DA5030.Proj.Parpattedar

Shruti Parpattedar

April 21, 2019

Data Acquisition

X1 Relative Compactness X2 Surface Area - m2 X3 Wall Area - m2 X4 Roof Area - m2 X5 Overall Height - m X6 Orientation - 2:North, 3:East, 4:South, 5:West X7 Glazing Area - 0%, 10%, 25%, 40% (of floor area) X8 Glazing Area Distribution - 1:Uniform, 2:North, 3:East, 4:South, 5:West y1 Heating Load - kWh/m² y2 Cooling Load - kWh/m²

Loading data from an excel sheet. Considering all features as continuous.

```r
library(readxl)
proj_data <- data.frame(read_excel("ENB2012_data.xlsx"))
colnames(proj_data) <- c("Rel_Com", "Surf_Area", "Wall_Area",
                         "Roof_Area", "Ov_Hght", "Orient",
                         "Glaz_Area", "Glaz_A_Dist", "Heating", "Cooling")

str(proj_data)
```

```
## 'data.frame':    768 obs. of  10 variables:
##  $ Rel_Com    : num  0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...
##  $ Surf_Area  : num  514 514 514 514 564 ...
##  $ Wall_Area  : num  294 294 294 294 318 ...
##  $ Roof_Area  : num  110 110 110 110 122 ...
##  $ Ov_Hght    : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ Orient     : num  2 3 4 5 2 3 4 5 2 3 ...
##  $ Glaz_Area  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Glaz_A_Dist: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Heating    : num  15.6 15.6 15.6 15.6 20.8 ...
##  $ Cooling    : num  21.3 21.3 21.3 21.3 28.3 ...
```

```r
summary(proj_data)
```

```
##     Rel_Com         Surf_Area       Wall_Area       Roof_Area    
##  Min.   :0.6200   Min.   :514.5   Min.   :245.0   Min.   :110.2  
##  1st Qu.:0.6825   1st Qu.:606.4   1st Qu.:294.0   1st Qu.:140.9  
##  Median :0.7500   Median :673.8   Median :318.5   Median :183.8  
##  Mean   :0.7642   Mean   :671.7   Mean   :318.5   Mean   :176.6  
##  3rd Qu.:0.8300   3rd Qu.:741.1   3rd Qu.:343.0   3rd Qu.:220.5  
##  Max.   :0.9800   Max.   :808.5   Max.   :416.5   Max.   :220.5  
##     Ov_Hght         Orient        Glaz_Area       Glaz_A_Dist   
##  Min.   :3.50   Min.   :2.00   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:3.50   1st Qu.:2.75   1st Qu.:0.1000   1st Qu.:1.750  
##  Median :5.25   Median :3.50   Median :0.2500   Median :3.000  
##  Mean   :5.25   Mean   :3.50   Mean   :0.2344   Mean   :2.812  
##  3rd Qu.:7.00   3rd Qu.:4.25   3rd Qu.:0.4000   3rd Qu.:4.000  
##  Max.   :7.00   Max.   :5.00   Max.   :0.4000   Max.   :5.000  
##     Heating          Cooling     
##  Min.   : 6.01   Min.   :10.90  
##  1st Qu.:12.99   1st Qu.:15.62  
##  Median :18.95   Median :22.08  
```

```
##  Mean   :22.31   Mean   :24.59
##  3rd Qu.:31.67   3rd Qu.:33.13
##  Max.   :43.10   Max.   :48.03
```

Data Exploration

Exploratory data plots

Displaying histogram for various features.

Observations from ggplot - - Roof and surface area range is high when the overall height is low, - Roof and surface area range is low when the overall height is high - There are no observations for high overall height and high surface area and also for low overall height and low surface area.

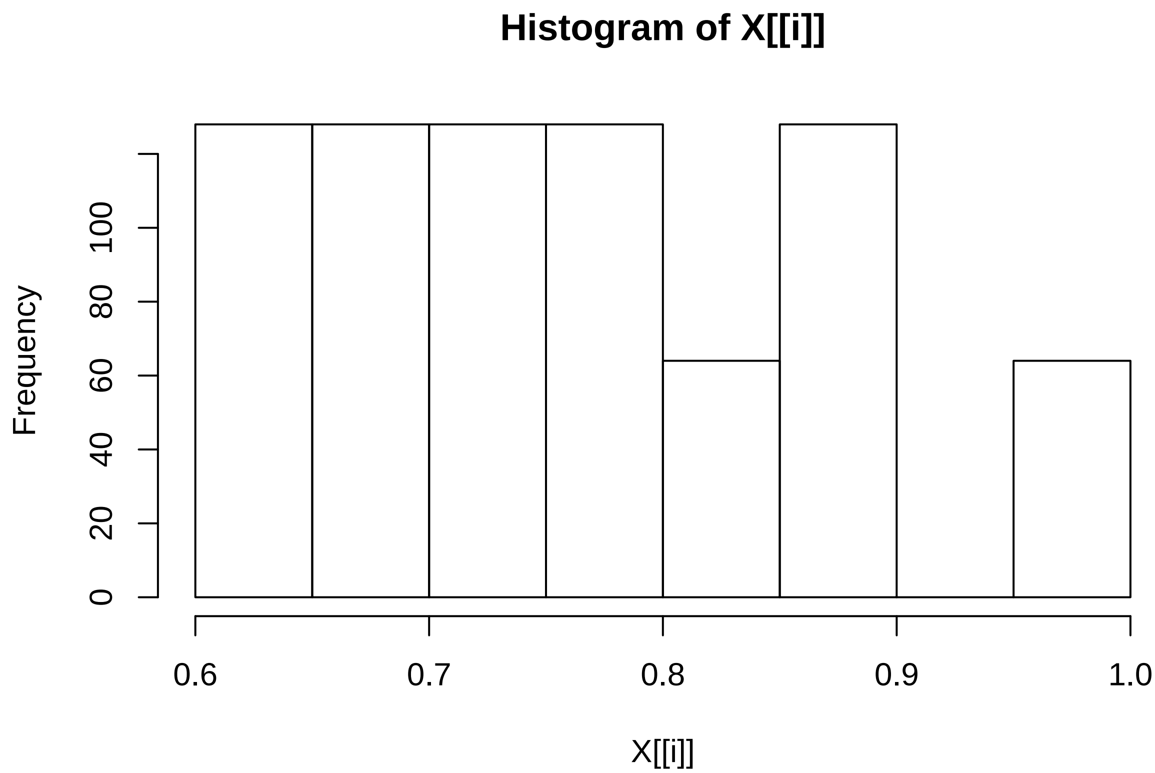
Outlier Detection

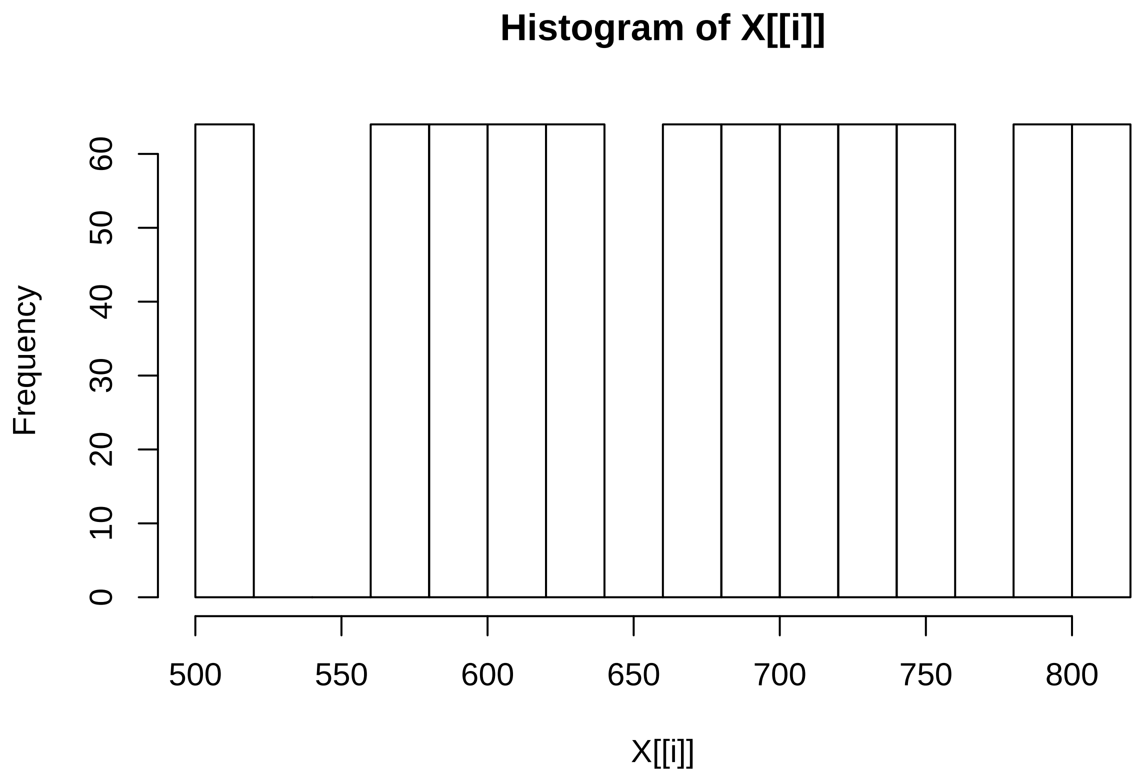
Considering 3 std dev from mean to be an outlier. Using this, no outliers were found.

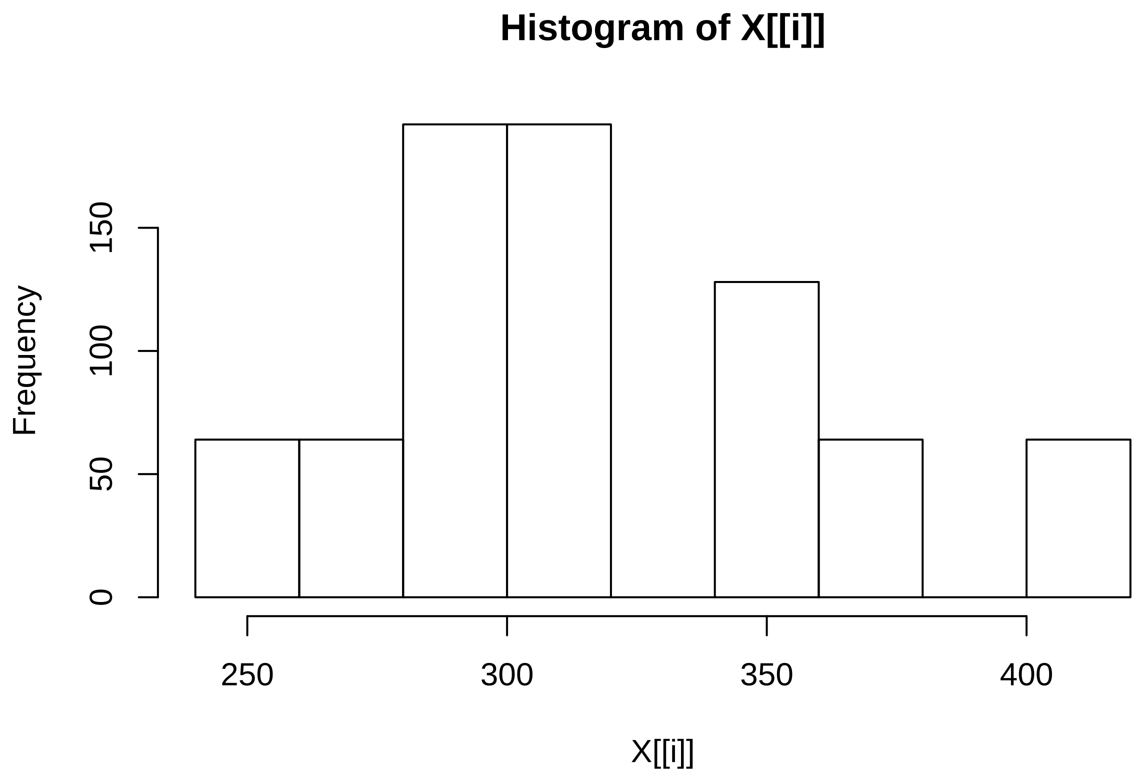
```r
library(psych)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

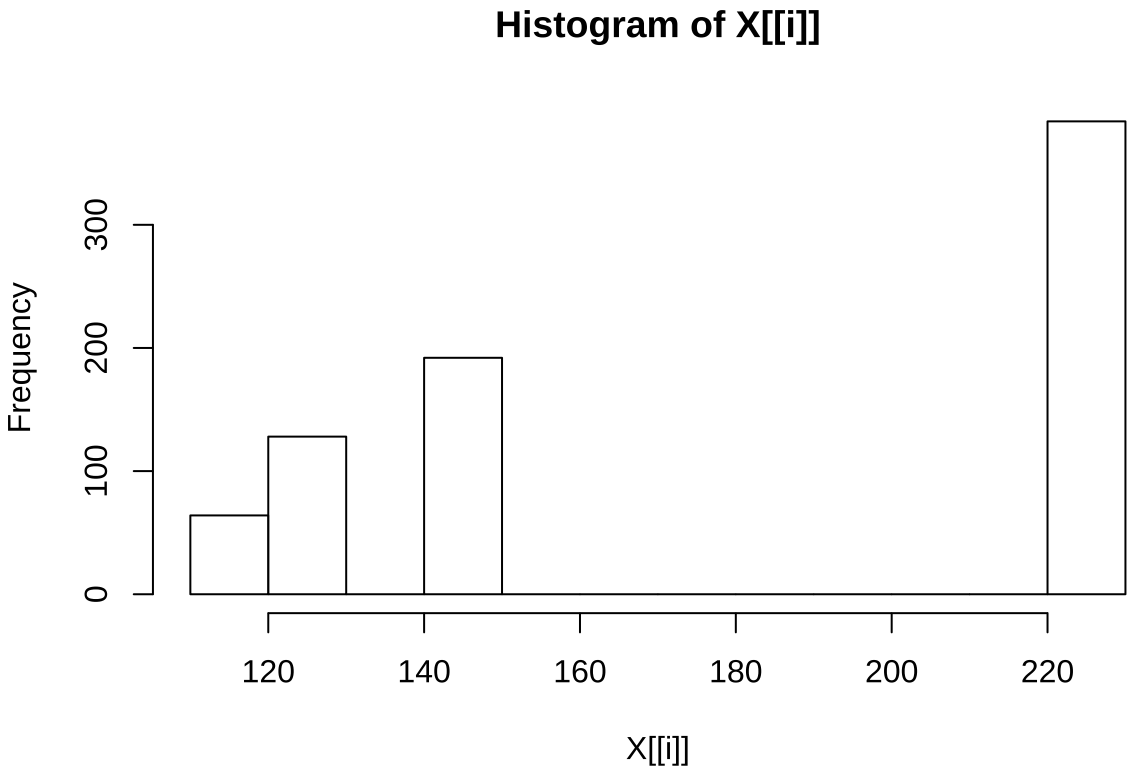
```r
# Displaying histograms for all attributes of the dataset
lapply(proj_data[,c(1:5, 9, 10)], hist)
```

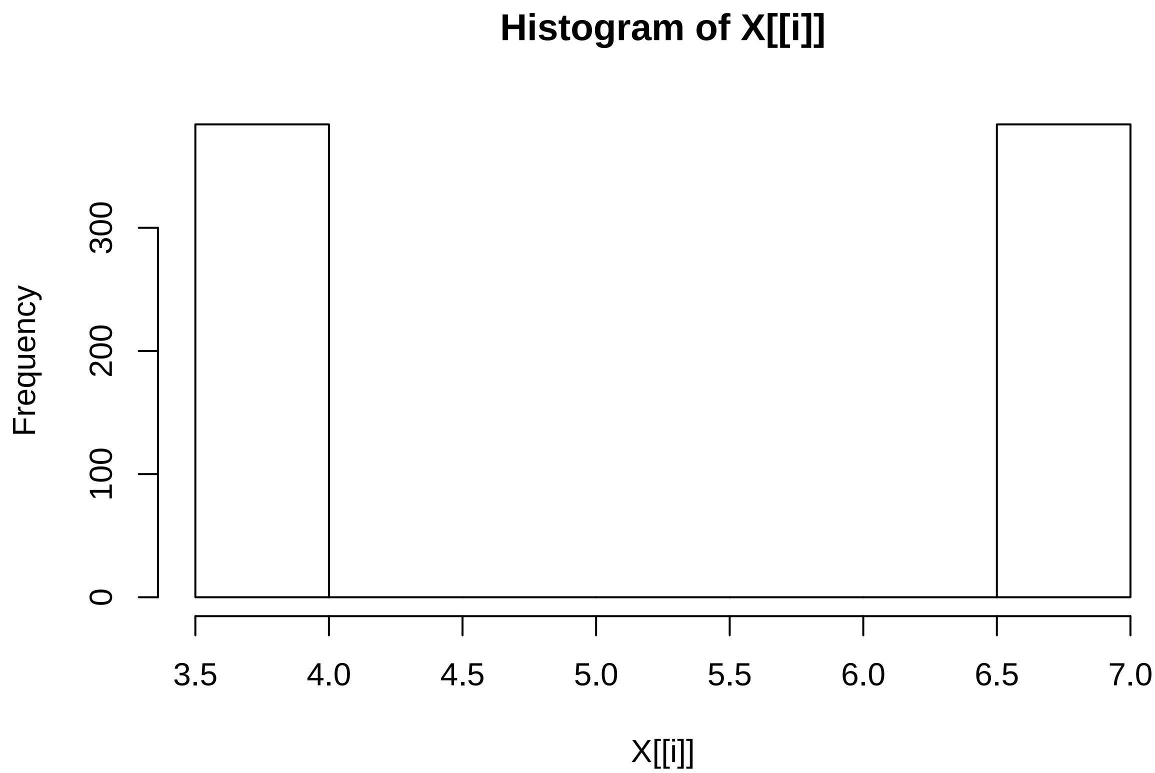

Histogram of X[[i]]
Frequency
0
20
40
60
80
100
0.6
0.7
0.8
0.9
1.0
X[[i]]


Histogram of X[[i]]
Frequency
0
10
20
30
40
50
60
500
550
600
650
700
750
800
X[[i]]

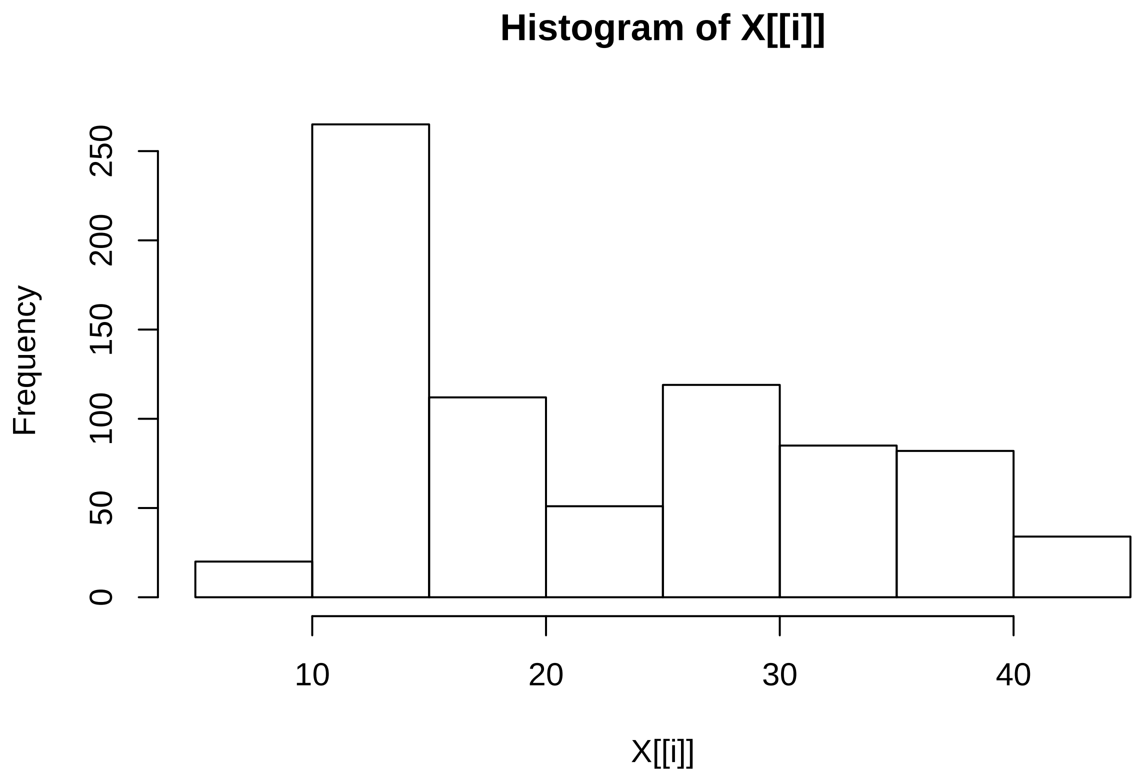


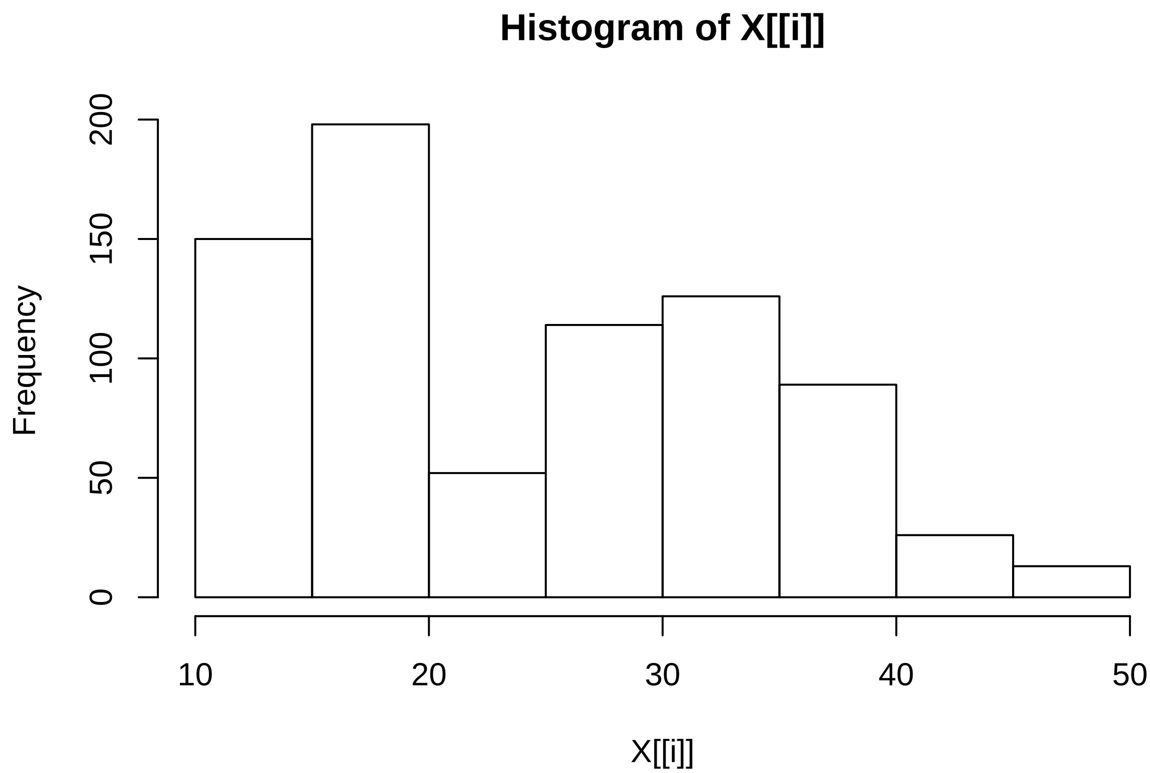
Histogram of X[[i]]
Frequency
0
50
100
150
250
300
350
400
X[[i]]


Histogram of X[[i]]
Frequency
0
100
200
300
120
140
160
180
200
220
X[[i]]



Histogram of X[[i]]
Frequency
0
100
200
300
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
X[[i]]


Histogram of X[[i]]
Frequency
0
50
100
150
200
250
10
20
30
40
X[[i]]



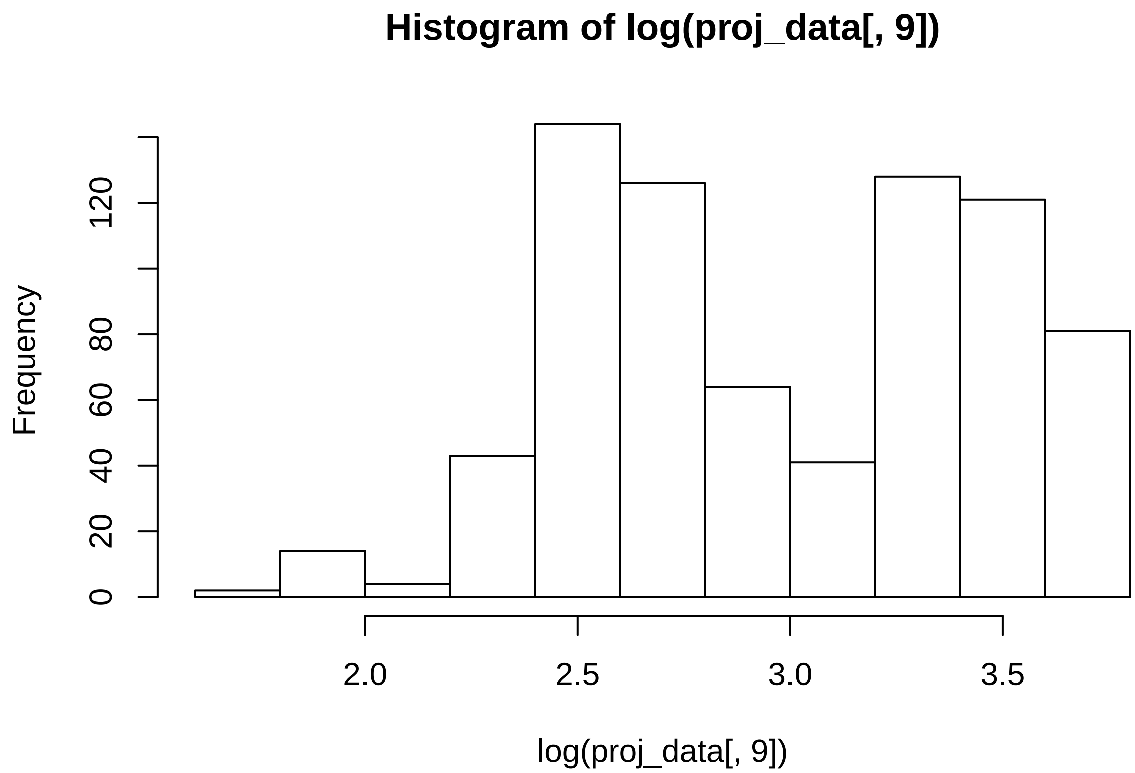

```
## $Rel_Com
## $breaks
## [1] 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00
##
## $counts
## [1] 128 128 128 128  64 128   0  64
##
## $density
## [1] 3.333333 3.333333 3.333333 3.333333 1.666667 3.333333 0.000000 1.666667
##
## $mids
## [1] 0.625 0.675 0.725 0.775 0.825 0.875 0.925 0.975
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Surf_Area
## $breaks
##  [1] 500 520 540 560 580 600 620 640 660 680 700 720 740 760 780 800 820
##
```

```
## $counts
##  [1] 64  0  0 64 64 64 64  0 64 64 64 64 64  0 64 64
##
## $density
##  [1] 0.004166667 0.000000000 0.000000000 0.004166667 0.004166667
##  [6] 0.004166667 0.004166667 0.000000000 0.004166667 0.004166667
## [11] 0.004166667 0.004166667 0.004166667 0.000000000 0.004166667
## [16] 0.004166667
##
## $mids
##  [1] 510 530 550 570 590 610 630 650 670 690 710 730 750 770 790 810
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Wall_Area
## $breaks
##  [1] 240 260 280 300 320 340 360 380 400 420
##
## $counts
## [1]  64  64 192 192   0 128  64   0  64
##
## $density
## [1] 0.004166667 0.004166667 0.012500000 0.012500000 0.000000000 0.008333333
## [7] 0.004166667 0.000000000 0.004166667
##
## $mids
## [1] 250 270 290 310 330 350 370 390 410
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Roof_Area
## $breaks
##  [1] 110 120 130 140 150 160 170 180 190 200 210 220 230
##
## $counts
##  [1]  64 128   0 192   0   0   0   0   0   0   0 384
##
## $density
##  [1] 0.008333333 0.016666667 0.000000000 0.025000000 0.000000000
##  [6] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
```

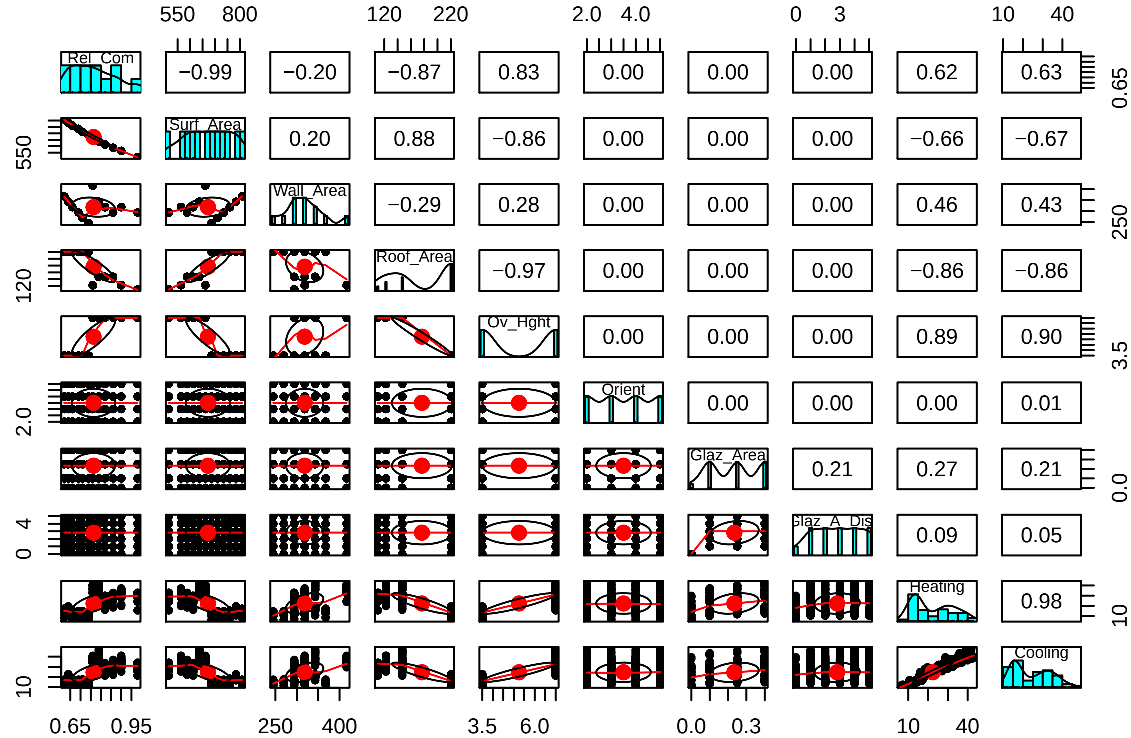
```
## [11] 0.000000000 0.050000000
##
## $mids
##  [1] 115 125 135 145 155 165 175 185 195 205 215 225
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Ov_Hght
## $breaks
## [1] 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
##
## $counts
## [1] 384   0   0   0   0   0 384
##
## $density
## [1] 1 0 0 0 0 0 1
##
## $mids
## [1] 3.75 4.25 4.75 5.25 5.75 6.25 6.75
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Heating
## $breaks
## [1]  5 10 15 20 25 30 35 40 45
##
## $counts
## [1]  20 265 112  51 119  85  82  34
##
## $density
## [1] 0.005208333 0.069010417 0.029166667 0.013281250 0.030989583 0.022135417
## [7] 0.021354167 0.008854167
##
## $mids
## [1]  7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5
##
## $xname
## [1] "X[[i]]"
##
## $equidist
```

```
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Cooling
## $breaks
## [1] 10 15 20 25 30 35 40 45 50
##
## $counts
## [1] 150 198  52 114 126  89  26  13
##
## $density
## [1] 0.039062500 0.051562500 0.013541667 0.029687500 0.032812500 0.023177083
## [7] 0.006770833 0.003385417
##
## $mids
## [1] 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5
##
## $xname
## [1] "X[[i]]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

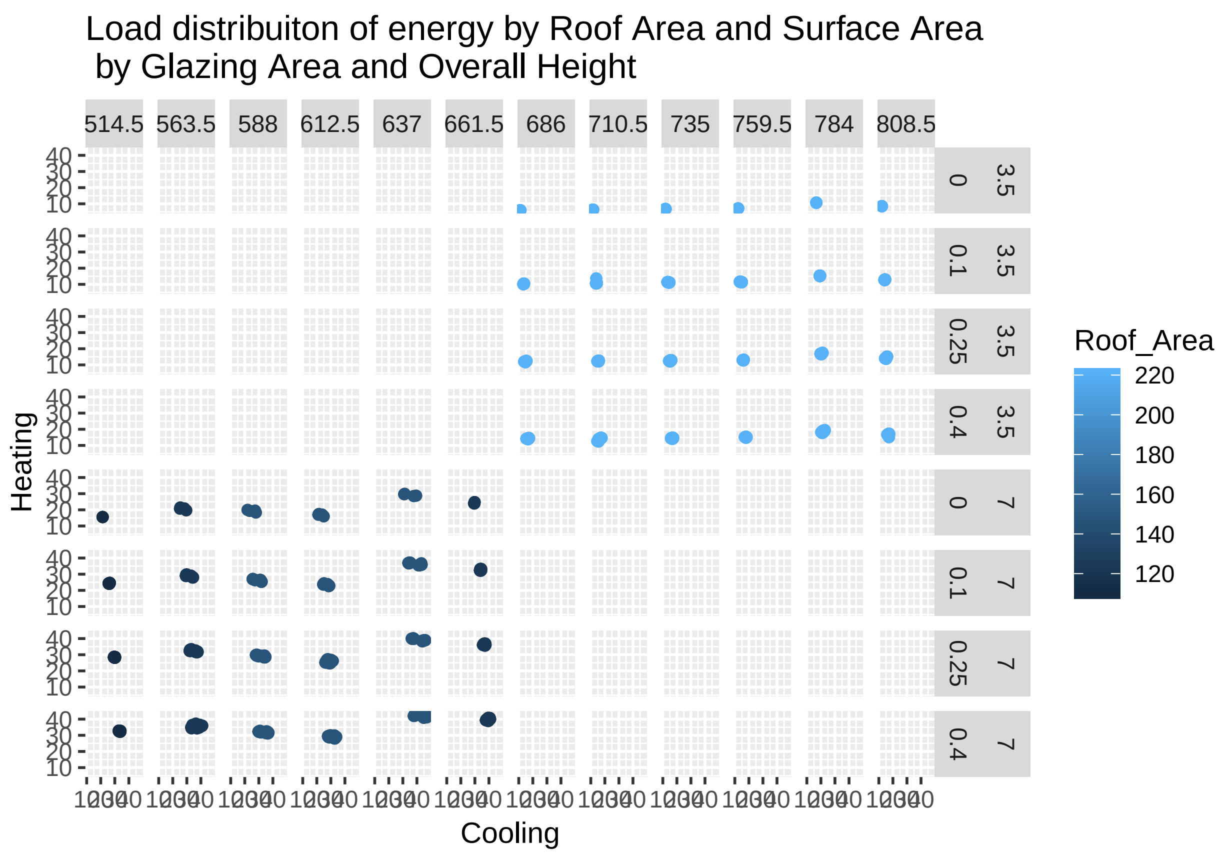
```r
hist(log(proj_data[,9]))
```




```r
# Displaying pairwise scatterplots and correlation, and histograms
pairs.panels(proj_data)
```




```
# Observing relation between Roof Area, Surface Area and Glazing Area and how the Load is distributed u
ggplot(proj_data, aes(x = Cooling, y = Heating), alpha = 0.3)+
  geom_point(aes(colour = Roof_Area ))+
  facet_grid(Ov_Hght + Glaz_Area ~ Surf_Area,  space = "free") +
  ggtitle("Load distribuiton of energy by Roof Area and Surface Area \n by Glazing Area and Overall Hei
```




```
# Function to detect outliers
outliers <- function(x)
{
  for(i in 1:ncol(x))
  {
    sd_i <- sd(x[,i])
    mean_i <- mean(x[,i])

    out = x[x[,i] > 3*sd_i+mean_i | x[,i] < mean_i-3*sd_i, ]
    if(nrow(out) > 0)
    {
      print(colnames(x)[i])
      paste("The outliers are -", out)
    }else
    {
      print(paste("No outliers for",colnames(x)[i]))
    }
  }
}

# Detecting outliers in the project dataset
outliers(proj_data)
```

```
## [1] "No outliers for Rel_Com"
## [1] "No outliers for Surf_Area"
## [1] "No outliers for Wall_Area"
```

```
## [1] "No outliers for Roof_Area"
## [1] "No outliers for Ov_Hght"
## [1] "No outliers for Orient"
## [1] "No outliers for Glaz_Area"
## [1] "No outliers for Glaz_A_Dist"
## [1] "No outliers for Heating"
## [1] "No outliers for Cooling"
```

Data Cleaning & Shaping

No NAs were found so adding NAs at random and then imputing them.

Normalizing the feature variables using min-max normalization.

Implementing principal component analysis.

```r
# Detecting NAs
proj_data[is.na(proj_data),]
```

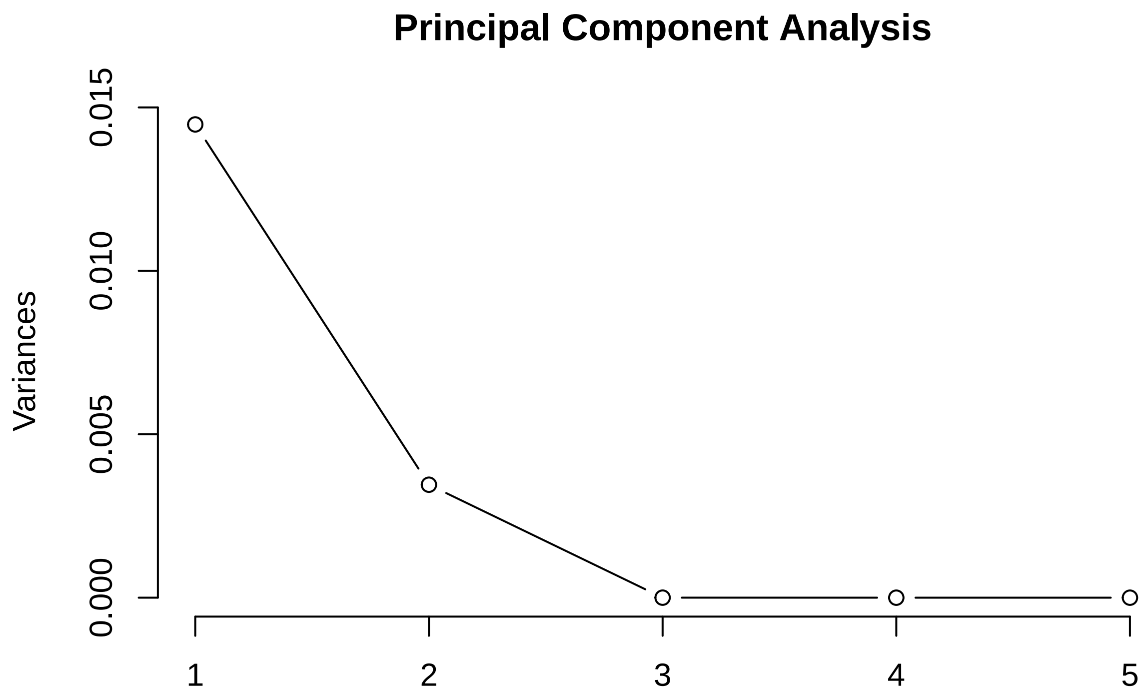
```
##  [1] Rel_Com     Surf_Area   Wall_Area   Roof_Area   Ov_Hght
##  [6] Orient      Glaz_Area   Glaz_A_Dist Heating     Cooling
## <0 rows> (or 0-length row.names)
```

```r
# Adding NAs for data imputation since none of them already exist
# Adding 10 NAs in random positions
data_w_NAs = proj_data
for (i in 1:10) {
  row = sample(1:768, 1)
  col = sample(1:8, 1)
  data_w_NAs[row, col] = NA
}
#t <- aggregate(data = proj_data, Rel_Com ~ Surf_Area, mean, na.rm = TRUE)

getMode <- function(x) {
  uniq <- unique(x)
  uniq[which.max(tabulate(match(x, uniq)))]
}

for(i in 1:nrow(data_w_NAs))
{
  for(j in 1:10)
  {
    if(j == 6 | j == 7 | j == 8)
    {
      if(is.na(data_w_NAs[i,j]))
      {
        data_w_NAs[i, j] = getMode(data_w_NAs[,j])
      }
    }
    else
    {
      if(is.na(data_w_NAs[i,j]))
      {
        paste(data_w_NAs[i,j])
        data_w_NAs[i, j] = mean(data_w_NAs[,j], na.rm = TRUE)
      }
```

```r
    }
  }
}

# Normalization function using min-max noramlization
normalize <- function(x)
{
  return ((x - min(x)) / (max(x) - min(x)))
}

# Normalizing the feature variables with continuous values
cont_v <- c(1:5)
data_norm <- cbind(normalize(proj_data[,cont_v]), proj_data[, c(6:10)])
data_norm_w_NA <- cbind(normalize(data_w_NAs[,cont_v]), data_w_NAs[, c(6:10)])

# PCA on the trained, scaled dataset
A1 = prcomp(data_norm[,1:5])

# Summary of the results
summary(A1)
```

```
## Importance of components:
##                           PC1     PC2       PC3       PC4      PC5
## Standard deviation     0.1203 0.05879 0.0005047 1.274e-05 2.73e-16
## Proportion of Variance 0.8073 0.19270 0.0000100 0.000e+00 0.00e+00
## Cumulative Proportion  0.8073 0.99999 1.0000000 1.000e+00 1.00e+00
```

```r
plot(A1, type="l", main = "Principal Component Analysis")
```



Model Construction & Evaluation

Cannot implement the hold-out method since unique combinations of data are observed in the dataset.

```
# Creating training and validation datasets
# sample <- sample.int(n = nrow(data_norm), size = 0.7*nrow(data_norm), replace = FALSE)
# train_data <- data_norm[sample,]
# validn_data <- data_norm[-sample,]
```

Building kNN regression model

Building a k Nearest Neighbors regression model to predict response values of the given data.

```
# Regression version of kNN
kNN.reg <- function(new_data, target_data, train_data, k)
{
  n <- nrow(train_data)
  d <- rep(0,n)
  for (i in 1:n)
  {
    d[i] <- sqrt(sum((train_data[i,1:8] - new_data[1:8])^2))
  }
  o <- order(d)
  m <- mean(target_data[o[1:k]])
  return(m)
}
```

```r
for(i in 1: nrow(data_norm))
{
  data_norm$Heating_reg_kNN[i] <- kNN.reg(data_norm[i,], data_norm[,9], data_norm[,1:8], 10)
  data_norm$Cooling_reg_kNN[i] <- kNN.reg(data_norm[i,], data_norm[,10], data_norm[,1:8], 10)
  data_norm_w_NA$Heating_reg_kNN[i] <- kNN.reg(data_norm_w_NA[i,], data_norm_w_NA[,9], data_norm_w_NA[,
  data_norm_w_NA$Cooling_reg_kNN[i] <- kNN.reg(data_norm_w_NA[i,], data_norm_w_NA[,10], data_norm_w_NA[
}

# data_norm_w_NA$Heating_reg_kNN <- reg_knn_heating_NA
# data_norm_w_NA$Cooling_reg_kNN <- reg_knn_cooling_NA

new_data <- c(0.69, 735.0, 294.0, 220.50, 3.5, 4, 0.25, 4)
for(i in 1:5)
{
  new_data[i] <- (new_data[i] - min(proj_data[,i])) / (max(proj_data[,i]) -
                                                          min(proj_data[,i]))
}

# Predicting response values of new data
kNN.reg(new_data, data_norm$Heating, data_norm[,1:8], 10)
```

```
## [1] 12.487
```

```r
kNN.reg(new_data, data_norm$Cooling, data_norm[,1:8], 10)
```

```
## [1] 15.468
```

Multiple Regression

Implementation of multiple regression for the two response variables.

```r
# Model for heating
model_heating <- lm(Heating ~ Rel_Com + Surf_Area + Wall_Area + Roof_Area + Ov_Hght +
                     Orient + Glaz_A_Dist + Glaz_Area, data = data_norm)

model_heating <- step(model_heating, direction = "backward")
```

```
## Start:  AIC=1661.42
## Heating ~ Rel_Com + Surf_Area + Wall_Area + Roof_Area + Ov_Hght +
##     Orient + Glaz_A_Dist + Glaz_Area
##
##
## Step:  AIC=1661.42
## Heating ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght + Orient +
##     Glaz_A_Dist + Glaz_Area
##
##               Df Sum of Sq    RSS    AIC
## - Orient       1       0.5 6544.3 1659.5
## <none>                     6543.8 1661.4
## - Glaz_A_Dist  1      73.1 6616.9 1668.0
## - Surf_Area    1     225.0 6768.8 1685.4
## - Rel_Com      1     341.2 6885.0 1698.5
## - Wall_Area    1     720.5 7264.3 1739.7
## - Ov_Hght      1    1310.6 7854.4 1799.6
```

```
## - Glaz_Area    1    5163.1 11706.9 2106.1
##
## Step:  AIC=1659.49
## Heating ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght + Glaz_A_Dist +
##     Glaz_Area
##
##               Df Sum of Sq     RSS    AIC
## <none>                      6544.3 1659.5
## - Glaz_A_Dist  1      73.1  6617.4 1666.0
## - Surf_Area    1     225.0  6769.3 1683.5
## - Rel_Com      1     341.2  6885.5 1696.5
## - Wall_Area    1     720.5  7264.8 1737.7
## - Ov_Hght      1    1310.6  7854.9 1797.7
## - Glaz_Area    1    5163.1 11707.4 2104.2
```

```r
summary(model_heating)
```

```
##
## Call:
## lm(formula = Heating ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght +
##     Glaz_A_Dist + Glaz_Area, data = data_norm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9315 -1.3189 -0.0262  1.3587  7.7169
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.634e+01  1.261e+01   3.675 0.000255 ***
## Rel_Com     -5.233e+04  8.308e+03  -6.299 5.06e-10 ***
## Surf_Area   -7.052e+01  1.379e+01  -5.115 3.97e-07 ***
## Wall_Area    4.913e+01  5.367e+00   9.153  < 2e-16 ***
## Ov_Hght      3.369e+03  2.729e+02  12.345  < 2e-16 ***
## Glaz_A_Dist  2.038e-01  6.987e-02   2.916 0.003646 **
## Glaz_Area    1.993e+01  8.135e-01  24.503  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.933 on 761 degrees of freedom
## Multiple R-squared:  0.9162, Adjusted R-squared:  0.9155
## F-statistic:  1387 on 6 and 761 DF,  p-value: < 2.2e-16
```

```r
data_norm$Heating_lm <- predict(model_heating, data_norm)
data_norm_w_NA$Heating_lm <- predict(model_heating, data_norm_w_NA)

# Model for cooling
model_cooling <- lm(Cooling ~ Rel_Com + Surf_Area + Wall_Area + Roof_Area + Ov_Hght +
                     Orient + Glaz_A_Dist + Glaz_Area, data = data_norm)

model_cooling <- step(model_cooling, direction = "backward")
```

```
## Start:  AIC=1795.13
## Cooling ~ Rel_Com + Surf_Area + Wall_Area + Roof_Area + Ov_Hght +
##     Orient + Glaz_A_Dist + Glaz_Area
##
```

```
##
## Step:  AIC=1795.13
## Cooling ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght + Orient +
##     Glaz_A_Dist + Glaz_Area
##
##               Df Sum of Sq     RSS    AIC
## - Glaz_A_Dist  1      2.92  7791.1 1793.4
## - Orient       1     14.17  7802.4 1794.5
## <none>                      7788.2 1795.1
## - Surf_Area    1    229.96  8018.2 1815.5
## - Wall_Area    1    388.96  8177.2 1830.6
## - Rel_Com      1    407.52  8195.7 1832.3
## - Ov_Hght      1   1383.16  9171.4 1918.7
## - Glaz_Area    1   2814.64 10602.8 2030.1
##
## Step:  AIC=1793.42
## Cooling ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght + Orient +
##     Glaz_Area
##
##             Df Sum of Sq     RSS    AIC
## - Orient     1     14.17  7805.3 1792.8
## <none>                    7791.1 1793.4
## - Surf_Area  1    229.96  8021.1 1813.8
## - Wall_Area  1    388.96  8180.1 1828.8
## - Rel_Com    1    407.52  8198.6 1830.6
## - Ov_Hght    1   1383.16  9174.3 1916.9
## - Glaz_Area  1   2988.93 10780.0 2040.8
##
## Step:  AIC=1792.81
## Cooling ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght + Glaz_Area
##
##             Df Sum of Sq     RSS    AIC
## <none>                    7805.3 1792.8
## - Surf_Area  1    229.96  8035.3 1813.1
## - Wall_Area  1    388.96  8194.3 1828.2
## - Rel_Com    1    407.52  8212.8 1829.9
## - Ov_Hght    1   1383.16  9188.5 1916.1
## - Glaz_Area  1   2988.93 10794.2 2039.8
```

```r
summary(model_cooling)
```

```
##
## Call:
## lm(formula = Cooling ~ Rel_Com + Surf_Area + Wall_Area + Ov_Hght +
##     Glaz_Area, data = data_norm)
##
## 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)  5.650e+01  1.376e+01   4.106 4.47e-05 ***
## Rel_Com     -5.719e+04  9.067e+03  -6.307 4.80e-10 ***
## Surf_Area   -7.129e+01  1.505e+01  -4.738 2.57e-06 ***
```

```
## Wall_Area    3.610e+01  5.858e+00   6.162 1.16e-09 ***
## Ov_Hght      3.461e+03  2.978e+02  11.620  < 2e-16 ***
## Glaz_Area    1.482e+01  8.675e-01  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
## F-statistic:  1203 on 5 and 762 DF,  p-value: < 2.2e-16
```

```
data_norm$Cooling_lm <- predict(model_cooling, data_norm)
data_norm_w_NA$Cooling_lm <- predict(model_cooling, data_norm_w_NA)
```

Evaluation with k-fold cross-validation

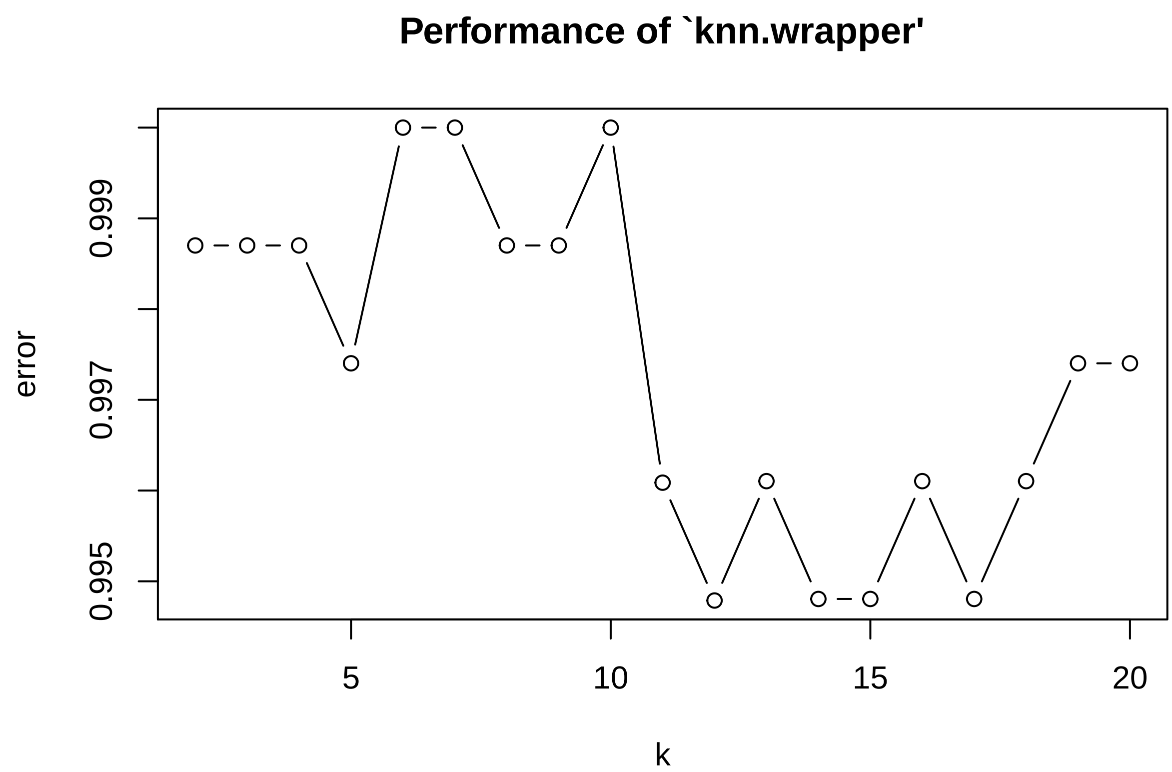
Implementing k-fold cross-validation

```
library(e1071)
# Heating
# Full Data set used for cross validation
knn.cross <- tune.knn(x = data_norm[,1:8], y = factor(data_norm[,9]),
                      k = 2:20,tunecontrol=tune.control(sampling = "cross"),
                      cross=10)

# Summarize the resampling results set
summary(knn.cross)
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   k
##  12
##
## - best performance: 0.9947881
##
## - Detailed performance results:
##     k     error  dispersion
## 1   2 0.9987013 0.004106854
## 2   3 0.9987013 0.004106854
## 3   4 0.9987013 0.004106854
## 4   5 0.9974026 0.005475805
## 5   6 1.0000000 0.000000000
## 6   7 1.0000000 0.000000000
## 7   8 0.9987013 0.004106854
## 8   9 0.9987013 0.004106854
## 9  10 1.0000000 0.000000000
## 10 11 0.9960868 0.006301009
## 11 12 0.9947881 0.006728706
## 12 13 0.9961039 0.006273323
## 13 14 0.9948052 0.006706465
## 14 15 0.9948052 0.006706465
```

```
## 15 16 0.9961039 0.006273323
## 16 17 0.9948052 0.006706465
## 17 18 0.9961039 0.006273323
## 18 19 0.9974026 0.005475805
## 19 20 0.9974026 0.005475805
```

```r
plot(knn.cross)
```

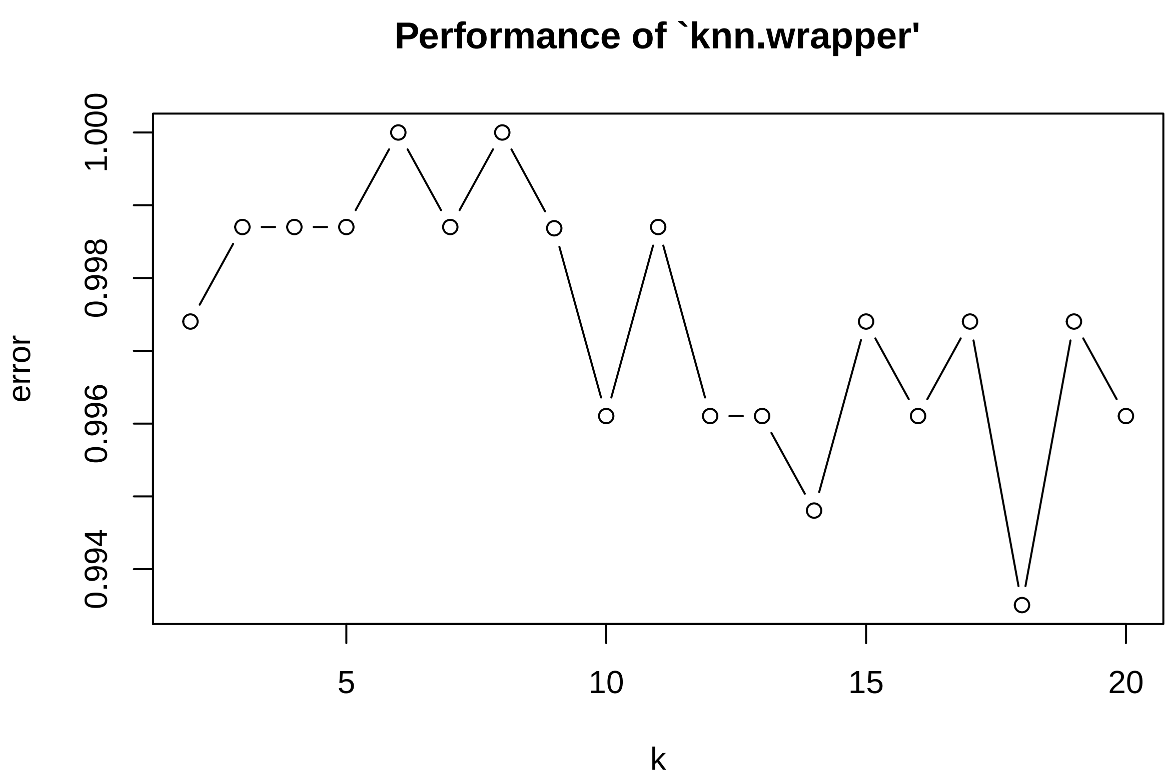



```r
# Cooling
# Full Data set used for cross validation
knn.cross <- tune.knn(x = data_norm[,1:8], y = factor(proj_data[,10]),
                      k = 2:20,tunecontrol=tune.control(sampling = "cross"),
                      cross=10)

# Summarize the resampling results set
summary(knn.cross)
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   k
##  18
##
## - best performance: 0.9935065
```

```
##
## - Detailed performance results:
##     k     error  dispersion
## 1   2 0.9974026 0.005475805
## 2   3 0.9987013 0.004106854
## 3   4 0.9987013 0.004106854
## 4   5 0.9987013 0.004106854
## 5   6 1.0000000 0.000000000
## 6   7 0.9987013 0.004106854
## 7   8 1.0000000 0.000000000
## 8   9 0.9986842 0.004160892
## 9  10 0.9961039 0.006273323
## 10 11 0.9987013 0.004106854
## 11 12 0.9961039 0.006273323
## 12 13 0.9961039 0.006273323
## 13 14 0.9948052 0.009080596
## 14 15 0.9974026 0.005475805
## 15 16 0.9961039 0.006273323
## 16 17 0.9974026 0.005475805
## 17 18 0.9935065 0.009183205
## 18 19 0.9974026 0.005475805
## 19 20 0.9961039 0.006273323
```

```r
plot(knn.cross)
```



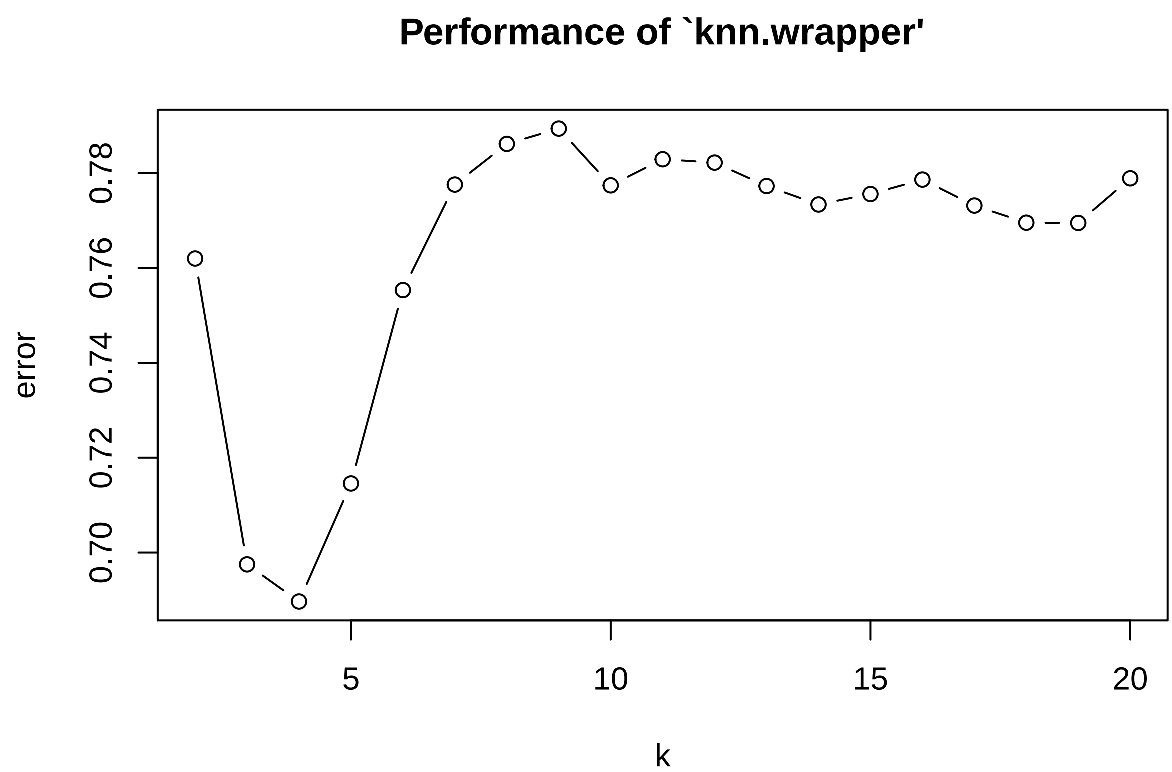
Tuning the models

Tuning the kNN model with bootstrap sampling

```
library(e1071)
# Heating
# Resampling using bootstrap
knn.boot <- tune.knn(x = proj_data[,1:8], y = factor(round(proj_data[,9])),
                     k = 2:20,tunecontrol=tune.control(sampling = "boot"),
                     boot=10)

# Summarize the resampling results set
summary(knn.boot)
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: bootstrapping
##
## - best parameters:
##  k
##  4
##
## - best performance: 0.6896787
##
## - Detailed performance results:
##     k     error dispersion
## 1   2 0.7619906 0.01989461
## 2   3 0.6975055 0.02659779
## 3   4 0.6896787 0.01469988
## 4   5 0.7145609 0.01811507
## 5   6 0.7553324 0.02079314
## 6   7 0.7775762 0.02579225
## 7   8 0.7861551 0.01600672
## 8   9 0.7893767 0.01861030
## 9  10 0.7774135 0.01496308
## 10 11 0.7829199 0.02084878
## 11 12 0.7822094 0.01477183
## 12 13 0.7772686 0.01912851
## 13 14 0.7733776 0.01528082
## 14 15 0.7755801 0.02543279
## 15 16 0.7786291 0.02179582
## 16 17 0.7731533 0.01940249
## 17 18 0.7695467 0.01680867
## 18 19 0.7694878 0.01359562
## 19 20 0.7788908 0.01683111
```

```
plot(knn.boot)
```

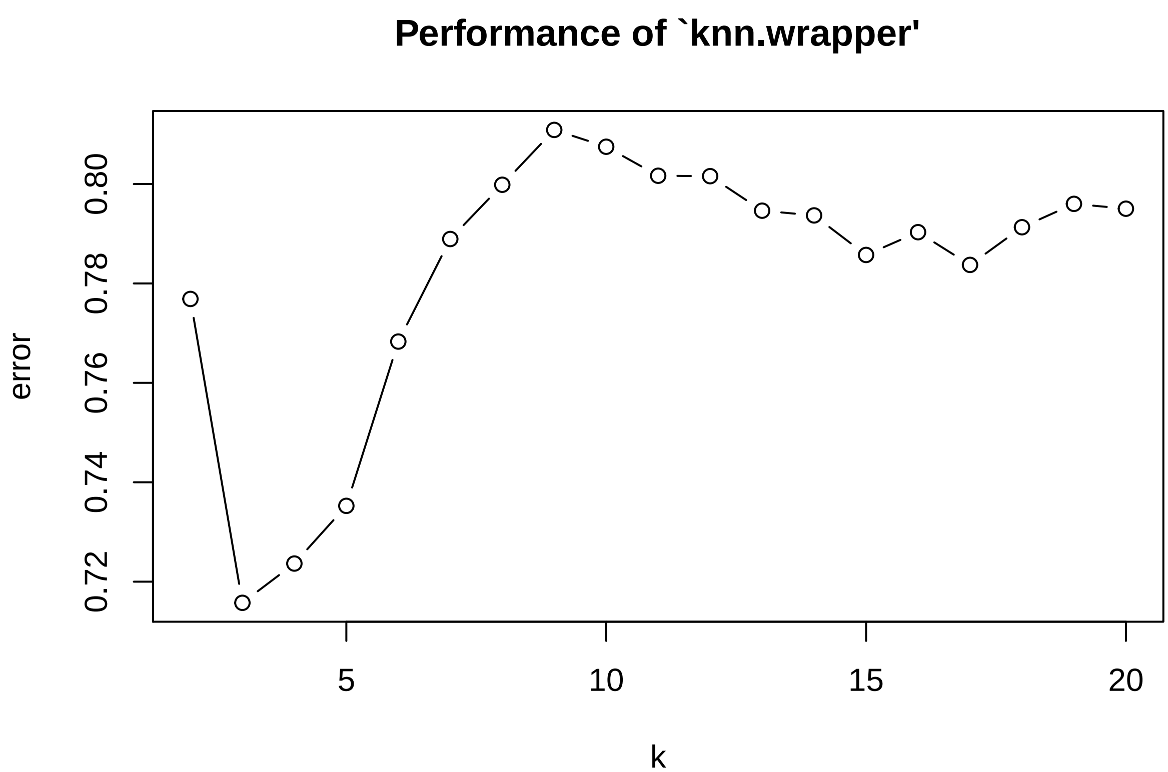



```
# Cooling
# Resampling using bootstrap
knn.boot <- tune.knn(x = proj_data[,1:8], y = factor(round(proj_data[,10])),
                     k = 2:20,tunecontrol=tune.control(sampling = "boot"),
                     boot=10)

# Summarize the resampling results set
summary(knn.boot)
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: bootstrapping
##
## - best parameters:
##  k
##  3
##
## - best performance: 0.715745
##
## - Detailed performance results:
##     k     error dispersion
## 1   2 0.7768812 0.02533222
## 2   3 0.7157450 0.01877200
## 3   4 0.7236445 0.01850626
## 4   5 0.7352533 0.02701647
```

```
## 5    6 0.7683032 0.02958669
## 6    7 0.7889383 0.01964626
## 7    8 0.7998505 0.01677028
## 8    9 0.8108874 0.01806963
## 9   10 0.8075052 0.01479634
## 10  11 0.8016689 0.02085991
## 11  12 0.8015838 0.01319653
## 12  13 0.7946382 0.01796927
## 13  14 0.7936899 0.01727310
## 14  15 0.7857330 0.01475060
## 15  16 0.7903163 0.01285067
## 16  17 0.7837333 0.01186579
## 17  18 0.7913104 0.01639699
## 18  19 0.7960108 0.01591710
## 19  20 0.7950369 0.01101564
```

```
plot(knn.boot)
```



Comparison using RMSE

Comparing the various models using RMSE.

```
calc_RMSE <- function(orig, pred)
{
  sqrt(mean((orig-pred)^2))
}
```

```r
RMSE_knn_heating <- calc_RMSE(data_norm$Heating, data_norm$Heating_reg_kNN)
RMSE_knn_cooling <- calc_RMSE(data_norm$Cooling, data_norm$Cooling_reg_kNN)
RMSE_lm_heating <- calc_RMSE(data_norm$Heating, data_norm$Heating_reg_kNN)
RMSE_lm_cooling <- calc_RMSE(data_norm$Cooling, data_norm$Cooling_reg_kNN)

crit = c("RMSE_knn_heating", "RMSE_knn_cooling", "RMSE_lm_heating", "RMSE_lm_cooling")
rmses = c(RMSE_knn_heating, RMSE_knn_cooling, RMSE_lm_heating, RMSE_lm_cooling)
Summary <- data.frame(Criteria = crit, RMSE = rmses)
Summary
```

```
##           Criteria     RMSE
## 1 RMSE_knn_heating 4.945846
## 2 RMSE_knn_cooling 4.863129
## 3  RMSE_lm_heating 4.945846
## 4  RMSE_lm_cooling 4.863129
```

```r
RMSE_knn_heating_NA <- calc_RMSE(data_norm_w_NA$Heating, data_norm_w_NA$Heating_reg_kNN)
RMSE_knn_cooling_NA <- calc_RMSE(data_norm_w_NA$Cooling, data_norm_w_NA$Cooling_reg_kNN)
RMSE_lm_heating_NA <- calc_RMSE(data_norm_w_NA$Heating, data_norm_w_NA$Heating_reg_kNN)
RMSE_lm_cooling_NA <- calc_RMSE(data_norm_w_NA$Cooling, data_norm_w_NA$Cooling_reg_kNN)

crit = c("RMSE_knn_heating_NA", "RMSE_knn_cooling_NA", "RMSE_lm_heating_NA", "RMSE_lm_cooling_NA")
rmses = c(RMSE_knn_heating_NA, RMSE_knn_cooling_NA, RMSE_lm_heating_NA, RMSE_lm_cooling_NA)
Summary_NA <- data.frame(Criteria = crit, RMSE = rmses)
Summary_NA
```

```
##              Criteria     RMSE
## 1 RMSE_knn_heating_NA 4.911248
## 2 RMSE_knn_cooling_NA 4.835057
## 3  RMSE_lm_heating_NA 4.911248
## 4  RMSE_lm_cooling_NA 4.835057
```