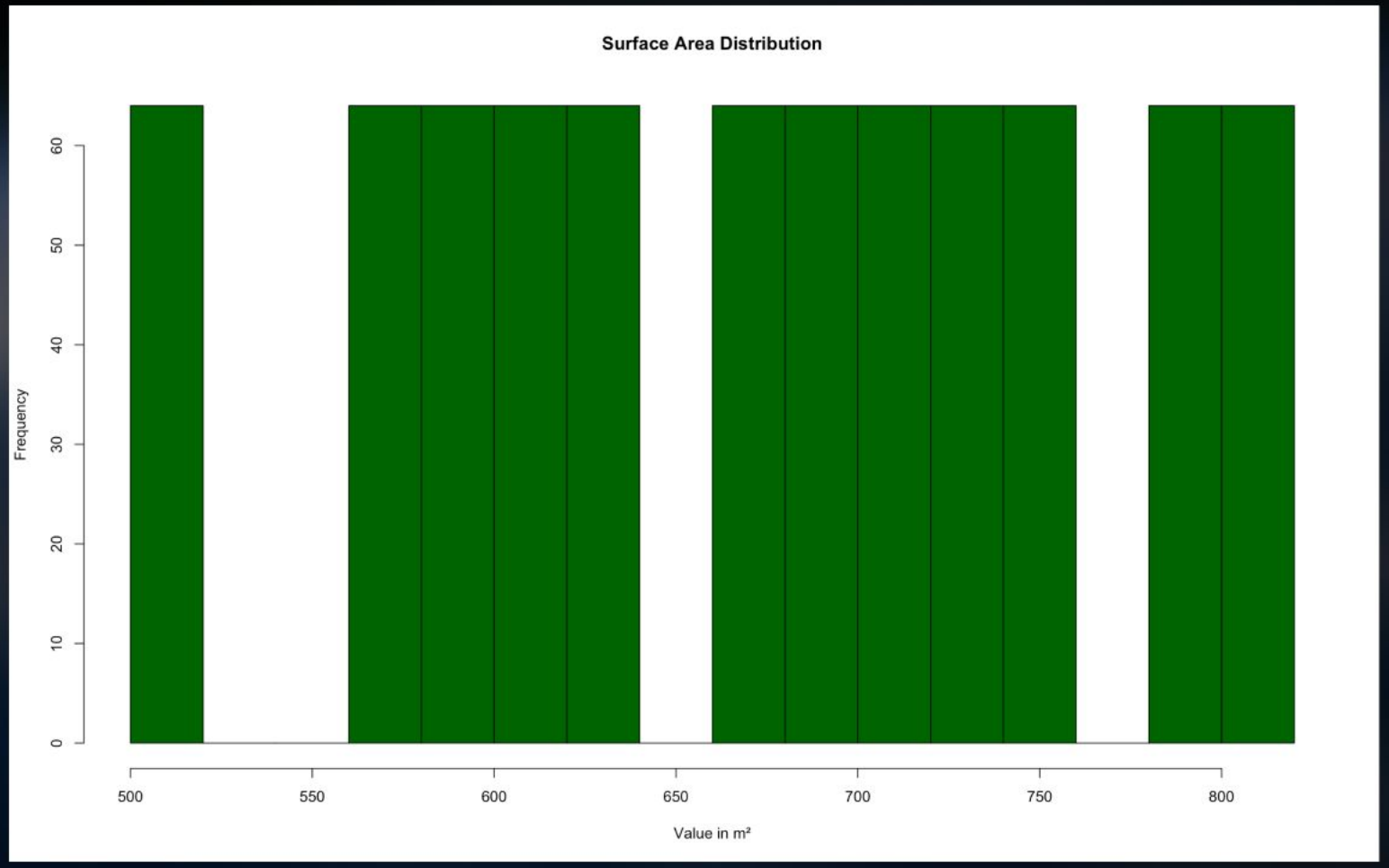
Histograms

Histogram of
Surface Area

Steps -

1. Using **hist()** function -

```
hist(main_energy$X2,
     main="Surface Area Distribution",
     xlab="Value in m²",
     col="darkgreen"
)
```



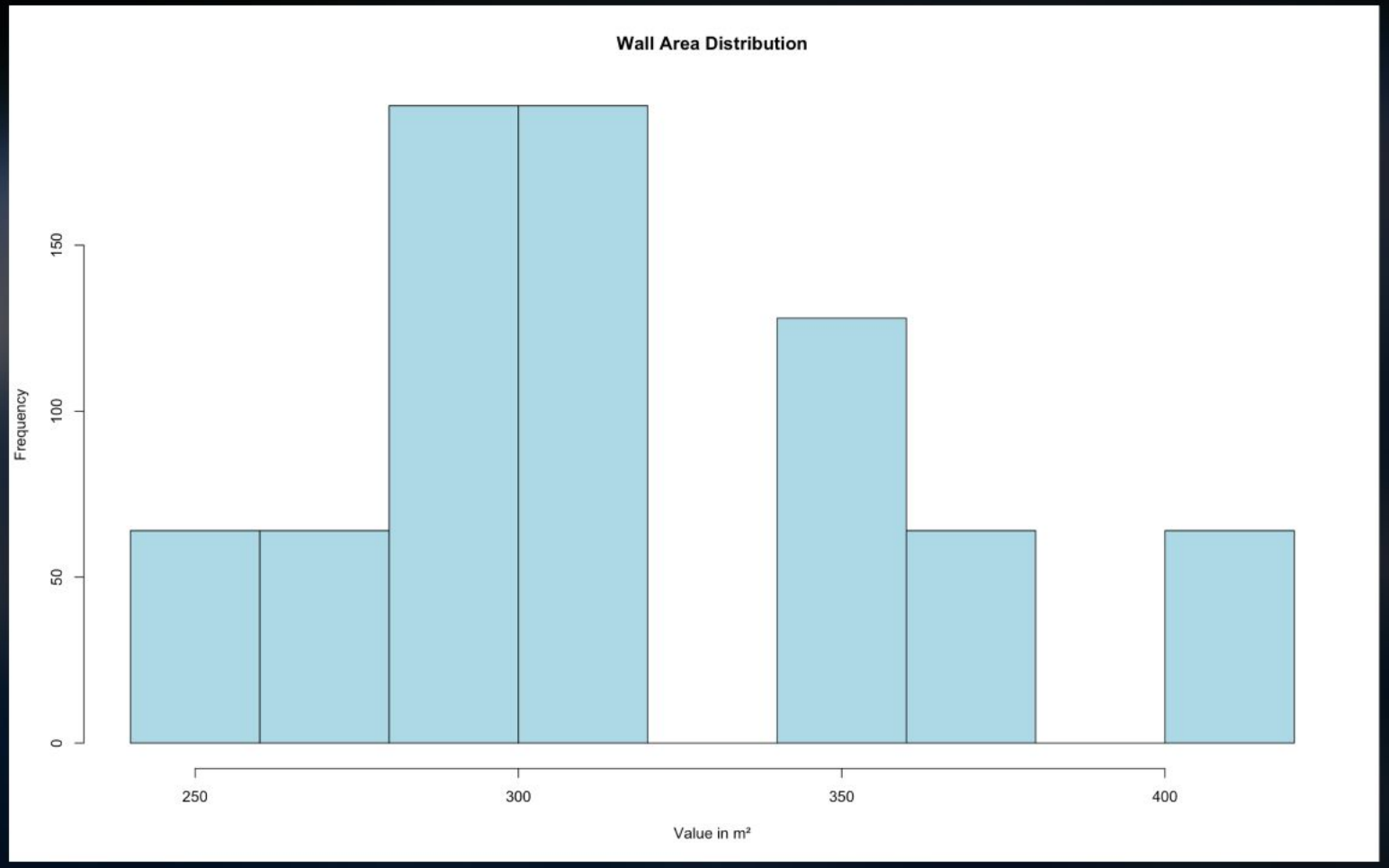
Histograms

Histogram of
Wall Area

Steps -

1. Using **hist()** function -

```
hist(main_energy$X3,
     main="Wall Area Distribution",
     xlab="Value in m²",
     col="lightblue"
)
```



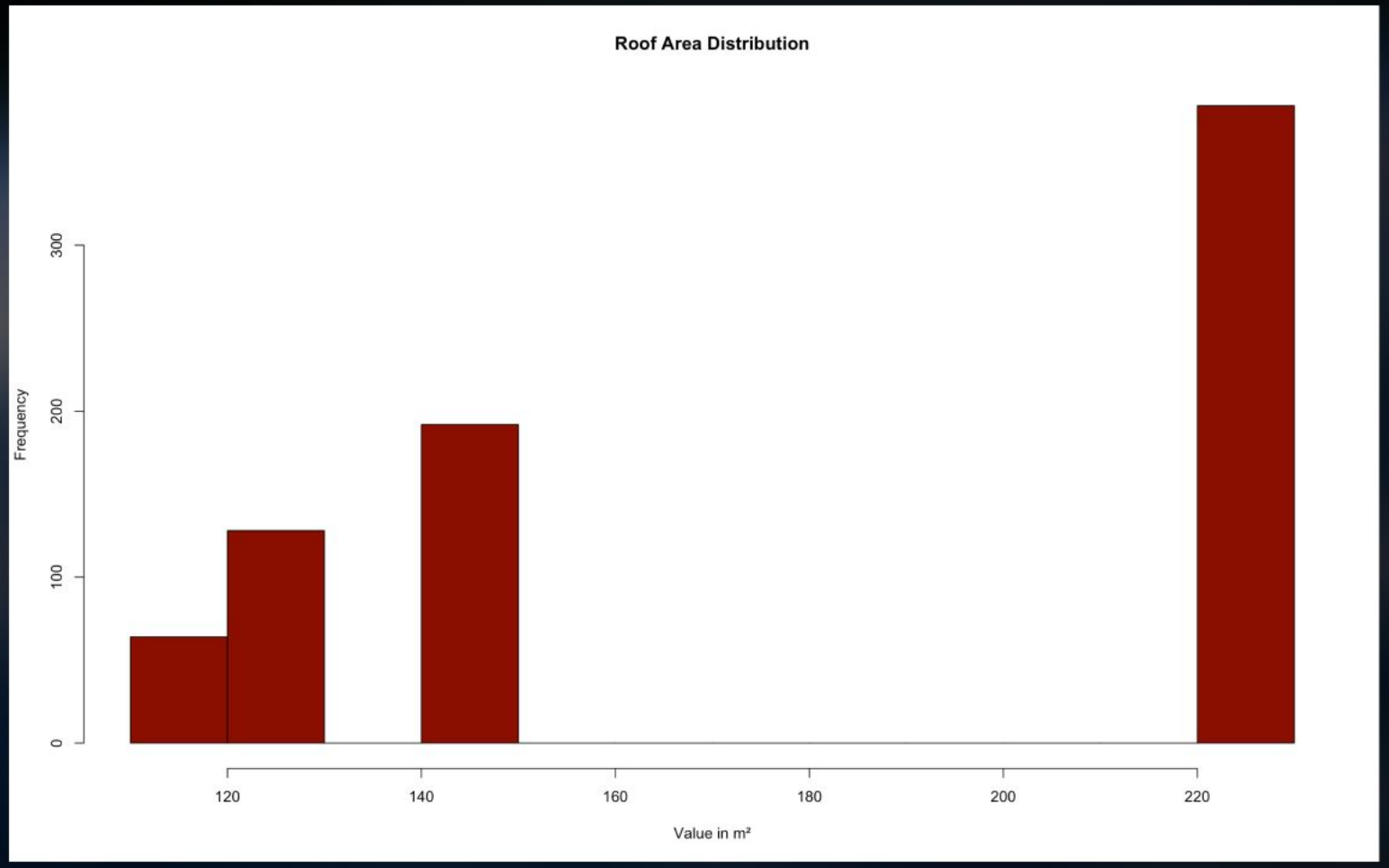
Histograms

Histogram of
Roof Area

Steps -

1. Using **hist()** function -

```
hist(main_energy$X4,
     main="Roof Area Distribution",
     xlab="Value in m²",
     col="darkred"
)
```



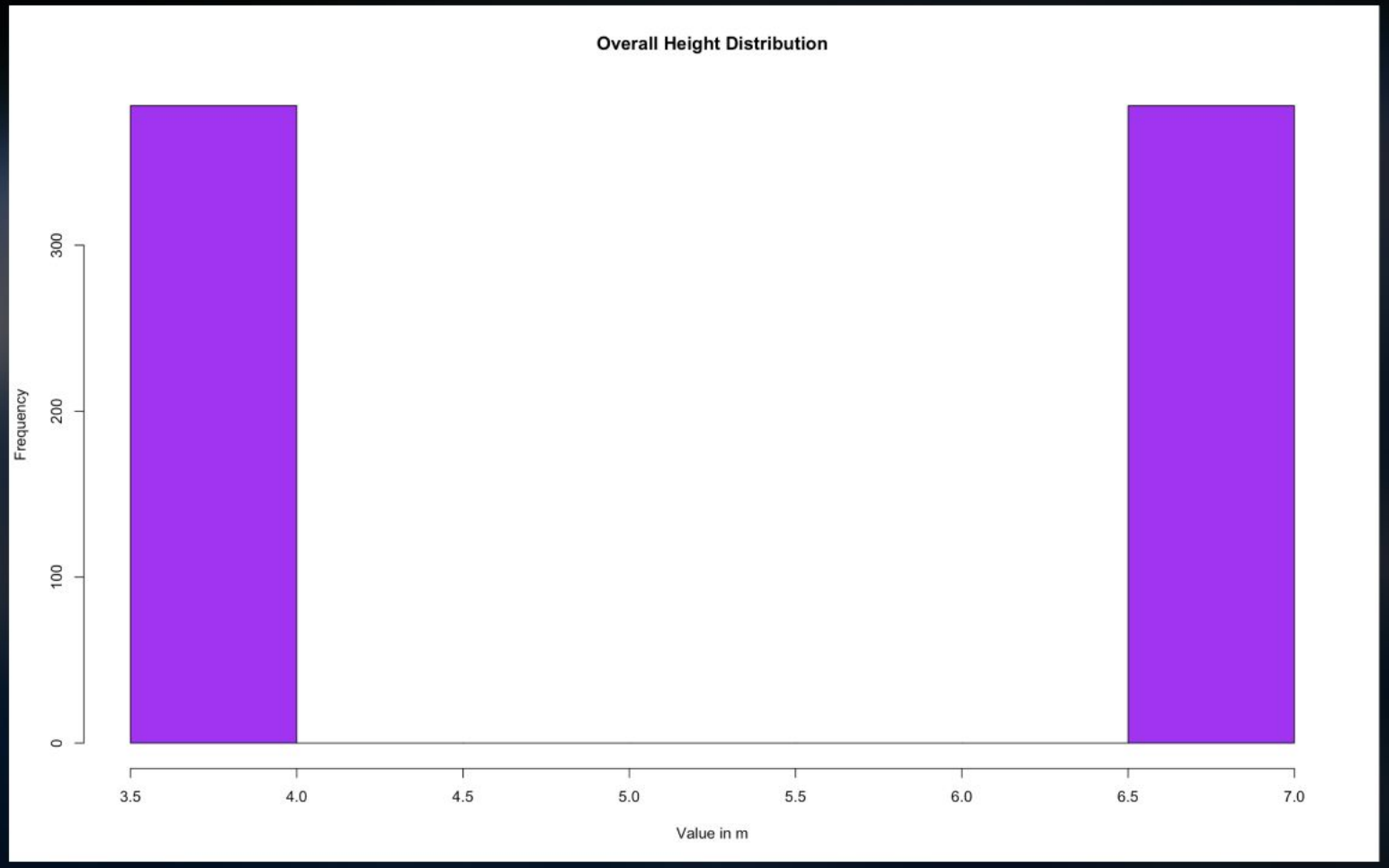
Histograms

Histogram of
Overall Height

Steps -

1. Using **hist()** function -

```
hist(main_energy$X5,
     main="Overall Height Distribution",
     xlab="Value in m",
     col="purple"
)
```



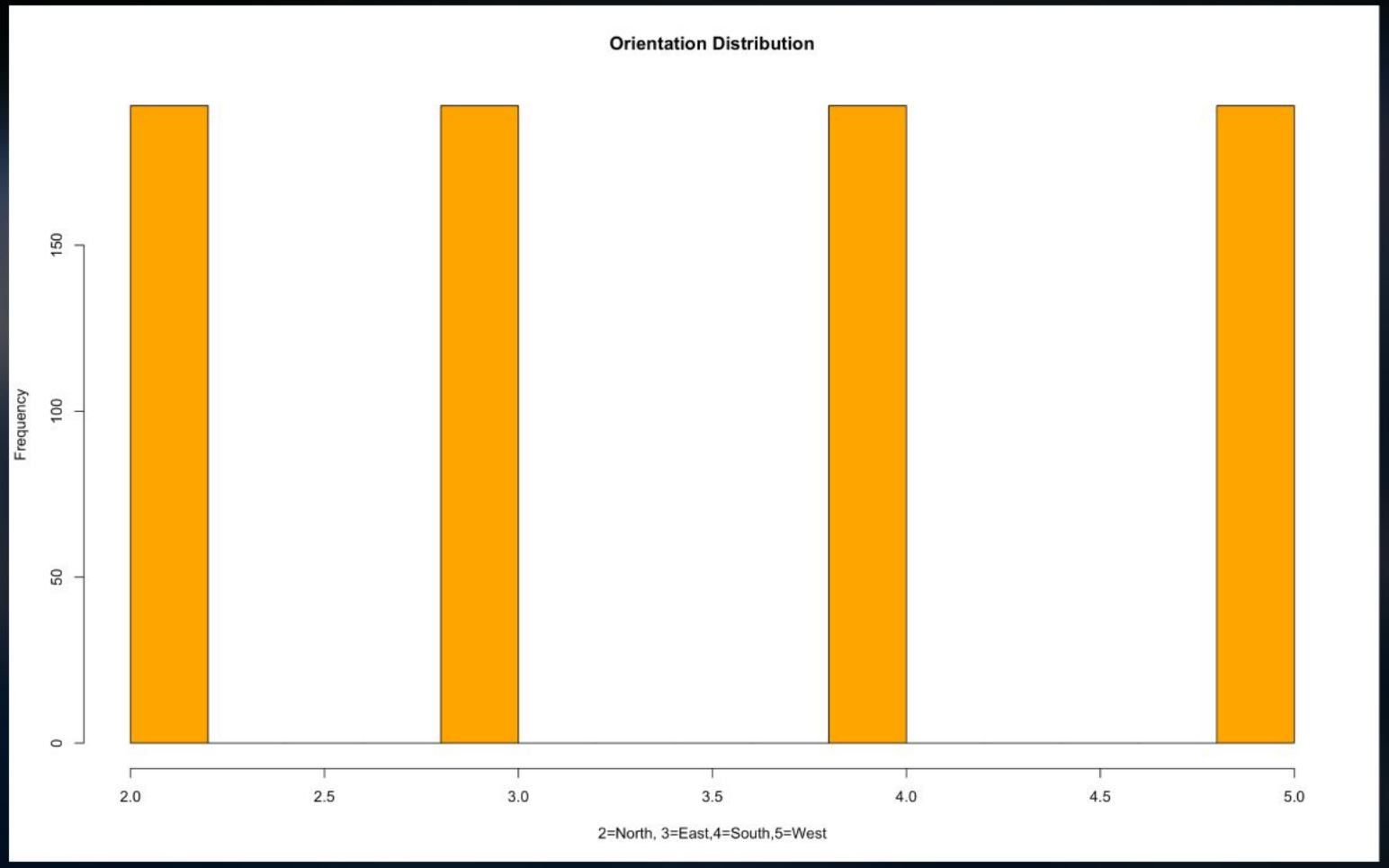
Histograms

Histogram of
Orientation Distribution

Steps -

1. Using **hist()** function -

```
hist(main_energy$X6,
     main="Orientation Distribution",
     xlab="2=North, 3=East,4=South,5=West",
     col="orange"
)
```



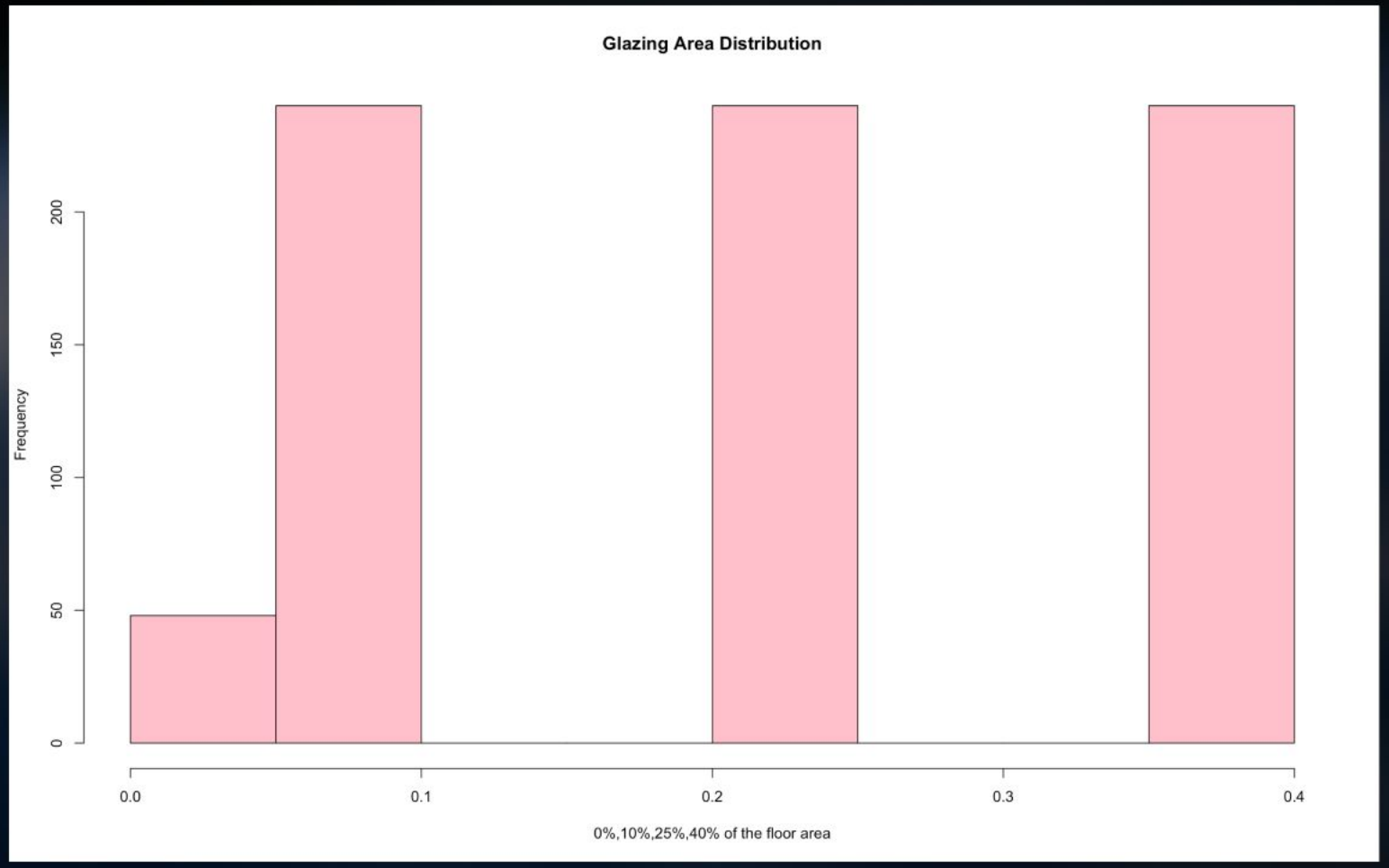
Histograms

Histogram of
Glazing Area

Steps -

1. Using **hist()** function -

```
hist(main_energy$X7,
     main="Glazing Area Distribution",
     xlab="0%,10%,25%,40% of the floor area",
     col="pink"
)
```



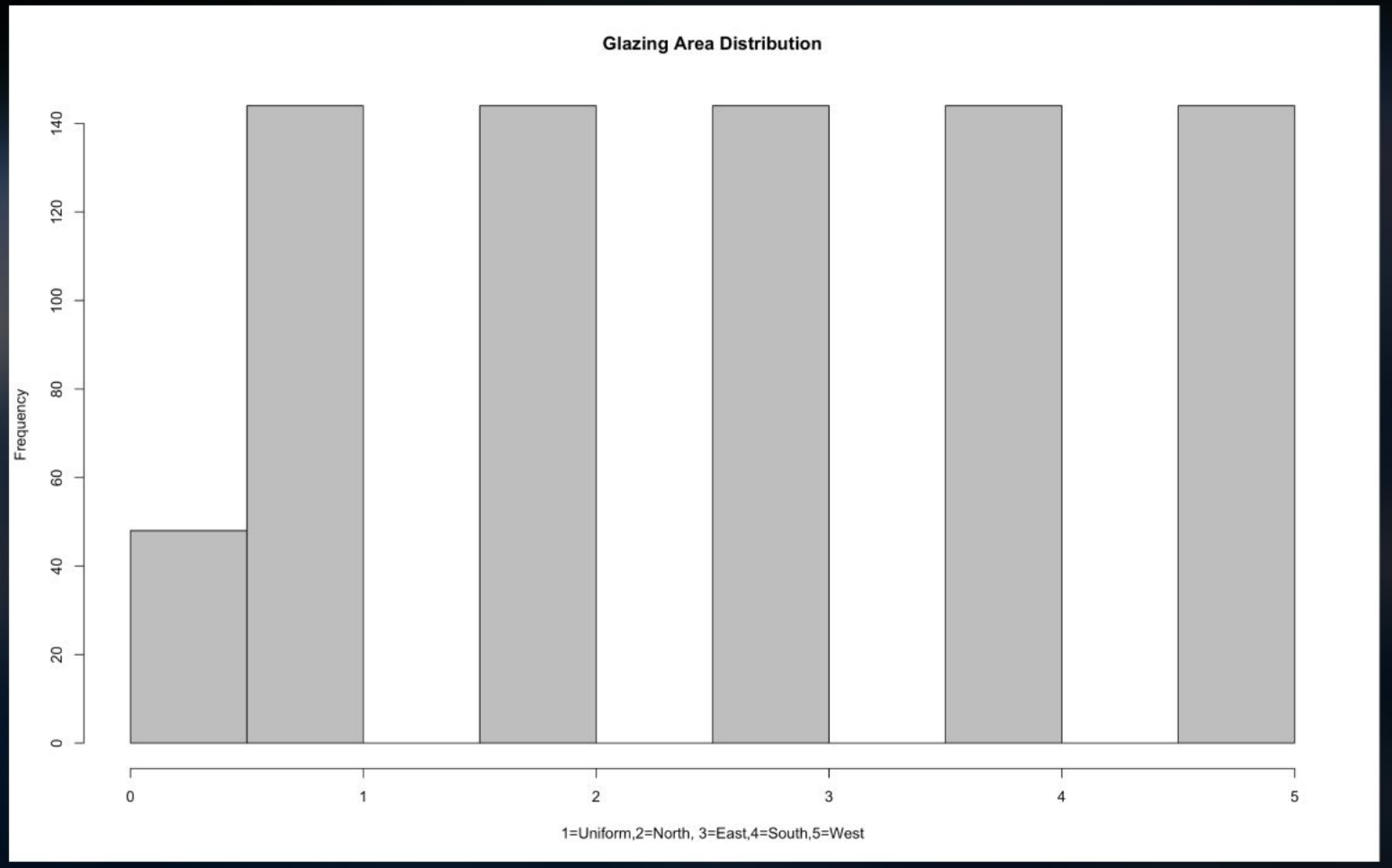
Histograms

Histogram of
Glazing Area Dist.

Steps -

1. Using **hist()** function -

```
hist(main_energy$X8,
     main="Glazing Area Distribution",
     xlab="1=Uniform,2=North, 3=East,4=South,5=West",
     col="grey"
)
```



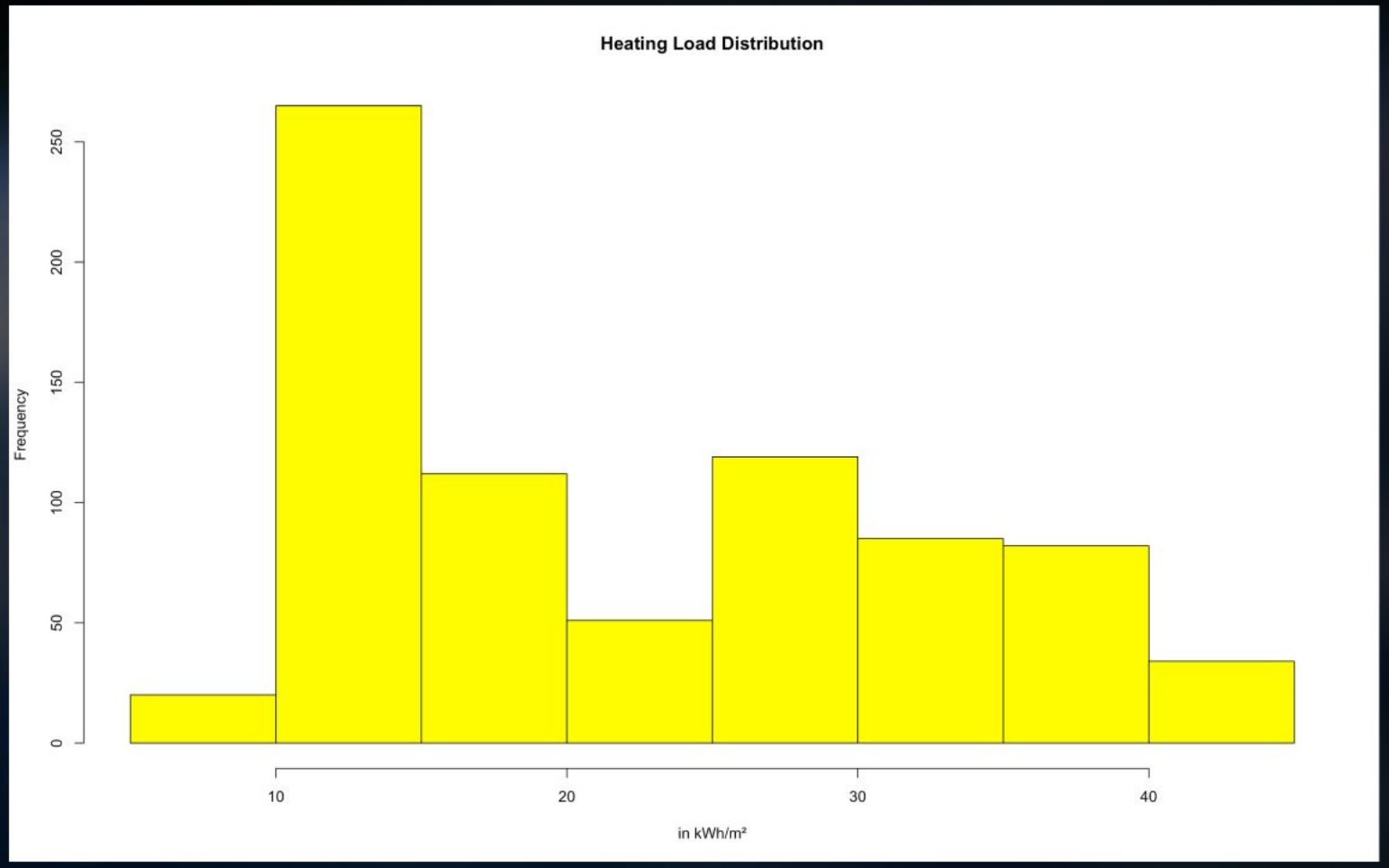
Histograms

Histogram of
y1(Heating Load)

Steps -

1. Using **hist()** function -

```
hist(main_energy$Y1,
     main="Heating Load Distribution",
     xlab="in kWh/m²",
     col="yellow"
)
```



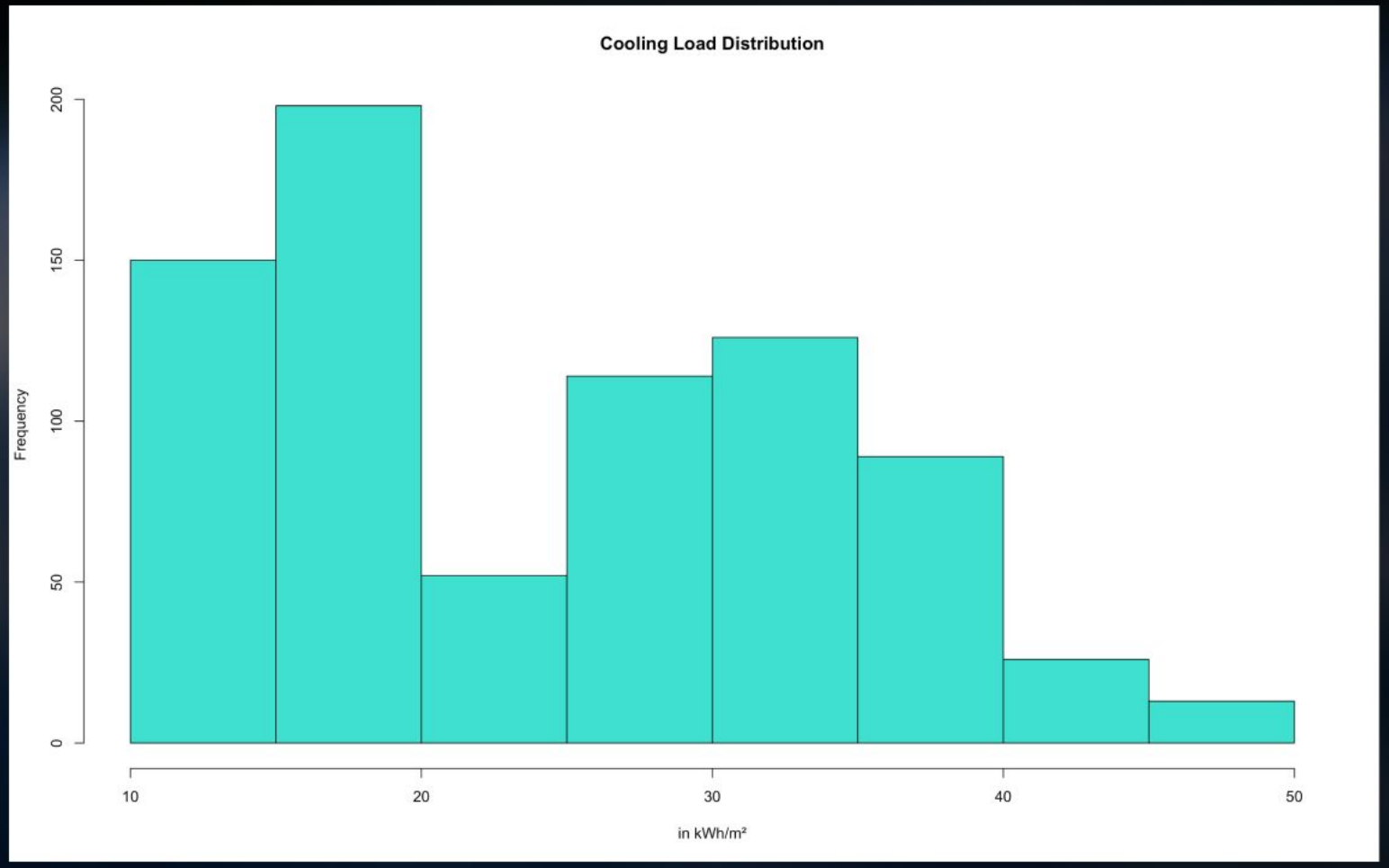
Histograms

Histogram of
y2(Cooling Load)

Steps -

1. Using **hist()** function -

```
hist(main_energy$Y2,
     main="Cooling Load Distribution",
     xlab="in kWh/m²",
     col="turquoise"
)
```



Descriptive Statistics

Overall **descriptive statistics** - mean, standard deviation, Inter-Quartile Region (IQR), skewness, kurtosis, quantiles and number of observations of all continuous variables. dependent variable (y)

```
library(Rcmdr)
numSummary(main_energy[,c("X1","X2","X3","X4","X5","X6","X7","X8","Y1","Y2"), drop=FALSE],
statistics = c("mean","sd","IQR","quantiles","skewness","kurtosis"), quantiles=c(0,.25,.50,.75,1),type="2")
```

	mean	sd	IQR	skewness	kurtosis	0%	25%	50%	75%	100%	n
X1	0.7641667	0.1057775	0.1475	0.49551251	-0.7065675	0.62	0.6825	0.75	0.8300	0.98	768
X2	671.7083333	88.0861161	134.7500	-0.12513088	-1.0594542	514.50	606.3750	673.75	741.1250	808.50	768
X3	318.5000000	43.6264814	49.0000	0.53341749	0.1165933	245.00	294.0000	318.50	343.0000	416.50	768
X4	176.6041667	45.1659502	79.6250	-0.16276400	-1.7769472	110.25	140.8750	183.75	220.5000	220.50	768
X5	5.2500000	1.7511404	3.5000	0.00000000	-2.0052288	3.50	3.5000	5.25	7.0000	7.00	768
X6	3.5000000	1.1187626	1.5000	0.00000000	-1.3610425	2.00	2.7500	3.50	4.2500	5.00	768
X7	0.2343750	0.1332206	0.3000	-0.06025423	-1.3276291	0.00	0.1000	0.25	0.4000	0.40	768
X8	2.8125000	1.5509597	2.2500	-0.08868918	-1.1487088	0.00	1.7500	3.00	4.0000	5.00	768
Y1	22.3071953	10.0902040	18.6750	0.36044568	-1.2455687	6.01	12.9925	18.95	31.6675	43.10	768
Y2	24.5877604	9.5133056	17.5125	0.39599247	-1.1471903	10.90	15.6200	22.08	33.1325	48.03	768

Correlation

Correlation is a statistical measure (expressed as a number) that describes the size and direction of a relationship between two or more variables. A correlation between variables, however, does not automatically mean that the change in one variable is the cause of the change in the values of the other variable.

```
myData <- main_energy[,c(1:10)]
res <- cor(myData)
round(res,4)
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
X1	1.0000	-0.9919	-0.2038	-0.8688	0.8277	0.0000	0.0000	0.0000	0.6223	0.6343
X2	-0.9919	1.0000	0.1955	0.8807	-0.8581	0.0000	0.0000	0.0000	-0.6581	-0.6730
X3	-0.2038	0.1955	1.0000	-0.2923	0.2810	0.0000	0.0000	0.0000	0.4557	0.4271
X4	-0.8688	0.8807	-0.2923	1.0000	-0.9725	0.0000	0.0000	0.0000	-0.8618	-0.8625
X5	0.8277	-0.8581	0.2810	-0.9725	1.0000	0.0000	0.0000	0.0000	0.8894	0.8958
X6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	-0.0026	0.0143
X7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.2130	0.2698	0.2075
X8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2130	1.0000	0.0874	0.0505
Y1	0.6223	-0.6581	0.4557	-0.8618	0.8894	-0.0026	0.2698	0.0874	1.0000	0.9759
Y2	0.6343	-0.6730	0.4271	-0.8625	0.8958	0.0143	0.2075	0.0505	0.9759	1.0000

Correlation

Correlation of all the variables in descending order with respect with **Heating Load (y1)**.

```
library(dplyr)
library(tidyr)

myData<-data.frame(EnergyEfficiency)
cor(myData[,c(1:9)])%>%
  as.data.frame(myData)%>%
  mutate(var1=rownames(.))%>%
  gather(var2, value, -var1)%>%
  arrange(desc(value))%>%
  group_by(value)%>%
  filter(row_number()==1)
```

```
# Groups:   value [29]
   var1  var2  value
   <chr> <chr> <dbl>
 1 1     X1    1
 2 9     X5    0.889
 3 4     X2    0.881
 4 5     X1    0.828
 5 9     X1    0.622
 6 9     X3    0.456
 7 5     X3    0.281
 8 9     X7    0.270
 9 8     X7    0.213
10 3     X2    0.196
```

Top 5 Correlation of all variables with Heating Load (y1) -

Rank -

1. **y1 vs Overall Height**
2. **Surface Area vs Roof Area**
3. **Relative Compactness vs Overall Height**
4. **y1 vs Relative Compactness**
5. **y1 vs Wall Area**

Correlation

Correlation of all the variables in descending order with respect with **Cooling Load (y2)**.

```
library(dplyr)
library(tidyr)
myData<-data.frame(corMatrix)
cor(myData[,c("X1", "X2", "X3", "X5", "X6", "X7", "X8", "Y2")])%>%
              as.data.frame(myData)%>%
              mutate(var1 = rownames(.))%>%
              gather(var2, value, -var1)%>%
              arrange(desc(value))%>%
              group_by(value)%>%
              filter(row_number()==1)
```

```
# Groups:   value [22]
   var1  var2   value
   <chr> <chr>  <dbl>
 1 1     X1     1
 2 8     X5     0.896
 3 4     X1     0.828
 4 8     X1     0.634
 5 8     X3     0.427
 6 4     X3     0.281
 7 7     X7     0.213
 8 8     X7     0.208
 9 3     X2     0.196
10 8     X8     0.0505
```

Top 5 Correlation of all variables with Cooling Load (y2) -

Rank -

1. **Glazing Area Distribution vs Overall Height**
2. **Relative Compactness vs Roof Area**
3. **Relative Compactness vs Glazing Area Distribution**
4. **Wall Area vs Glazing Area Distribution**
5. **Roof Area vs Wall Area**

Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. We regress all the variables keeping **y1(Heating Load)** as dependent variable.

```
lm(formula = Y1 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data = energyefficiency)

Residuals:
    Min      1Q  Median      3Q     Max
-9.8965 -1.3196 -0.0252  1.3532  7.7052

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  84.013418  19.033613   4.414 1.16e-05 ***
X1          -64.773432  10.289448  -6.295 5.19e-10 ***
X2           -0.087289   0.017075  -5.112 4.04e-07 ***
X3            0.060813   0.006648   9.148  < 2e-16 ***
X4                  NA         NA      NA       NA
X5            4.169954   0.337990  12.338  < 2e-16 ***
X6           -0.023330   0.094705  -0.246  0.80548
X7           19.932736   0.813986  24.488  < 2e-16 ***
X8            0.203777   0.069918   2.915  0.00367 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.934 on 760 degrees of freedom
Multiple R-squared:  0.9162, Adjusted R-squared:  0.9154
```

Inferences -

1. Adjusted R-Squared value is very high **91.54%.**
2. X4 variable shows **NA**, which means “not defined due to singularities”. This error occurs when two or more variables in the dataset are strongly correlated or perfectly collinear.
3. All variables are statistically significant instead of X6.

Linear Regression

Hence, we drop **X4** and regress again with all the variables keeping y1(Heating Load) as dependent variable.

```
Call:
lm(formula = Y1 ~ X1 + X2 + X3 + X5 + X6 + X7 + X8, data = energyefficiency)

Residuals:
    Min      1Q  Median      3Q     Max
-9.8965 -1.3196 -0.0252  1.3532  7.7052

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  84.013418  19.033613   4.414 1.16e-05 ***
X1          -64.773432  10.289448  -6.295 5.19e-10 ***
X2           -0.087289   0.017075  -5.112 4.04e-07 ***
X3            0.060813   0.006648   9.148  < 2e-16 ***
X5            4.169954   0.337990  12.338  < 2e-16 ***
X6           -0.023330   0.094705  -0.246  0.80548
X7           19.932736   0.813986  24.488  < 2e-16 ***
X8            0.203777   0.069918   2.915  0.00367 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.934 on 760 degrees of freedom
Multiple R-squared:  0.9162, Adjusted R-squared:  0.9154
```

Inferences -

1. Adjusted R-Squared value is very high **91.54%**.
2. All variables are statistically significant instead of X6.

Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. We regress all the variables keeping **y2(Cooling Load)** as dependent variable.

```
lm(formula = Y2 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data = energyefficiency)

Residuals:
    Min      1Q  Median      3Q     Max
-8.6940 -1.5606 -0.2668  1.3968 11.1775

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  97.245749  20.764711   4.683 3.34e-06 ***
X1          -70.787707  11.225269  -6.306 4.85e-10 ***
X2           -0.088245   0.018628  -4.737 2.59e-06 ***
X3            0.044682   0.007253   6.161 1.17e-09 ***
X4                  NA         NA      NA       NA
X5            4.283843   0.368730  11.618  < 2e-16 ***
X6            0.121510   0.103318   1.176    0.240
X7           14.717068   0.888018  16.573  < 2e-16 ***
X8            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
```

Inferences -

1. Adjusted R-Squared value is very high **88.68%.**
2. X4 variable shows **NA**, which means “not defined due to singularities”. This error occurs when two or more variables in the dataset are strongly correlated or perfectly collinear.
3. All variables are statistically significant instead of X6 & X8.

Linear Regression

Hence, we drop **X4, X6 and X8,** regress again with all the variables keeping y2(Cooling Load) as dependent variable.

```
> summary(RegModel.5)

Call:
lm(formula = Y2 ~ X1 + X2 + X3 + X5 + X7, data = energyefficiency)

Residuals:
    Min      1Q  Median      3Q     Max
-8.7240 -1.6017 -0.2631  1.3417 11.3251

Coefficients:
             Estimate Std. Error t value      Pr(>|t|)
(Intercept)  97.761848  20.756339   4.710 0.00000294390 ***
X1          -70.787707  11.222822  -6.307 0.00000000048 ***
X2           -0.088245   0.018624  -4.738 0.00000257328 ***
X3            0.044682   0.007251   6.162 0.00000000116 ***
X5            4.283843   0.368650  11.620       < 2e-16 ***
X7           14.817971   0.867458  17.082       < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.2 on 762 degrees of freedom
Multiple R-squared:  0.8876, Adjusted R-squared:  0.8868
```

Inferences -

1. Adjusted R-Squared value is very high **88.68%**.
2. All variables are statistically significant.

Variance Inflation Factor (VIF)

VIF is a measure of the amount of multicollinearity in regression analysis. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model.

Steps -

1. Conduct linear regression with “y1” and “y2” as dependent variable and columns 1-8 as independent variables using Rcmdr.
2. Run numerical diagnosis using ‘vif’ function.
3. Interpreting the results making note of variables having VIF value less than 1, between 1 and 5 and greater than 5.

VIF OUTPUT	
X2	201.531134
X1	105.524054
X5	31.205474
X3	7.492984
X7	1.047508
X8	1.047508
X6	1

Variance Inflation Factor (VIF)

In general terms,

- VIF greater than 5 = variables are highly correlated & higher posibility of multicollinearity to exist.
- Hence, we will drop the variables having a VIF value of 10 and above. i.e. **X1, X2, X5.**

VIF OUTPUT	
X2	201.531134
X1	105.524054
X5	31.205474
X3	7.492984
X7	1.047508
X8	1.047508
X6	1

Linear Regression

We drop the highly multicollinear variables and try doing linear regression again, but now with dummy variables. Y1(Heating Load) as the dependent variable.

```
Call:
lm(formula = Y1 ~ X3 + X4 + X6E + X6S + X6W + X7 + X8E + X8N +
    X8S + X8U + X8W, data = NehalEnergyEfficiency)

Residuals:
    Min      1Q  Median      3Q     Max
-9.3576 -2.1323 -0.4584  1.0318 11.0464

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 29.314061   1.402874  20.896  < 2e-16 ***
X3           0.051526   0.003174  16.233  < 2e-16 ***
X4          -0.177986   0.003066 -58.052  < 2e-16 ***
X6E          0.067813   0.374322   0.181    0.856
X6S         -0.052990   0.374322  -0.142    0.887
X6W         -0.037500   0.374322  -0.100    0.920
X7          16.848333   1.116013  15.097  < 2e-16 ***
X8E          4.183000   0.671929   6.225 7.96e-10 ***
X8N          4.435986   0.671929   6.602 7.64e-11 ***
X8S          4.388208   0.671929   6.531 1.20e-10 ***
X8U          4.527653   0.671929   6.738 3.18e-11 ***
X8W          4.182444   0.671929   6.225 8.00e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.668 on 756 degrees of freedom
Multiple R-squared:  0.8698,    Adjusted R-squared:  0.8679
F-statistic:   459 on 11 and 756 DF,  p-value: < 2.2e-16
```

```
#Linear Regression
RegModel.1 <- lm(Y1~X3+X4+X6E+X6S+X6W+X7+X8E+X8N+X8S+X8U+X8W, data=NehalEnergyEfficiency)
summary(RegModel.1)
```

Inferences -

1. Adjusted R-Squared value is very high **86.79%**.
2. All variables are statistically significant instead of dummies created for **X6 (X6E, X6S and X6W)**.
3. Rest all variables are statistically significant and directly affect the Heating Load.

Linear Regression

We drop the highly multicollinear variables and try doing linear regression again, but now with dummy variables. Y2(Cooling Load) as the dependent variable.

```
Call:
lm(formula = Y2 ~ X3 + X4 + X6E + X6S + X6W + X7 + X8E + X8N +
    X8S + X8U + X8W, data = Y2)

Residuals:
    Min      1Q  Median      3Q     Max
-8.9934 -2.1244 -0.5079  1.1454 14.8103

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.439323   1.540376  23.656  < 2e-16 ***
X3           0.041722   0.003485  11.971  < 2e-16 ***
X4          -0.169898   0.003366 -50.467  < 2e-16 ***
X6E         -0.291979   0.411011  -0.710  0.47768
X6S         -0.124219   0.411011  -0.302  0.76256
X6W          0.349115   0.411011   0.849  0.39593
X7          13.252917   1.225399  10.815  < 2e-16 ***
X8E          1.639965   0.737788   2.223  0.02652 *
X8N          1.977396   0.737788   2.680  0.00752 **
X8S          1.995660   0.737788   2.705  0.00699 **
X8U          2.160035   0.737788   2.928  0.00352 **
X8W          1.695521   0.737788   2.298  0.02183 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.027 on 756 degrees of freedom
Multiple R-squared:  0.8234,    Adjusted R-squared:  0.8208
F-statistic: 320.4 on 11 and 756 DF,  p-value: < 2.2e-16
```

```
#Linear Regression
RegModel.3 <- lm(Y2~X3+X4+X6E+X6S+X6W+X7+X8E+X8N+X8S+X8U+X8W, data=Y2)
summary(RegModel.3)
```

Inferences -

1. Adjusted R-Squared value is very high **82.08%**.
2. All variables are statistically significant instead of dummies created for **X6 (X6E, X6S and X6W)**.
3. Rest all variables are statistically significant and directly affect the Cooling Load.

Categorization

Assigning categories for each Cooling load and Heating load (A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile

```
> quantile(main_energy$Y1)
     0%     25%     50%     75%    100% 
 6.0100 12.9925 18.9500 31.6675 43.1000 
> quantile(main_energy$Y2)
     0%     25%     50%     75%    100% 
10.9000 15.6200 22.0800 33.1325 48.0300 
```

```
quantile(main_energy$Y1)
#Categories for heating load -
#A = Highest Quantile - 75% to 100% - 31.6675 to 43.1
#B = Second highest Quantile - 50% to 75% - 18.95 to 31.6675
#C = Third highest Quantile - 25% to 50% - 12.9925 to 18.95
#D = Lowest Quantile - 0% to 25% - 6.01 to 12.9925

quantile(main_energy$Y2)
#Categories for cooling load -
#A = Highest Quantile - 75% to 100% - 33.1325 to 48.03
#B = Second highest Quantile - 50% to 75% - 22.08 to 33.1325
#C = Third highest Quantile - 25% to 50% - 15.62 to 22.08
#D = Lowest Quantile - 0% to 25% - 10.9 to 15.62
```

Perceptron

Case 1:

For Perceptron, we assigned 1 for categories A and B, -1 for categories C and D for **y1 (Heating Load)** and used all variables (including dummy variables for X6 and X8) and excluding X1, X2 and X5 for classification.

```r
#Perceptrons
energySubset2D <- energyd[, c("X3","X4","X6E","X6S","X6W","X7","X8U","X8N","X8E","X8S","X8W","Y1_dummy")]
energySubset2D$class <- lapply(energySubset2D$Y1_dummy, function(x) {
  if(x == '-1')
    energySubset2D$class <- -1
  else if(x == '1')
    energySubset2D$class <- 1
  else
    energySubset2D$class <- NULL
})
X <- energySubset2D[, c("X3","X4","X6E","X6S","X6W","X7","X8U","X8N","X8E","X8S","X8W")] # Input Matrix
y <- energySubset2D$class # Output Vector
perceptron <- function(X, y, numEpochs) {
  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)
  }
}
results = perceptron(X,y, 8)
```

Perceptron

Case 1: For Perceptron, we assigned 1 for categories A and B, -1 for categories C and D for **y1(Heating Load)** and used all variables ((including dummy variables for X6 and X8) and excluding X1, X2 and X5 for classification.
We got 12 incorrect classification.

```
Epoch #:  1
Number Incorrect:  33
Final Weight:  510.1409 -1019.396 9.673677 -6.995507 -5.589554 -3.949053 8.624786 -3.518943 6.420059 9.028215 10.67298
Epoch #:  2
Number Incorrect:  12
Final Weight:  534.6409 -1007.146 8.673677 -6.995507 -6.589554 -3.949053 8.624786 -3.518943 6.420059 9.028215 10.67298
Epoch #:  3
Number Incorrect:  25
Final Weight:  755.1409 -1362.396 7.673677 -6.995507 -7.589554 -3.599053 10.62479 -3.518943 6.420059 9.028215 10.67298
Epoch #:  4
Number Incorrect:  20
Final Weight:  632.6409 -1680.896 6.673677 -6.995507 -8.589554 -3.499053 11.62479 -3.518943 6.420059 9.028215 10.67298
Epoch #:  5
Number Incorrect:  21
Final Weight:  706.1409 -1950.396 5.673677 -6.995507 -9.589554 -3.399053 11.62479 -2.518943 6.420059 9.028215 10.67298
Epoch #:  6
Number Incorrect:  11
Final Weight:  1024.641 -1827.896 4.673677 -6.995507 -10.58955 -3.399053 11.62479 -2.518943 6.420059 9.028215 10.67298
Epoch #:  7
Number Incorrect:  16
Final Weight:  975.6409 -1889.146 2.673677 -6.995507 -11.58955 -3.299053 12.62479 -2.518943 6.420059 9.028215 10.67298
Epoch #:  8
Number Incorrect:  12
Final Weight:  1000.141 -1876.896 1.673677 -6.995507 -12.58955 -3.299053 12.62479 -2.518943 6.420059 9.028215 10.67298
```

Perceptron

Case 2:

For Perceptron, we assigned 1 for categories A and B, -1 for categories C and D for **y2(Cooling Load)** and used all variables (including dummy variables for X6 and X8) and excluding X1, X2 and X5 for classification.

```r
#Perceptrons
energySubset2D <- energyd[, c("X3","X4","X6E","X6S","X6W","X7","X8U","X8N","X8E","X8S","X8W","Y2_dummy")]
energySubset2D$class <- lapply(energySubset2D$Y2_dummy, function(x) {
  if(x == '-1')
    energySubset2D$class <- -1
  else if(x == '1')
    energySubset2D$class <- 1
  else
    energySubset2D$class <- NULL
})
X <- energySubset2D[, c("X3","X4","X6E","X6S","X6W","X7","X8U","X8N","X8E","X8S","X8W")] # Input Matrix
y <- energySubset2D$class # Output Vector
perceptron <- function(X, y, numEpochs) {
  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)
  }
}
results = perceptron(X,y, 8)
```

Perceptron

Case 2: For Perceptron, we assigned 1 for categories A and B, -1 for categories C and D for **y2(Cooling Load)** and used all variables ((including dummy variables for X6 and X8) and excluding X1, X2 and X5 for classification.
We got 12 incorrect classification.

```
Epoch #:  1
Number Incorrect:  33
Final Weight:  504.5202 -1023.389 4.248724 -5.641808 -7.187055 -9.623919 -0.773408 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  2
Number Incorrect:  16
Final Weight:  553.5202 -1158.139 2.248724 -2.641808 -7.187055 -9.623919 -0.773408 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  3
Number Incorrect:  20
Final Weight:  529.0202 -1439.889 0.2487238 0.3581925 -7.187055 -9.623919 -0.773408 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  4
Number Incorrect:  13
Final Weight:  749.5202 -1427.639 -1.751276 3.358192 -7.187055 -9.523919 0.226592 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  5
Number Incorrect:  12
Final Weight:  725.0202 -1415.389 -3.751276 6.358192 -7.187055 -9.523919 0.226592 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  6
Number Incorrect:  14
Final Weight:  700.5202 -1476.639 -5.751276 9.358192 -7.187055 -9.523919 0.226592 -9.632043 -3.991051 -8.781049 -7.762916
Epoch #:  7
Number Incorrect:  21
Final Weight:  872.0202 -1574.639 -7.751276 11.35819 -7.187055 -9.423919 0.226592 -9.632043 -3.991051 -7.781049 -7.762916
Epoch #:  8
Number Incorrect:  12
Final Weight:  847.5202 -1562.389 -9.751276 14.35819 -7.187055 -9.423919 0.226592 -9.632043 -3.991051 -7.781049 -7.762916
```

Support Vector Machine

Assigning categories for each Cooling load and Heating load (A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile and then performing SVM technique.

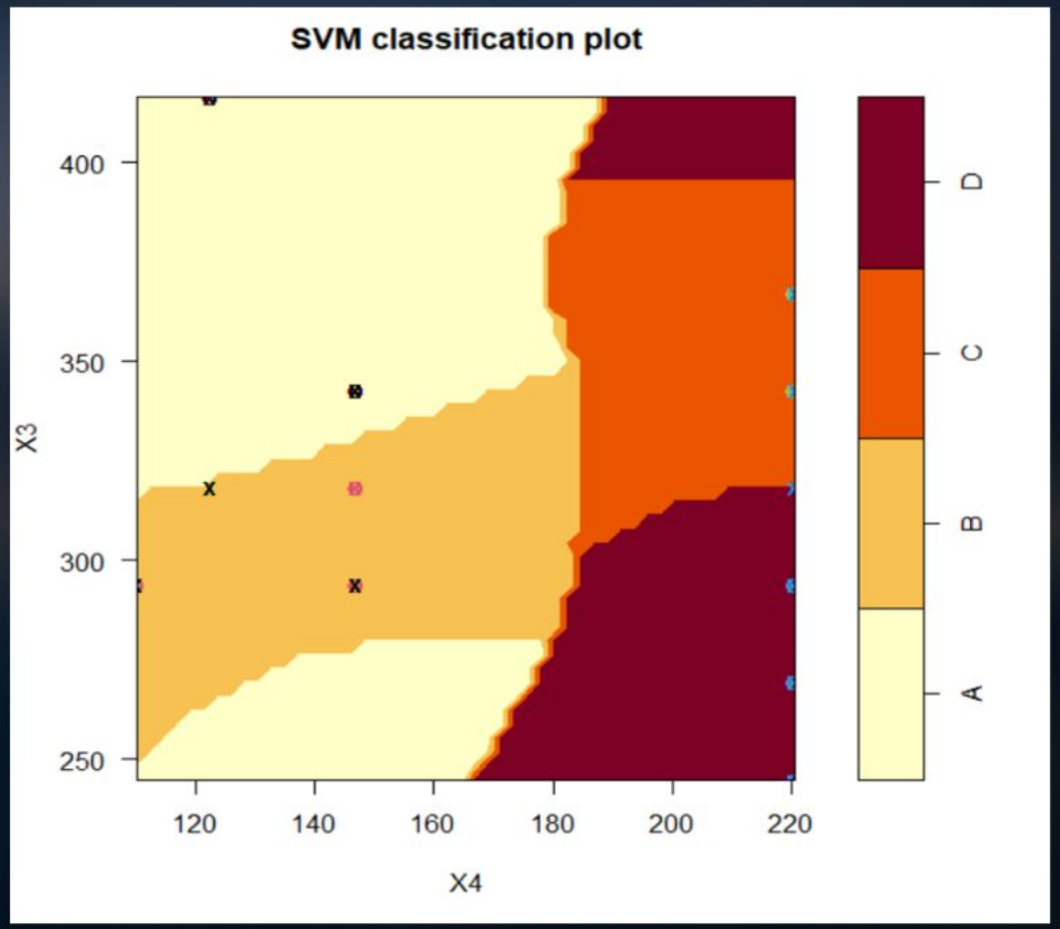
```
> quantile(main_energy$Y1)
     0%     25%     50%     75%    100% 
 6.0100 12.9925 18.9500 31.6675 43.1000 
> quantile(main_energy$Y2)
     0%     25%     50%     75%    100% 
10.9000 15.6200 22.0800 33.1325 48.0300 
```

```
quantile(main_energy$Y1)
#Categories for heating load -
#A = Highest Quantile - 75% to 100% - 31.6675 to 43.1
#B = Second highest Quantile - 50% to 75% - 18.95 to 31.6675
#C = Third highest Quantile - 25% to 50% - 12.9925 to 18.95
#D = Lowest Quantile - 0% to 25% - 6.01 to 12.9925

quantile(main_energy$Y2)
#Categories for cooling load -
#A = Highest Quantile - 75% to 100% - 33.1325 to 48.03
#B = Second highest Quantile - 50% to 75% - 22.08 to 33.1325
#C = Third highest Quantile - 25% to 50% - 15.62 to 22.08
#D = Lowest Quantile - 0% to 25% - 10.9 to 15.62
```

Support Vector Machine

svm (Y1_cat ~ X3+X4, data=energyefficiency)

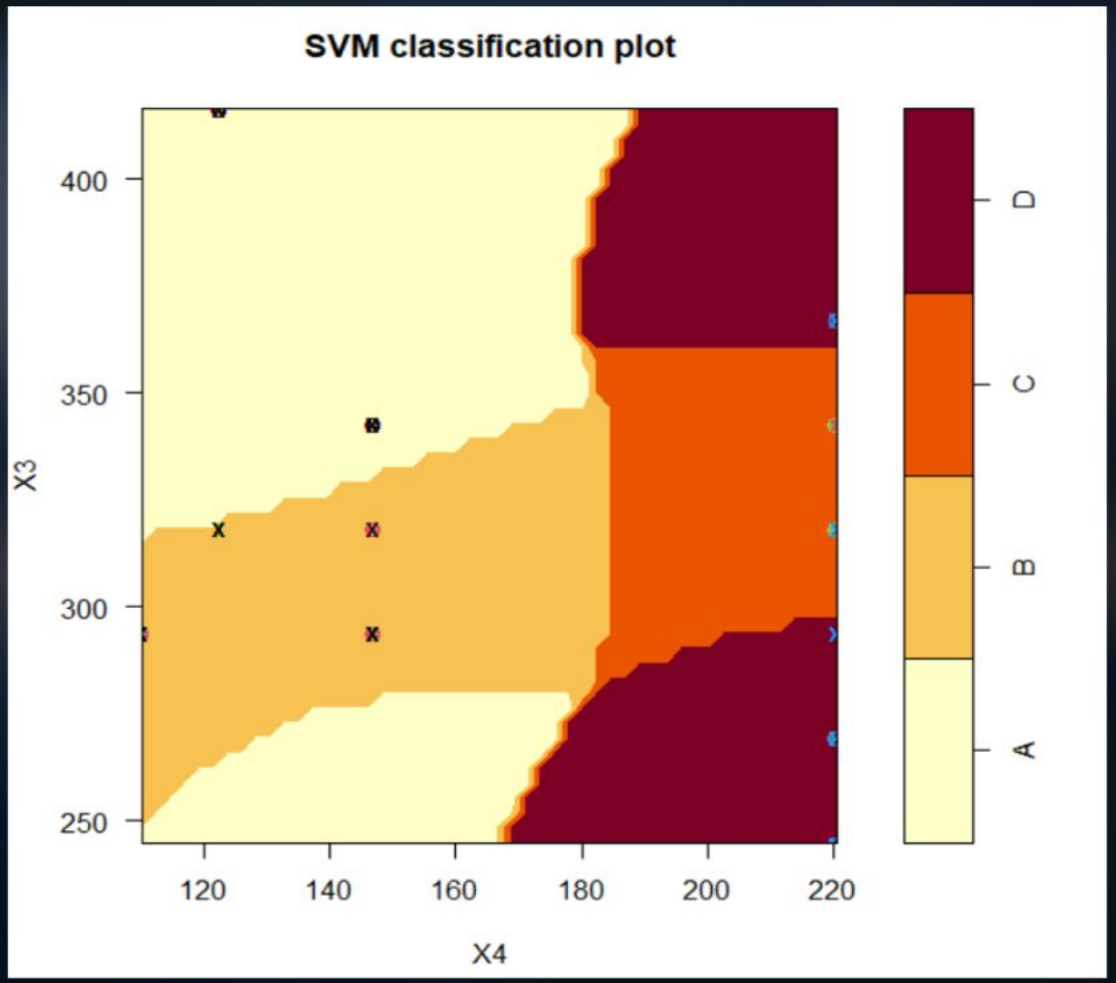
```
> library(e1071)
> energyefficiency$Y1_cat <- as.factor(energyefficiency$Y1_cat)
> model1 <- svm(Y1_cat ~ X3+X4, data=energyefficiency)
> print(model1)

Call:
svm(formula = Y1_cat ~ X3 + X4, data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  449

> summary(model1)

Call:
svm(formula = Y1_cat ~ X3 + X4, data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  449

 ( 128 107 119 95 )


Number of Classes:  4

Levels:
 A B C D


> plot(model1,energyefficiency)
```



Support Vector Machine

svm(Y2_cat ~ X3+X4, data=energyefficiency)

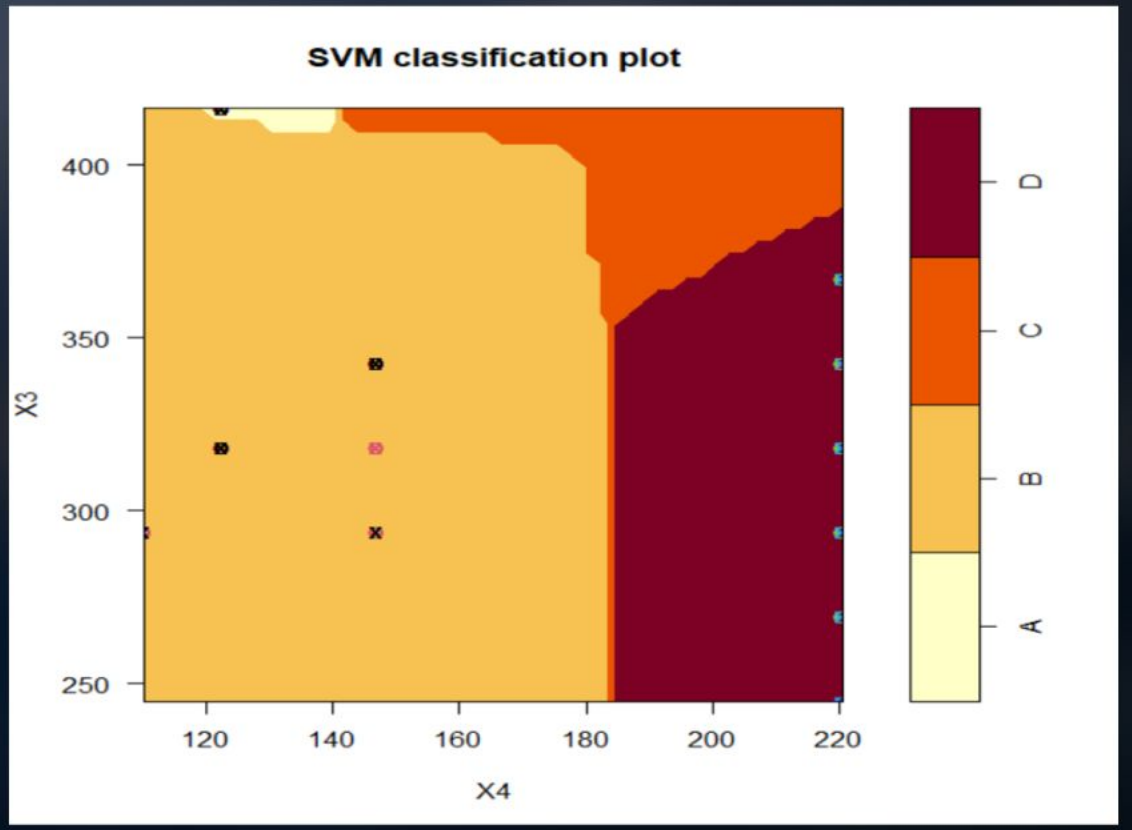
```
Call:
svm(formula = Y2_cat ~ X3 + X4, data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  465

> summary(model1)

Call:
svm(formula = Y2_cat ~ X3 + X4, data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  465

 ( 140 100 91 134 )


Number of Classes:  4

Levels:
 A B C D



> plot(model1,energyefficiency)
>
```



Support Vector Machine

SVM 3- plot(model1, energyefficiency, X3 ~ X4, slice = list(X6E=0, X6S=1, X6W=0, X7=0, X8U=0, X8N=0, X8E=0, X8S=0, X8W=0))

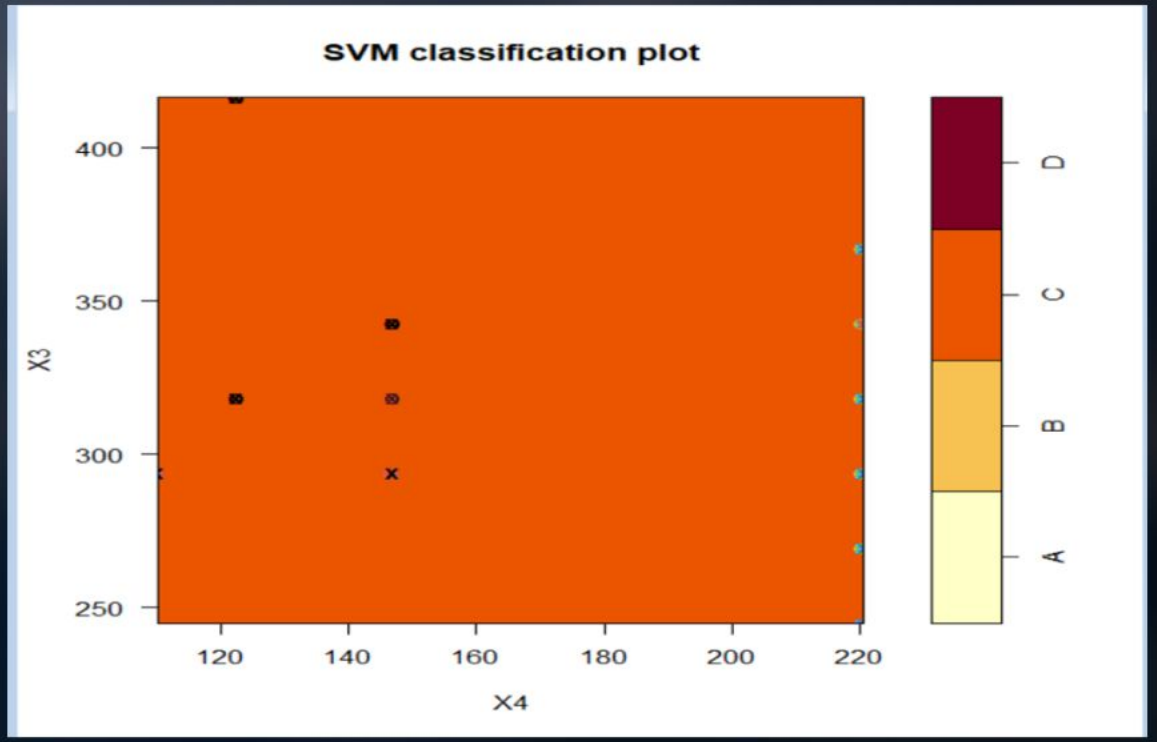
```
Call:
svm(formula = Y1_cat ~ ., data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  568

> summary(model1)

Call:
svm(formula = Y1_cat ~ ., data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  568

 ( 157 154 113 144 )


Number of Classes:  4

Levels:
 A B C D


> plot(model1, energyefficiency, X3 ~ X4, slice = list(X6E=0, X6S=1, X6W=0, X7=0, X8U=0, X8N=0, X8E=0, X8S=0, X8W=0))
```



Support Vector Machine

plot(model1, energyefficiency, X3 ~ X4, slice = list(X6E=0, X6S=1, X6W=0, X7=0, X8U=0, X8N=0, X8E=0, X8S=0, X8W=0))

```
Call:
svm(formula = Y2_cat ~ ., data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  556

> summary(model1)

Call:
svm(formula = Y2_cat ~ ., data = energyefficiency)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  556

 ( 131 153 146 126 )


Number of Classes:  4

Levels:
 A B C D


> plot(model1, energyefficiency, X3 ~ X4, slice = list(X6E=0, X6S=1, X6W=0, X7=0, X8U=0, X8N=0, X8E=0, X8S=0, X8W=0))
```



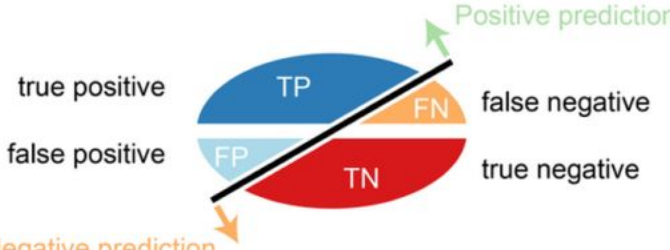
Support Vector Machine

```r
str(energy)
energy_svm_Y2 <- svm(formula = Y2_cat ~ X3 + X4 +
                       X6  + X7 + X8 ,data = training_set, type = 'C-classification',
                     kernel = 'linear')
Y2_cat_pred = predict(energy_svm_Y2, newdata = test_set)

conMat <- confusionMatrix(Y2_cat_pred, test_set$Y2_cat)
conMat

f1_y2 <- F1_Score(Y2_cat_pred, test_set$Y2_cat)
f1_y2
|
mcc_Y2 <- mcc(Y2_cat_pred, test_set$Y2_cat)
mcc_Y2
```

```r
set.seed(818)

library(caTools)
split = sample.split(energy$Y1_cat, SplitRatio = 0.55)

training_set = subset(energy, split == TRUE)
test_set = subset(energy, split == FALSE)

View(test_set)
```

- True positive (TP): correct positive prediction
- False positive (FP): incorrect positive prediction
- True negative (TN): correct negative prediction
- False negative (FN): incorrect negative prediction

Four outcomes of a classifier



Classification of a test dataset produces four outcomes – true positive, false positive, true negative, and false negative.

- $\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FN}+\mathrm{FP}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}}$

$$\mathrm{PREC} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

- $\mathrm{SN} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \frac{\mathrm{TP}}{\mathrm{P}}$

- $\mathrm{MCC} = \frac{\mathrm{TP}\cdot\mathrm{TN}-\mathrm{FP}\cdot\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Support Vector Machine

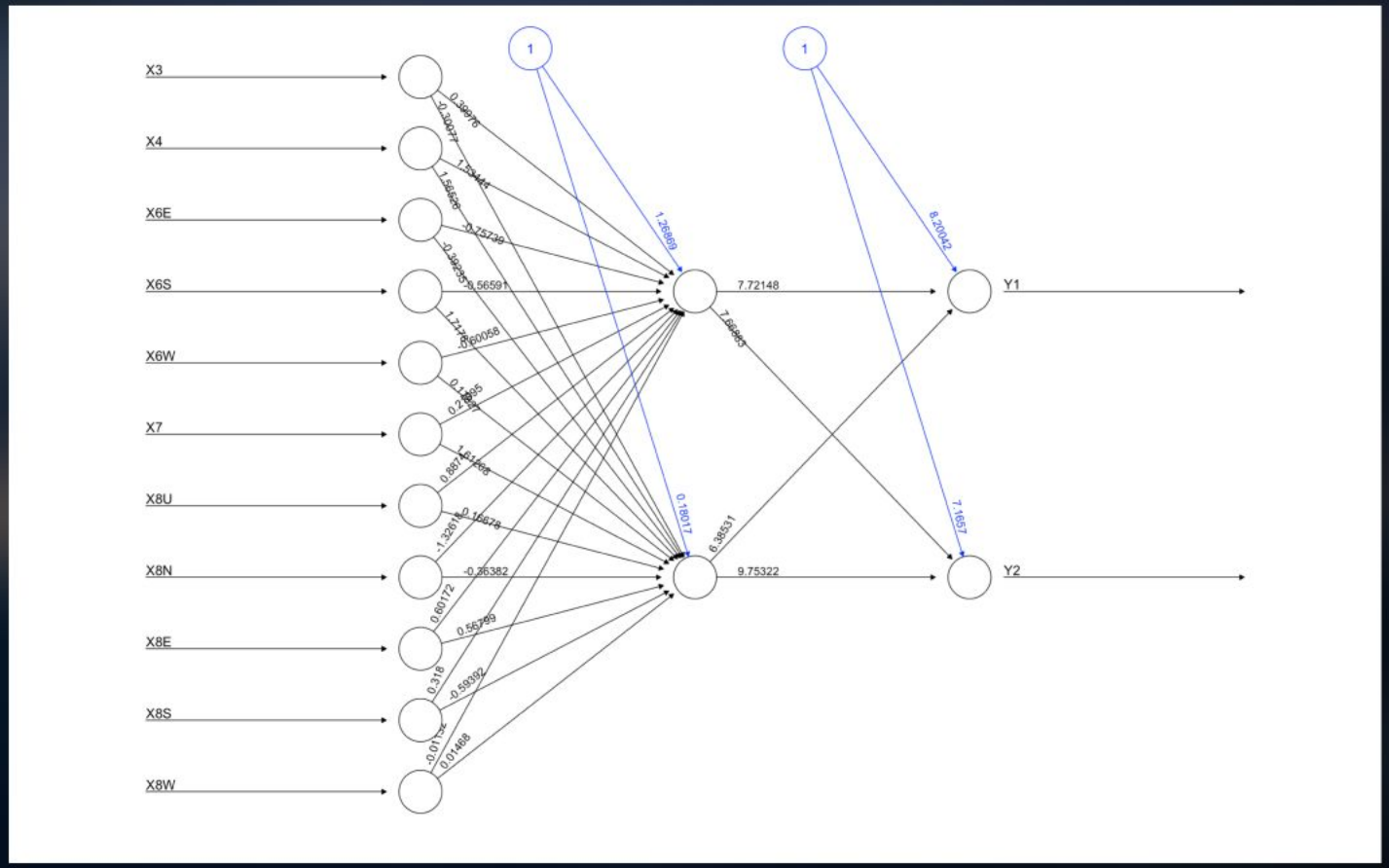
Accuracy of the models

	Y1	Y2
Accuracy	0.7762	0.7762
F1 Score	0.807229	0.785276
MCC	0.736766	0.702515

Neural Networks

We create Neural Network with multiple outputs (Cooling Load and Heating Load), using actual numeric values for loads, continuous output neural network with two outputs.

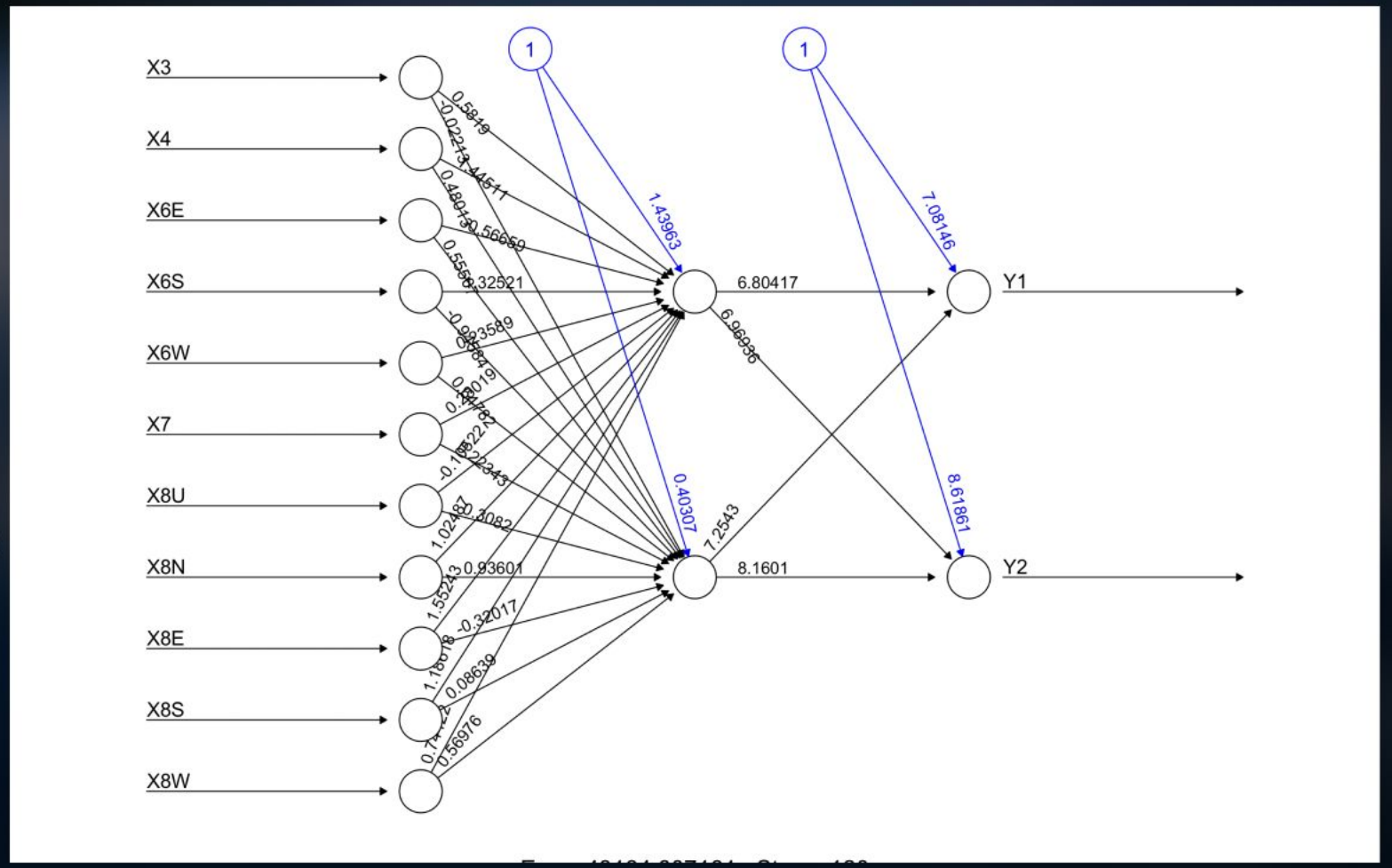
```
#Neural Networks
library(neuralnet)
energynet <- neuralnet(Y1 + Y2 ~ X3 + X4 + X6E + X6S + X6W + X7 + X8U + X8N + X8E + X8S + X8W, energyd, hidden=2,
                       lifesign="minimal", linear.output=TRUE, threshold=0.01)
plot(energynet)
trainingdata <- energyd[1:538,] #70% training
```



Neural Networks

We create Neural Network with multiple outputs (Cooling Load and Heating Load), using actual numeric values for loads, continuous output neural network with two outputs.
We use 70% training and 30% testing and using neuralnet again on trainingdata.

```
trainingdata <- energyd[1:538,] #70% training
testingdata <- energyd[539:768,] #30% testing
energynet1 <- neuralnet(Y1 + Y2 ~ X3 + X4 + X6E + X6S + X6W + X7 + X8U + X8N + X8E + X8S + X8W, trainingdata, hidden=2,
                        lifesign="minimal", linear.output=TRUE, threshold=0.01)
plot(energynet1)
```



k-Nearest Neighbor

Assigning categories for each Cooling load and Heating load (A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile and then performing k-nearest neighbor technique.

```
> quantile(main_energy$Y1)
     0%     25%     50%     75%    100%
 6.0100 12.9925 18.9500 31.6675 43.1000
> quantile(main_energy$Y2)
     0%     25%     50%     75%    100%
10.9000 15.6200 22.0800 33.1325 48.0300
```

```
quantile(main_energy$Y1)
#Categories for heating load -
#A = Highest Quantile - 75% to 100% - 31.6675 to 43.1
#B = Second highest Quantile - 50% to 75% - 18.95 to 31.6675
#C = Third highest Quantile - 25% to 50% - 12.9925 to 18.95
#D = Lowest Quantile - 0% to 25% - 6.01 to 12.9925

quantile(main_energy$Y2)
#Categories for cooling load -
#A = Highest Quantile - 75% to 100% - 33.1325 to 48.03
#B = Second highest Quantile - 50% to 75% - 22.08 to 33.1325
#C = Third highest Quantile - 25% to 50% - 15.62 to 22.08
#D = Lowest Quantile - 0% to 25% - 10.9 to 15.62
```

k-Nearest Neighbor

Assigning categories for each Cooling load and Heating load (A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile and then performing k-nearest neighbor technique.

Case 1 - Y1 (Heating Load)

For **k=30**, we get an accuracy of **65.1%.**
For **k=200**, we get an accuracy of **70.05%**
For **k=250**, we get an accuracy of **73.69%**

```
> energyeff$Y1_cat <- factor(energyeff$Y1_cat, levels= c("A","B","C","D"), labels= c("A","B","C","D"))
> normalize <- function(x) { return ((x-min(x))/(max(x)-min(x))) }
> energy_norm <- as.data.frame(lapply(energyeff[1:11], normalize))
> energy_index <- sample(nrow(energy_norm), 1/2 * nrow(energy_norm))
> energy_train <- energy_norm[energy_index, ]
> energy_test <- energy_norm[-energy_index, ]
> energy_train_labels <- energyeff[energy_index,12]
> energy_test_labels <- energyeff[-energy_index,12]
> library(class)
> energy_test_pred <- knn(train = energy_train, test = energy_test, cl=energy_train_labels, k=30)
> library(gmodels)
> CrossTable(x=energy_test_labels, y=energy_test_pred, prop.chisq=FALSE)
```

Total Observations in Table: 384

energy_test_labels	energy_test_pred A	B	C	D	Row Total
A	53	42	3	1	99
	0.535	0.424	0.030	0.010	0.258
	0.654	0.396	0.043	0.008	
	0.138	0.109	0.008	0.003	
B	28	54	10	8	100
	0.280	0.540	0.100	0.080	0.260
	0.346	0.509	0.143	0.063	
	0.073	0.141	0.026	0.021	
C	0	8	53	28	89
	0.000	0.090	0.596	0.315	0.232
	0.000	0.075	0.757	0.220	
	0.000	0.021	0.138	0.073	
D	0	2	4	90	96
	0.000	0.021	0.042	0.938	0.250
	0.000	0.019	0.057	0.709	
	0.000	0.005	0.010	0.234	
Column Total	81	106	70	127	384
	0.211	0.276	0.182	0.331	

k-Nearest Neighbor

Case 2 - Y2 (Cooling Load)

For **k=30**, we get an accuracy of **61.19%**
For **k=100**, we get an accuracy of **64.06%**
For **k=175**, we get an accuracy of **69.01%**

```
> energyeff$Y2_cat <- factor(energyeff$Y2_cat, levels= c("A","B","C","D"), labels= c("A","B","C","D"))
> normalize <- function(x) { return ((x-min(x))/(max(x)-min(x))) }
> energy_norm <- as.data.frame(lapply(energyeff[1:11], normalize))
> energy_index <- sample(nrow(energy_norm), 1/2 * nrow(energy_norm))
> energy_train <- energy_norm[energy_index, ]
> energy_test <- energy_norm[-energy_index, ]
> energy_train_labels <- energyeff[energy_index,12]
> energy_test_labels <- energyeff[-energy_index,12]
> library(class)
> energy_test_pred <- knn(train = energy_train, test = energy_test, cl=energy_train_labels, k=30)
> CrossTable(x=energy_test_labels, y=energy_test_pred, prop.chisq=FALSE)
```

Total Observations in Table: 384

energy_test_labels	energy_test_pred A	B	C	D	Row Total
A	52	36	2	3	93
	0.559	0.387	0.022	0.032	0.242
	0.658	0.383	0.024	0.024	
	0.135	0.094	0.005	0.008	
B	27	54	9	12	102
	0.265	0.529	0.088	0.118	0.266
	0.342	0.574	0.106	0.095	
	0.070	0.141	0.023	0.031	
C	0	4	54	36	94
	0.000	0.043	0.574	0.383	0.245
	0.000	0.043	0.635	0.286	
	0.000	0.010	0.141	0.094	
D	0	0	20	75	95
	0.000	0.000	0.211	0.789	0.247
	0.000	0.000	0.235	0.595	
	0.000	0.000	0.052	0.195	
Column Total	79	94	85	126	384
	0.206	0.245	0.221	0.328	

Naive Bayes

Assigning categories for each Cooling load and Heating load (A for highest quartile, B for second highest quartile, C for third highest quartile, D for lowest quartile and then performing Naive Bayes technique.

```
> quantile(main_energy$Y1)
     0%     25%     50%     75%    100% 
 6.0100 12.9925 18.9500 31.6675 43.1000 
> quantile(main_energy$Y2)
     0%     25%     50%     75%    100% 
10.9000 15.6200 22.0800 33.1325 48.0300 
```

```
quantile(main_energy$Y1)
#Categories for heating load -
#A = Highest Quantile - 75% to 100% - 31.6675 to 43.1
#B = Second highest Quantile - 50% to 75% - 18.95 to 31.6675
#C = Third highest Quantile - 25% to 50% - 12.9925 to 18.95
#D = Lowest Quantile - 0% to 25% - 6.01 to 12.9925

quantile(main_energy$Y2)
#Categories for cooling load -
#A = Highest Quantile - 75% to 100% - 33.1325 to 48.03
#B = Second highest Quantile - 50% to 75% - 22.08 to 33.1325
#C = Third highest Quantile - 25% to 50% - 15.62 to 22.08
#D = Lowest Quantile - 0% to 25% - 10.9 to 15.62
```

Naive Bayes

Case1 - For Heating Load (Y1)

1. Naive Bayes code using R on main dataset.
2. Creating prediction table and checking the accuracy of the model.

```r
#Naive Bayes
energyefficiency$Y1 <- factor(energyefficiency$Y1, levels=c("A","B","C","D"), labels= c("A","B","C","D"))
energy_index <- sample(nrow(energyefficiency), 1/2 * nrow(energyefficiency))
energy_train <- energyefficiency[energy_index, ]
energy_test <- energyefficiency[-energy_index, ]
energy_model <- naiveBayes(Y1 ~ ., data=energy_train, laplace=1)
energy_model
energy_pred <- predict(energy_model, energy_test, type="class")
energy_pred_table <- table(energy_test$Y1, energy_pred)
energy_pred_table
sum(diag(energy_pred_table))/nrow(energy_test)
```

```
> table(energyefficiency$Y1)

  A   B   C   D
192 192 192 192
> summary(energyefficiency)
       X3              X4             X6E             X6S             X6W              X7              X8U              X8N              X8E
 Min.   :245.0   Min.   :110.2   Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:294.0   1st Qu.:140.9   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.1000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :318.5   Median :183.8   Median :0.00   Median :0.00   Median :0.00   Median :0.2500   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :318.5   Mean   :176.6   Mean   :0.25   Mean   :0.25   Mean   :0.25   Mean   :0.2344   Mean   :0.1875   Mean   :0.1875   Mean   :0.1875
 3rd Qu.:343.0   3rd Qu.:220.5   3rd Qu.:0.25   3rd Qu.:0.25   3rd Qu.:0.25   3rd Qu.:0.4000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :416.5   Max.   :220.5   Max.   :1.00   Max.   :1.00   Max.   :1.00   Max.   :0.4000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
      X8S              X8W         Y1_cat
 Min.   :0.0000   Min.   :0.0000   A:192
 1st Qu.:0.0000   1st Qu.:0.0000   B:192
 Median :0.0000   Median :0.0000   C:192
 Mean   :0.1875   Mean   :0.1875   D:192
 3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000
>
```

Naive Bayes

```
Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
        A         B         C         D
0.2395833 0.2578125 0.2656250 0.2369792

Conditional probabilities:
   X3
Y       [,1]     [,2]
  A 347.2609 44.86938
  B 313.3030 23.80842
  C 323.0637 41.18875
  D 292.1154 35.82752

   X4
Y       [,1]     [,2]
  A 131.2880 13.75012
  B 139.3283 25.93225
  C 217.2574 16.43749
  D 220.5000  0.00000

   X6E
Y       [,1]      [,2]
  A 0.3043478 0.4626519
  B 0.2121212 0.4108907
  C 0.2647059 0.4433551
  D 0.1978022 0.4005491

   X6S
Y       [,1]      [,2]
  A 0.2282609 0.4220114
  B 0.2525253 0.4366719
  C 0.2254902 0.4199685
  D 0.2747253 0.4488488
```

```
   X6W
Y       [,1]      [,2]
  A 0.2391304 0.4288898
  B 0.2727273 0.4476283
  C 0.2647059 0.4433551
  D 0.2967033 0.4593354

   X7
Y       [,1]       [,2]
  A 0.2923913 0.12042388
  B 0.1878788 0.12496753
  C 0.3166667 0.11843311
  D 0.1461538 0.09580964

   X8U
Y       [,1]      [,2]
  A 0.1847826 0.3902478
  B 0.1919192 0.3958140
  C 0.2352941 0.4262772
  D 0.1538462 0.3628001

   X8N
Y       [,1]      [,2]
  A 0.2282609 0.4220114
  B 0.2121212 0.4108907
  C 0.2058824 0.4063417
  D 0.1868132 0.3919209

   X8E
Y       [,1]      [,2]
  A 0.1956522 0.3988756
  B 0.1515152 0.3603750
  C 0.2058824 0.4063417
  D 0.1978022 0.4005491

   X8S
Y       [,1]      [,2]
  A 0.2173913 0.4147311
  B 0.1717172 0.3790537
  C 0.1372549 0.3458156
  D 0.1318681 0.3402219
```

```
   X8W
Y       [,1]      [,2]
  A 0.1739130 0.3811116
  B 0.1818182 0.3876574
  C 0.1764706 0.3831026
  D 0.1978022 0.4005491

   Y1_cat
Y             A           B           C           D
  A 0.968750000 0.010416667 0.010416667 0.010416667
  B 0.009708738 0.970873786 0.009708738 0.009708738
  C 0.009433962 0.009433962 0.971698113 0.009433962
  D 0.010526316 0.010526316 0.010526316 0.968421053

> energy_pred <- predict(energy_model, energy_test, type="class")
> energy_pred_table <- table(energy_test$Y1, energy_pred)
> energy_pred_table
   energy_pred
     A   B   C   D
  A 100   0   0   0
  B   1  89   0   3
  C   0   5   1  84
  D   0   0   0 101
> sum(diag(energy_pred_table))/nrow(energy_test)
[1] 0.7578125
```

Naive Bayes

Case2 - For Cooling Load (Y2)

1. Naive Bayes code using R on main dataset.
2. Creating prediction table and checking the accuracy of the model.

```
table(energyefficiency$Y2)
summary(energyefficiency)
energyefficiency$Y2 <- factor(energyefficiency$Y2, levels=c("A","B","C","D"), labels= c("A","B","C","D"))
energy_index <- sample(nrow(energyefficiency), 1/2 * nrow(energyefficiency))
energy_train <- energyefficiency[energy_index, ]
energy_test <- energyefficiency[-energy_index, ]
energy_model <- naiveBayes(Y2 ~ ., data=energy_train, laplace=1)
energy_model
energy_pred <- predict(energy_model, energy_test, type="class")
energy_pred_table <- table(energy_test$Y2, energy_pred)
energy_pred_table
sum(diag(energy_pred_table))/nrow(energy_test)
```

```
> energy_pred <- predict(energy_model, energy_test, type="class")
> energy_pred_table <- table(energy_test$Y2, energy_pred)
> energy_pred_table
   energy_pred
     A   B   C   D
  A 98   0   0   0
  B  1  92   0   4
  C  0   3   0  86
  D  0   0   0 100
> sum(diag(energy_pred_table))/nrow(energy_test)
[1] 0.7552083
>
```

```
> table(energyefficiency$Y2)

  A   B   C   D
192 192 192 192
> summary(energyefficiency)
       X3              X4             X6E             X6S             X6W              X7              X8U              X8N              X8E
 Min.   :245.0   Min.   :110.2   Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:294.0   1st Qu.:140.9   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.1000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :318.5   Median :183.8   Median :0.00   Median :0.00   Median :0.00   Median :0.2500   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :318.5   Mean   :176.6   Mean   :0.25   Mean   :0.25   Mean   :0.25   Mean   :0.2344   Mean   :0.1875   Mean   :0.1875   Mean   :0.1875
 3rd Qu.:343.0   3rd Qu.:220.5   3rd Qu.:0.25   3rd Qu.:0.25   3rd Qu.:0.25   3rd Qu.:0.4000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :416.5   Max.   :220.5   Max.   :1.00   Max.   :1.00   Max.   :1.00   Max.   :0.4000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
      X8S             X8W         Y2_cat
 Min.   :0.0000   Min.   :0.0000   A:192
 1st Qu.:0.0000   1st Qu.:0.0000   B:192
 Median :0.0000   Median :0.0000   C:192
 Mean   :0.1875   Mean   :0.1875   D:192
 3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000
>
```

Decision Trees

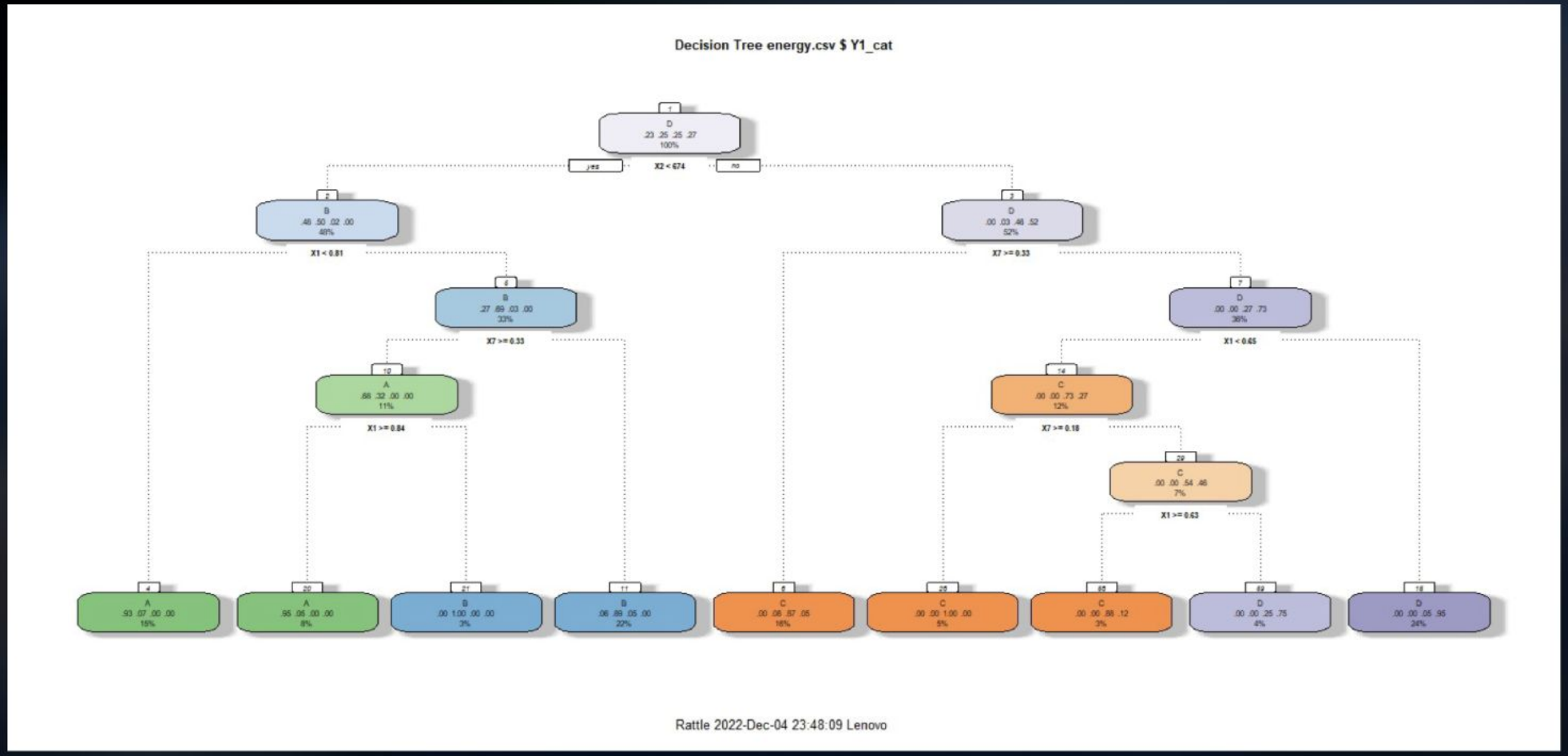
A decision tree evaluates the explanatory (X) variables in a dataset and determines the most important variables in making a decision. Using actual numerical values for loads.

Steps -

1. Using R version 4.1.0 with rattle( ) function we get the above descriptive summary. Rattle selects 70% of the data for a training dataset; 30% for testing.
2. Building & drawing decision trees.
3. Testing the decision tree model & plotting the error matrix.

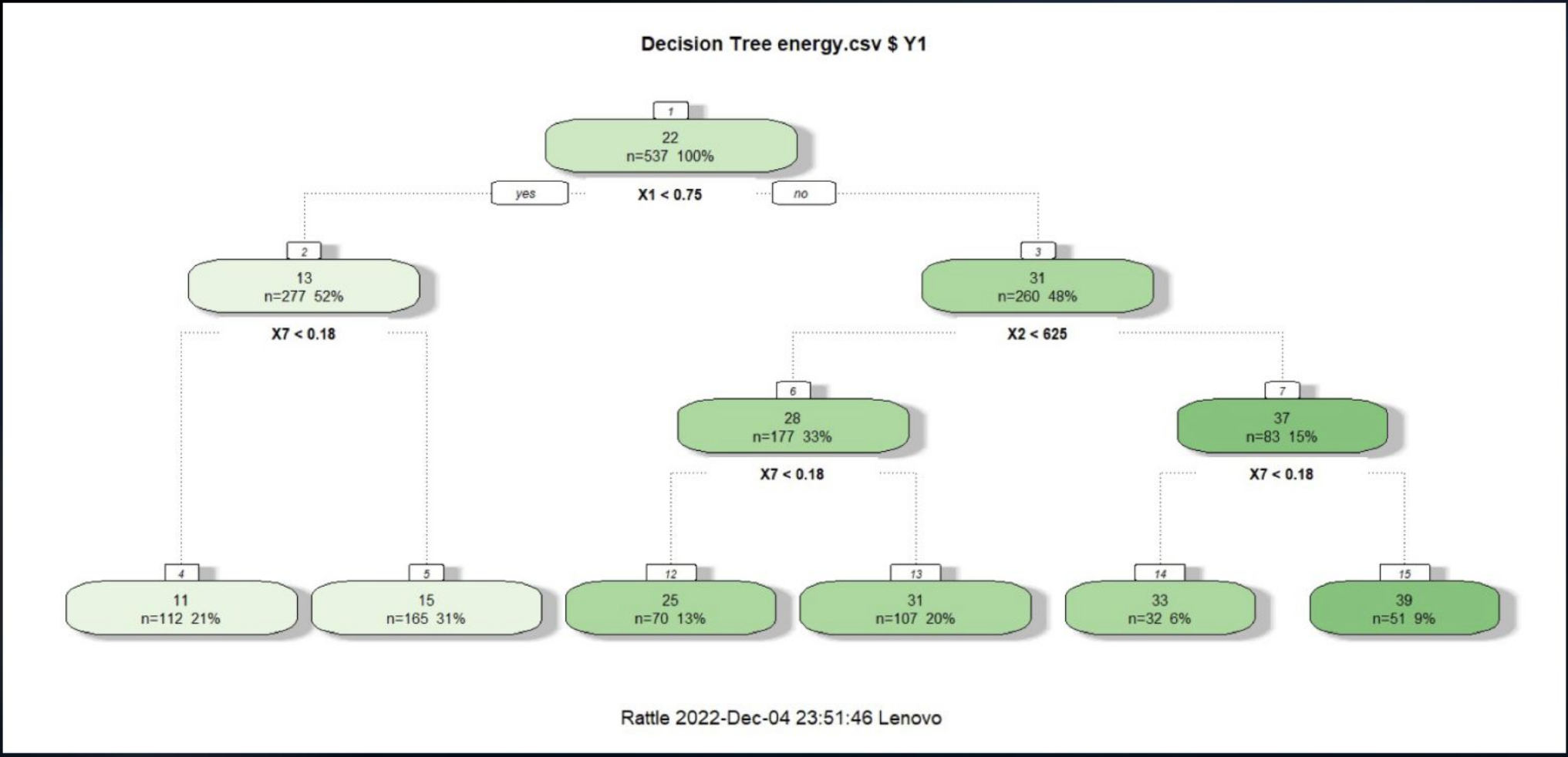
Decision Trees

Decision Tree of categorical Y1 (Heating Load)



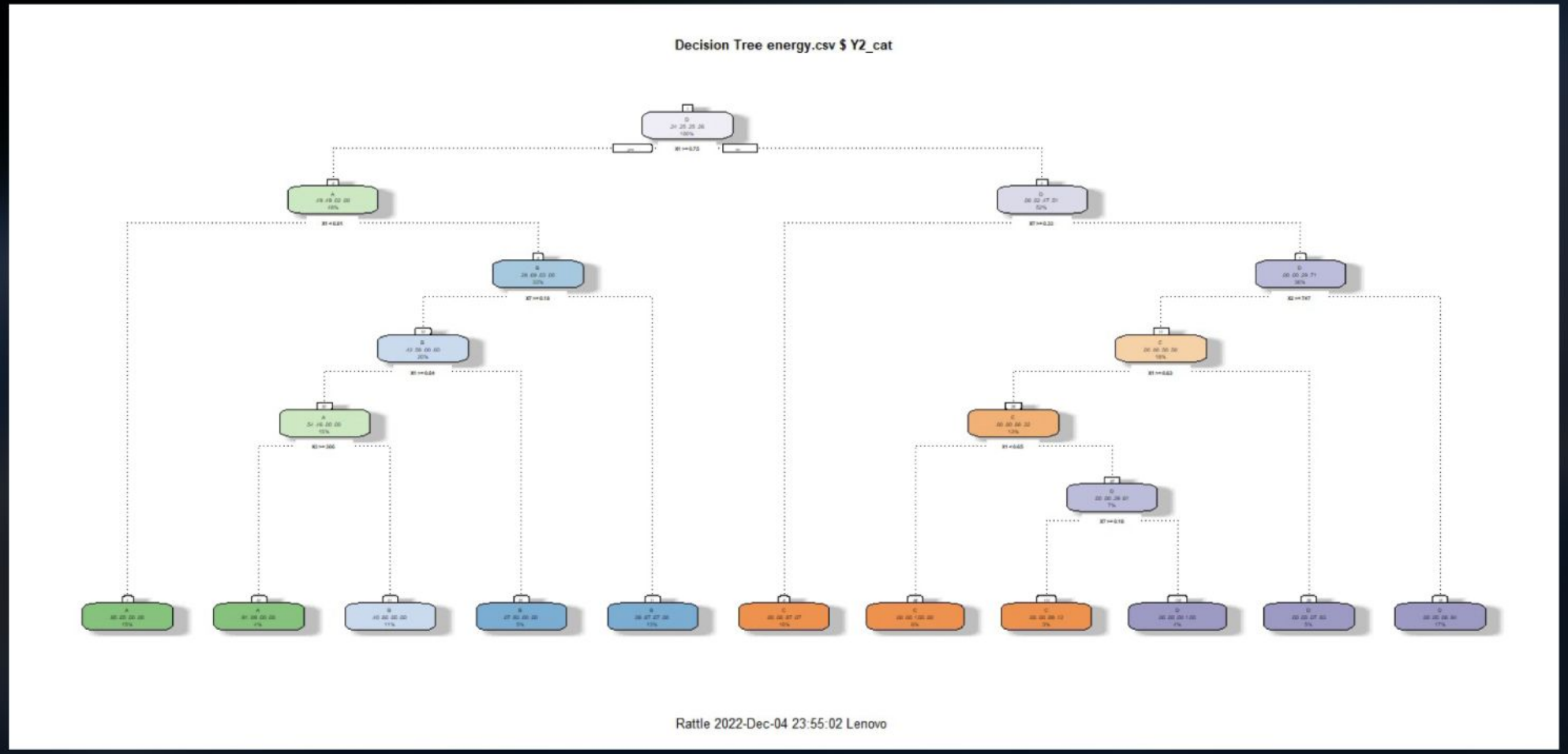
Decision Trees

Decision Tree of numerical Y1 (Heating Load)



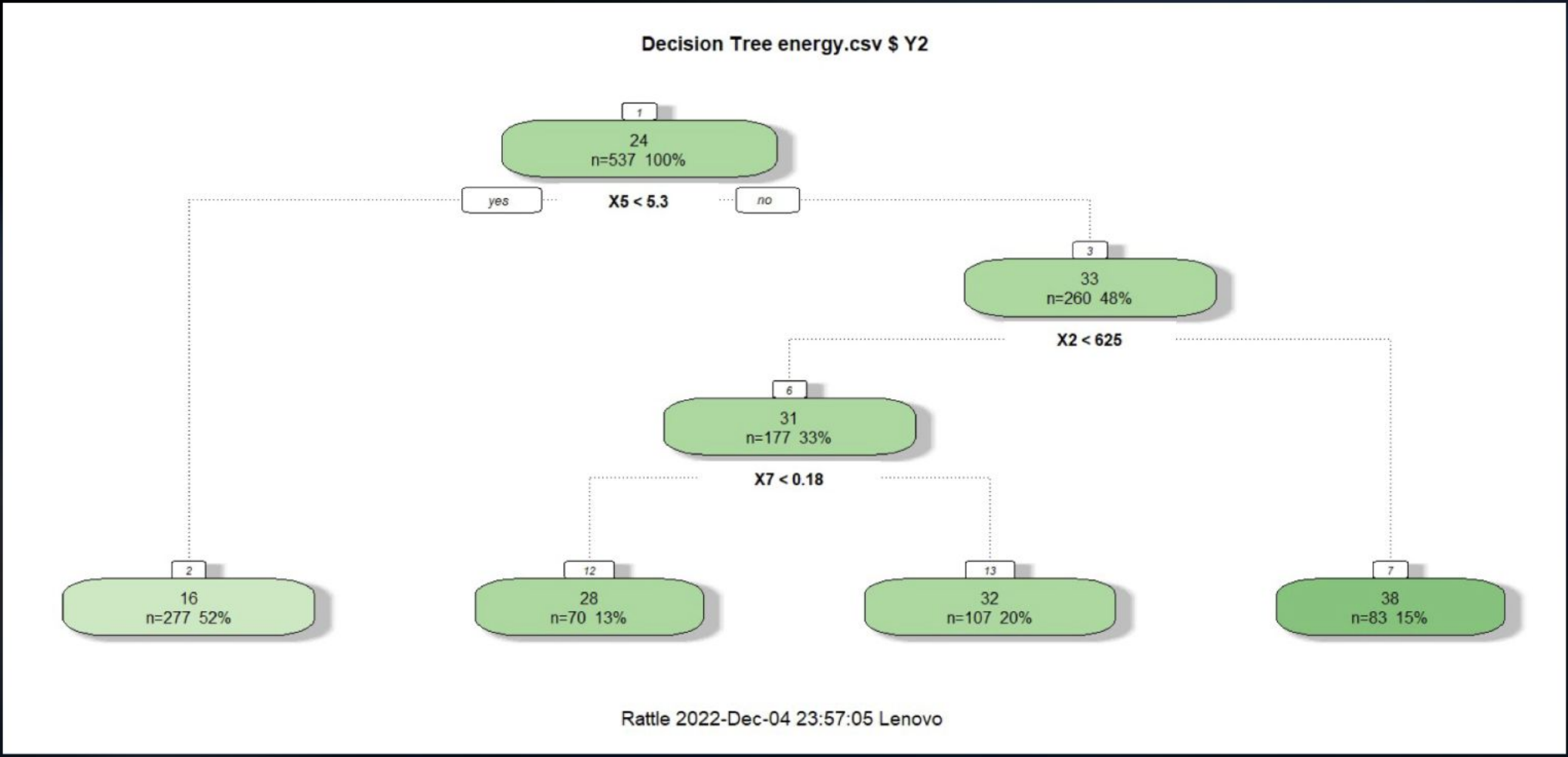
Decision Trees

Decision Tree of categorical Y2 (Cooling Load)



Decision Trees

Decision Tree of numerical Y2 (Cooling Load)



Random Forests

Random Forest is a powerful and versatile supervised machine learning algorithm that grows and combines multiple decision trees to create a “forest.” The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone.

Steps -

1. Using rattle( ) function executing random forests with 500 forests and 2 variables.
2. Plotting ‘mean decrease accuracy’ using rattle ( ).

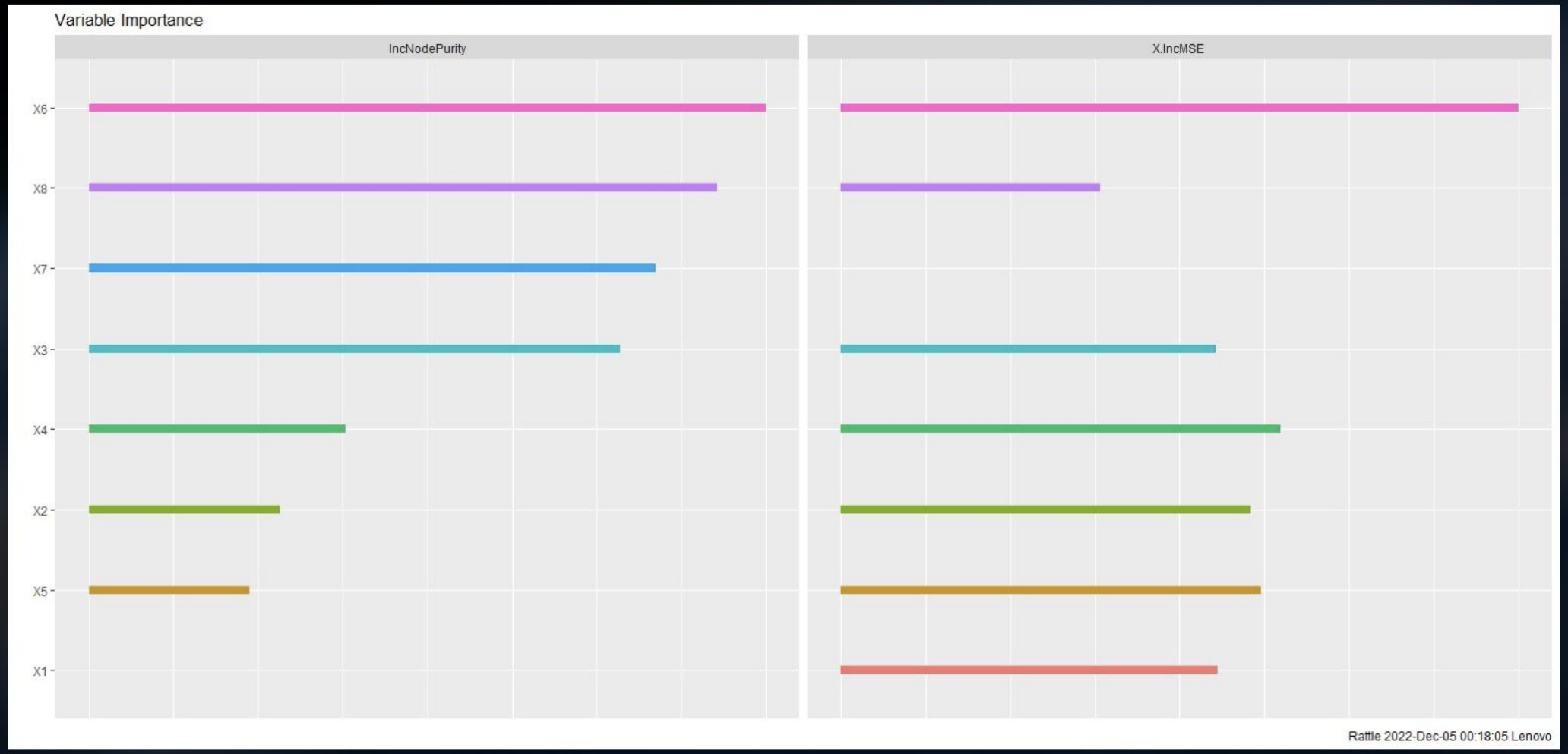
Random Forests

For continuous variable of Y1(Heating Load)

```
Number of observations used to build the model: 537
Missing value imputation is active.

Call:
 randomForest(formula = Y1 ~ .,
              data = crs$dataset[crs$train, c(crs$input, crs$target)],
              ntree = 500, mtry = 2, importance = TRUE, replace = FALSE, na.action = randomForest::na.roughfix)

               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 1.264581
                    % Var explained: 98.73

Variable Importance
===================

   %IncMSE IncNodePurity
X7   50.65       1532.30
X8   27.70        690.65
X3   17.52       2010.85
X1   17.36       9231.60
X2   14.43       6625.27
X5   13.50       7044.36
X4   11.83       5745.61
X6   -9.22         31.65

Time taken: 0.35 secs

Rattle timestamp: 2022-12-05 00:33:07 Lenovo
=====================================================================
```

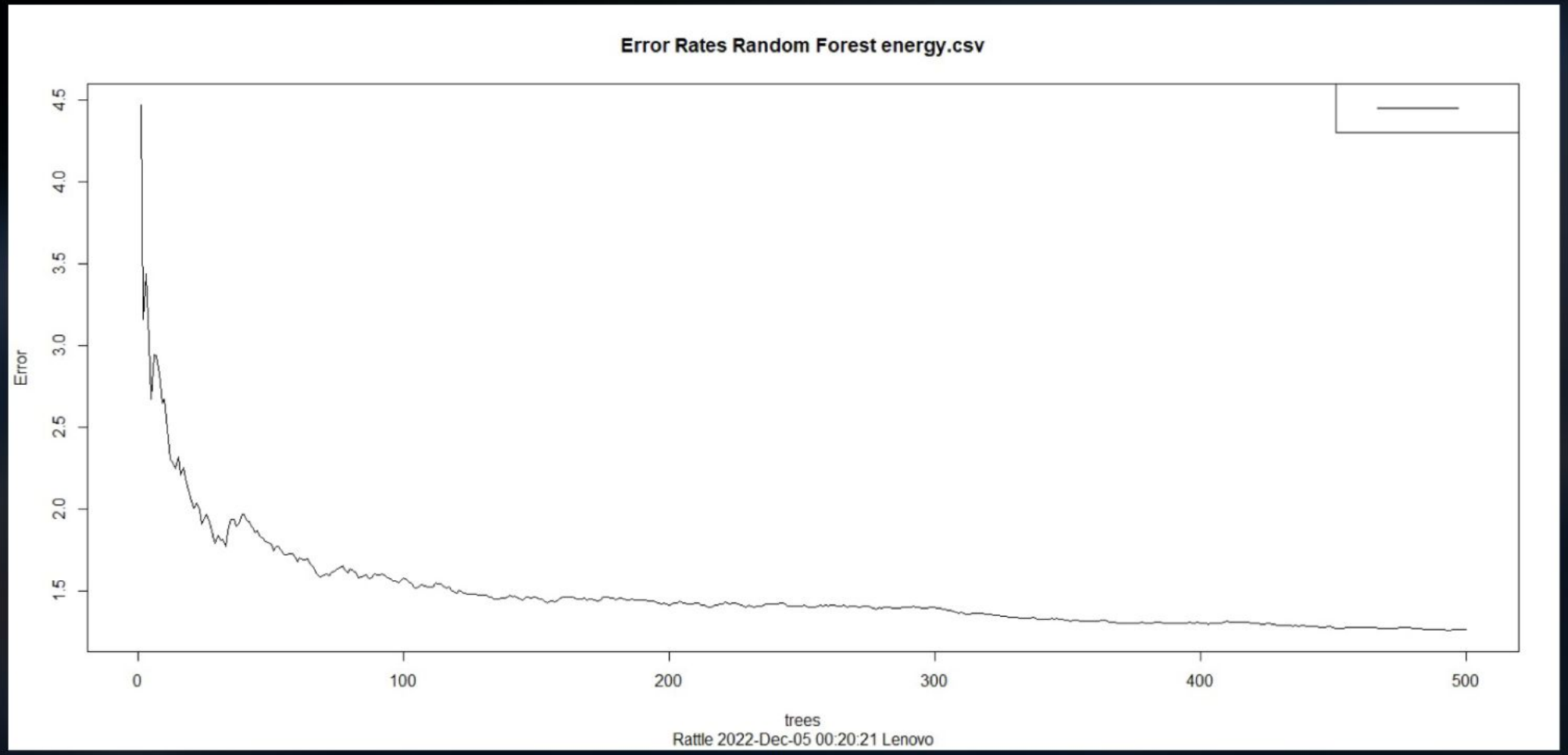
Random Forests

Mean Decrease Accuracy for Y1(Heating Load)



Random Forests

Random Forests Error for Y1(Heating Load)



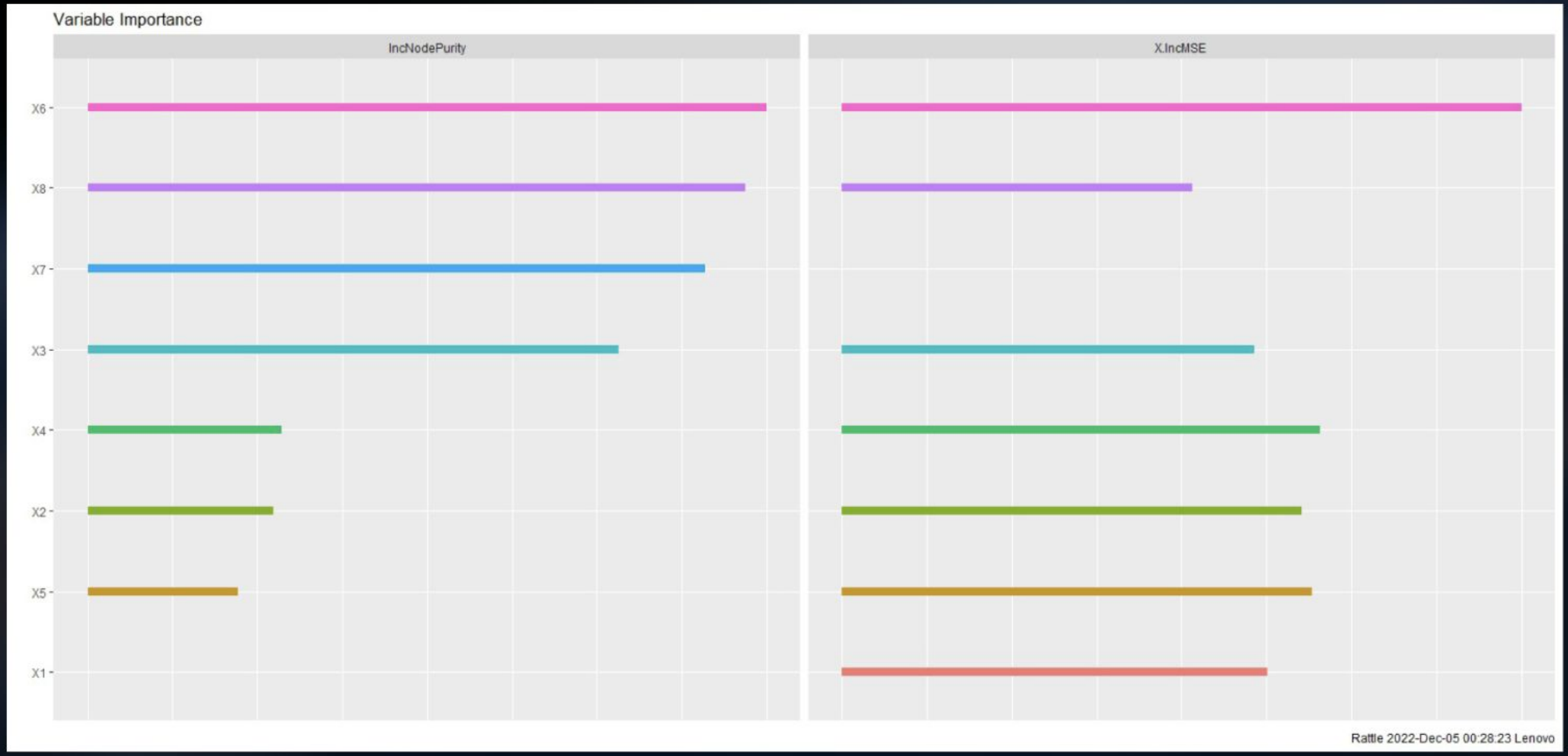
Random Forests

For continuous variable of Y2(Cooling Load)

```
Number of observations used to build the model: 537
Missing value imputation is active.

Call:
 randomForest(formula = Y2 ~ .,
              data = crs$dataset[crs$train, c(crs$input, crs$target)],
              ntree = 500, mtry = 2, importance = TRUE, replace = FALSE, na.action = randomForest::na.roughfix)

               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 3.526295
                    % Var explained: 96.07

Variable Importance
===================

   %IncMSE IncNodePurity
X7   49.45        810.20
X8   22.64        326.74
X3   17.88       1831.17
X1   16.88       8135.69
X2   14.25       5932.33
X5   13.43       6348.34
X4   12.86       5829.21
X6   -2.58         74.94

Time taken: 0.28 secs

Rattle timestamp: 2022-12-05 00:28:22 Lenovo
=====================================================================
```

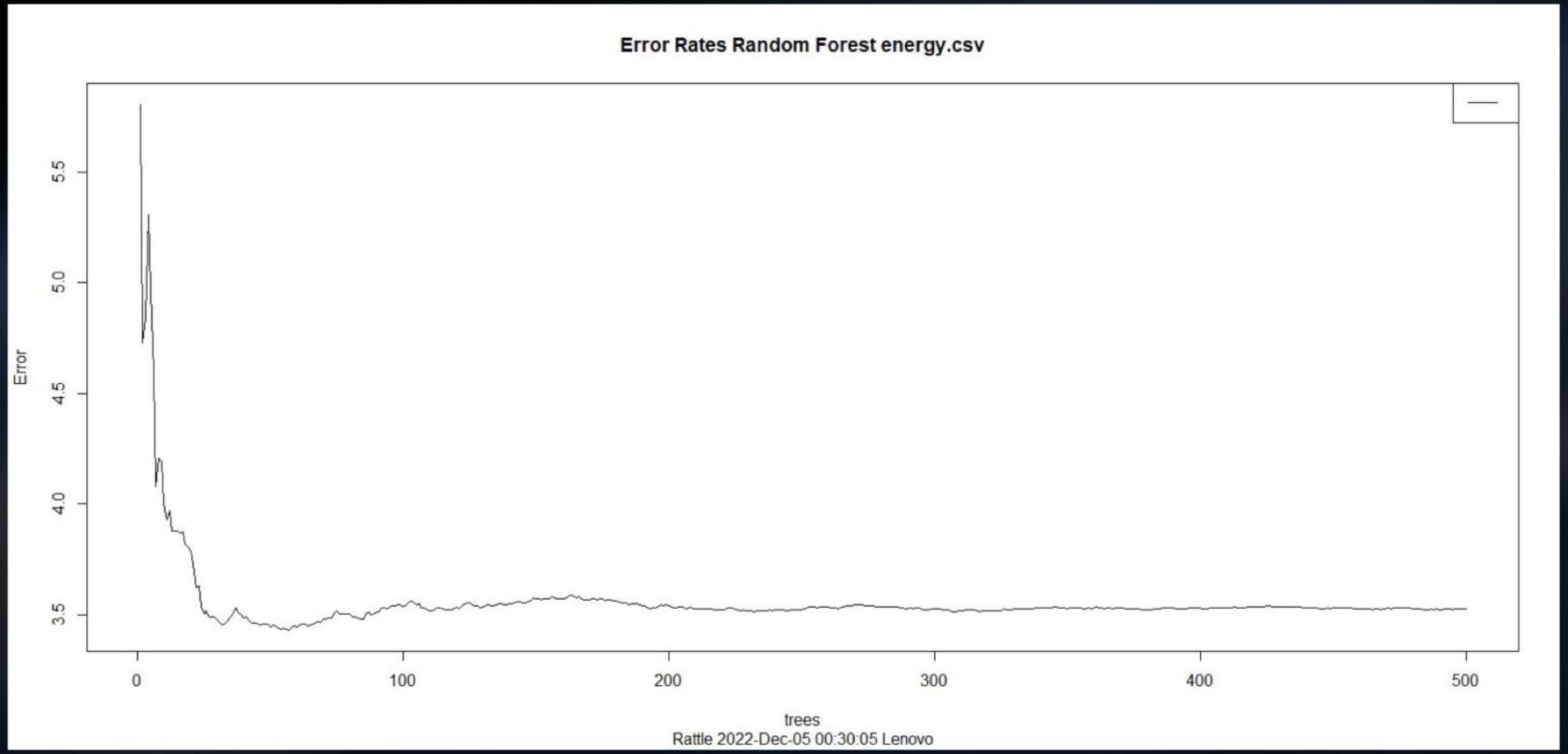
Random Forests

Mean Decrease Accuracy for Y2(Cooling Load)



Random Forests

Random Forests Error for Y2(Cooling Load)



Insights

All the Machine Learning techniques that were used for analysis worked well.

- VIF Analysis was crucial to determine variables with high multicollinearity, where we dropped some variables for some ML models (except for Decision Trees and Random Forests, as they are not affected by multicollinearity)
- When we consider variables with high multicollinearity, we observe that Heating Load depends on variables **X1 (Compactness) less than 0.75, X7 (Glazing Area) between 0% and 10% floor area and X2 (Surface Area)** the most and Cooling Load depends on variables **X5 (Overall Height), X2 (Surface Area) and X7** (**Glazing Area) between 0% and 10% floor area** the most.
- When we consider variables with low multicollinearity, we observe that Heating Load depends on variables **X3 (Wall Area), X4 (Roof Area), X7 (Glazing Area) and X8 (Glazing Area Distribution)** whereas Cooling Load depends on variables **X3 (Wall Area), X4 (Roof Area) and X7 (Glazing Area)** only. For both **Orientation** does not matter.

Insights

Accuracy for Machine Learning techniques used -

ML TECHNIQUE	HEATING LOAD	COOLING LOAD
Linear Regression -		
• All variables	• **R-squared=91.56%**	• **R-squared=88.68%**
• Dropping X1,X2,X5	• **R-squared=86.79%**	• **R-squared=82.08%**
Perceptrons	**12 Incorrect**	**12 Incorrect**
SVM	**X3 and X4**	**X3 and X4**
kNN	**k=250; Accuracy=73.69%**	**k=175; Accuracy=69.01%**
Naive Bayes	**Accuracy=75.78%**	**Accuracy=75.52%**
Decision Trees	**Decision=X1,X7,X2**	**Decision=X5,X7,X2**
Random Forests	**Accuracy=X6,X8,X7**	**Accuracy=X6,X8,X7**

THANK YOU!

Group 2 -

Akshata Ravinder
Nehal Taya
Akash Kumar Singh
Elmir Guluyev