Energy Efficiency Prediction

Hrithik Agarwal

07/09/2020

About the dataset

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.

Attribute Informtion

The dataset contains eight attributes (or features, denoted by X1…X8) and two responses (or outcomes, denoted by Y1 and Y2). The aim is to use the eight features to predict each of the two responses.

Specifically:

1. X1 Relative Compactness
2. X2 Surface Area
3. X3 Wall Area
4. X4 Roof Area
5. X5 Overall Height
6. X6 Orientation
7. X7 Glazing Area
8. X8 Glazing Area Distribution
9. Y1 Heating Load
10. Y2 Cooling Load

Citation

The data was collected from - A. Tsanas, A. Xifara: ‘Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools’, Energy and Buildings, Vol. 49, pp. 560-567, 2012

#Loading the required libraries

```
library(caTools)
library(caret)
library(lattice)
library(corrplot)
library(ggplot2)
library(Metrics)
```

Importing the dataset

```
data <- read.csv('energy csv.csv' , header =T)
```

Observation of dataset

```
head(data)
```

```
##     X1    X2    X3     X4 X5 X6 X7 X8    Y1    Y2
## 1 0.98 514.5 294.0 110.25  7  2  0  0 15.55 21.33
## 2 0.98 514.5 294.0 110.25  7  3  0  0 15.55 21.33
## 3 0.98 514.5 294.0 110.25  7  4  0  0 15.55 21.33
## 4 0.98 514.5 294.0 110.25  7  5  0  0 15.55 21.33
## 5 0.90 563.5 318.5 122.50  7  2  0  0 20.84 28.28
## 6 0.90 563.5 318.5 122.50  7  3  0  0 21.46 25.38
```

```
str(data)
```

```
## 'data.frame':    768 obs. of  10 variables:
##  $ X1: num  0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...
##  $ X2: num  514 514 514 514 564 ...
##  $ X3: num  294 294 294 294 318 ...
##  $ X4: num  110 110 110 110 122 ...
##  $ X5: num  7 7 7 7 7 7 7 7 7 7 ...
##  $ X6: int  2 3 4 5 2 3 4 5 2 3 ...
##  $ X7: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X8: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Y1: num  15.6 15.6 15.6 15.6 20.8 ...
##  $ Y2: num  21.3 21.3 21.3 21.3 28.3 ...
```

From the str() there is an indication that variable X4, X5, X6, X7, X8 may be factors.

```
table(data$X4)
```

```
##
## 110.25  122.5    147  220.5
##     64    128    192    384
```

```
table(data$X5)
```

```
##
## 3.5   7
## 384 384
```

```
table(data$X6)
```

```
##
##   2   3   4   5
## 192 192 192 192
```

```
table(data$X7)
```

```
##
##    0  0.1 0.25  0.4
##   48  240  240  240
```

```
table(data$X8)
```

```
##
##   0   1   2   3   4   5
##  48 144 144 144 144 144
```

It is clearly seen that the above variables are factors so we'll convert them for better performance of model

Converting num to factor

```
data$X4 <- as.factor(data$X4)
data$X5 <- as.factor(data$X5)
data$X6 <- as.factor(data$X6)
data$X7 <- as.factor(data$X7)
data$X8 <- as.factor(data$X8)
```

Now that The required variables data type has been corrected. Let's see for missing data

```
summary(data)
```

```
##        X1                X2               X3              X4         X5
##  Min.   :0.6200   Min.   :514.5   Min.   :245.0   110.25: 64   3.5:384
##  1st Qu.:0.6825   1st Qu.:606.4   1st Qu.:294.0   122.5 :128   7  :384
##  Median :0.7500   Median :673.8   Median :318.5   147   :192
##  Mean   :0.7642   Mean   :671.7   Mean   :318.5   220.5 :384
##  3rd Qu.:0.8300   3rd Qu.:741.1   3rd Qu.:343.0
##  Max.   :0.9800   Max.   :808.5   Max.   :416.5
##  X6         X7        X8             Y1              Y2
##  2:192   0    : 48   0: 48   Min.   : 6.01   Min.   :10.90
##  3:192   0.1 :240    1:144   1st Qu.:12.99   1st Qu.:15.62
##  4:192   0.25:240    2:144   Median :18.95   Median :22.08
##  5:192   0.4 :240    3:144   Mean   :22.31   Mean   :24.59
##                      4:144   3rd Qu.:31.67   3rd Qu.:33.13
##                      5:144   Max.   :43.10   Max.   :48.03
```

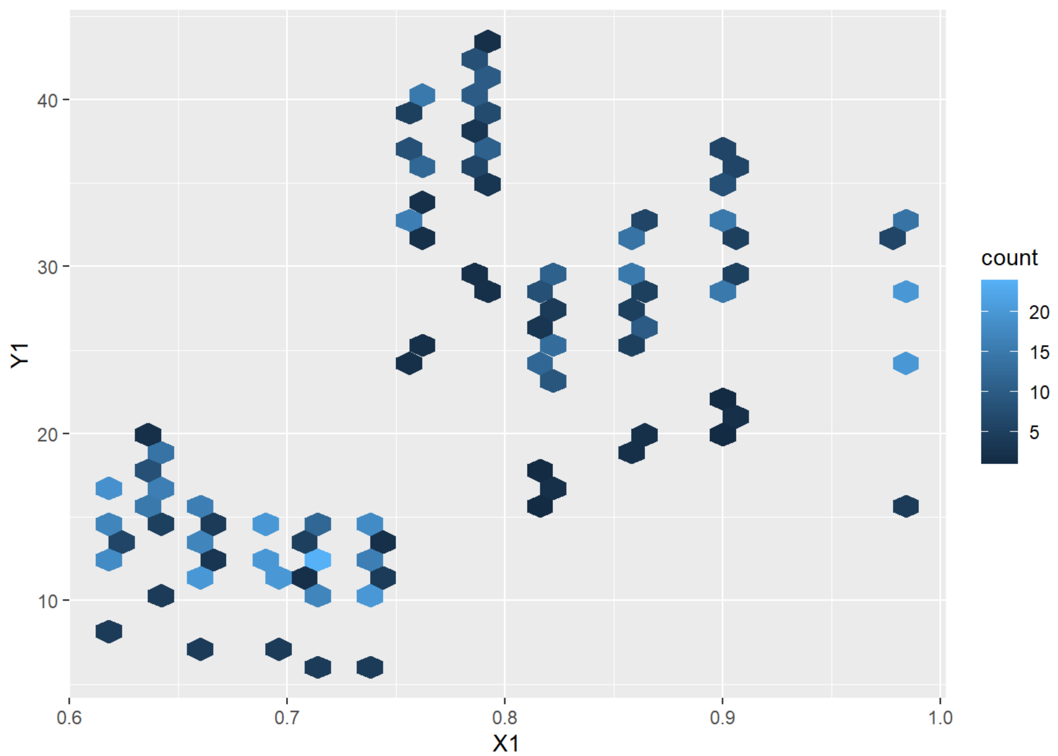
This indicates that there are no missing values. So we can proceed with data splitting , data visualization. and model making.

Data visualization

Now Let's visualize all the variables with respect to Y1 and Y2.

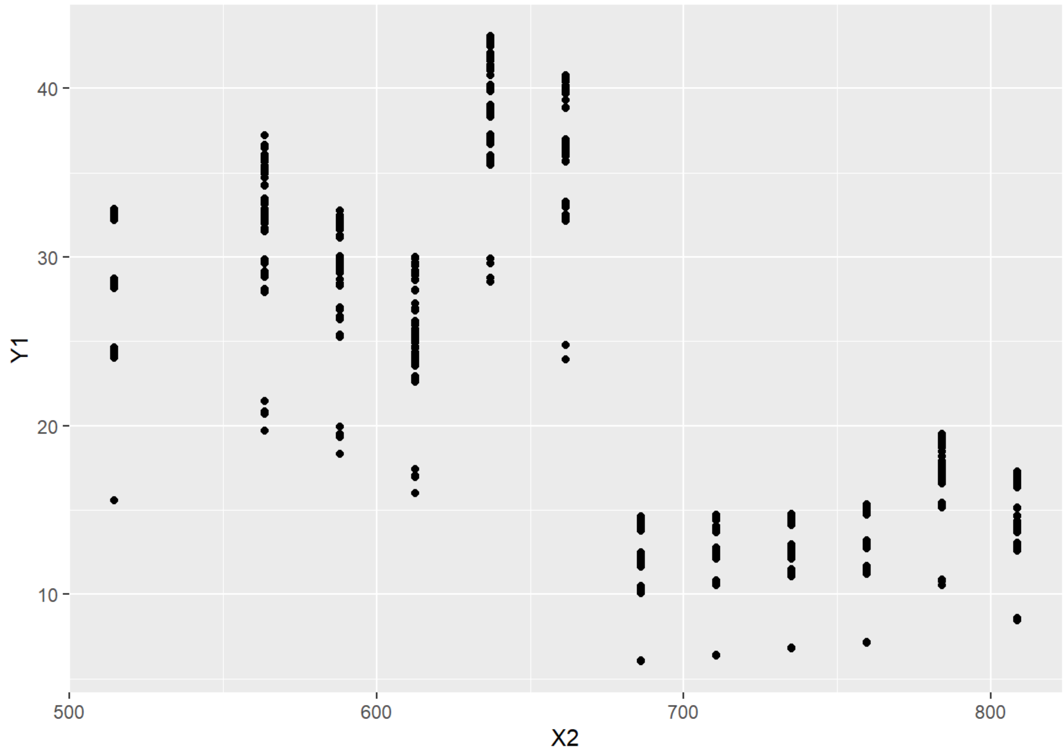
Data visualization w.r.t Y1

```
ggplot(data, aes(X1,Y1)) + geom_hex()
```



This plot gives the impression that when X1 is less than 0.75 , Y1 is less than 20. X1 is greater than 0.75, Y1 is mostly greater than 20.

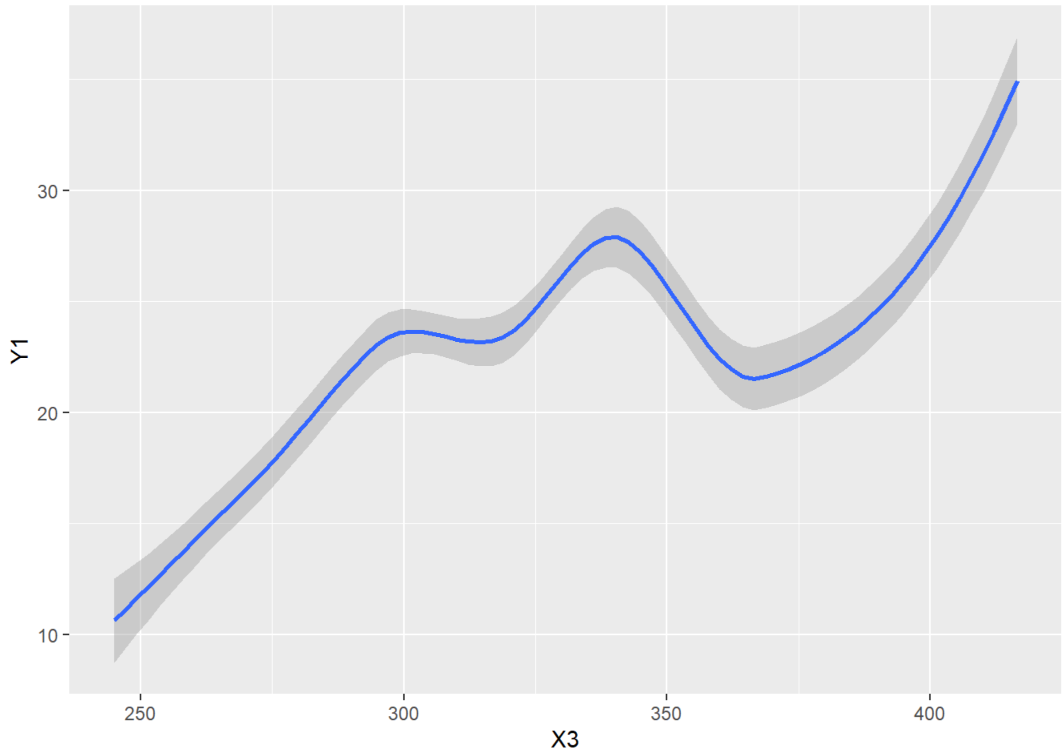
```
ggplot(data, aes(X2,Y1)) + geom_point()
```



This graph shows the reverse of of the previous graph.

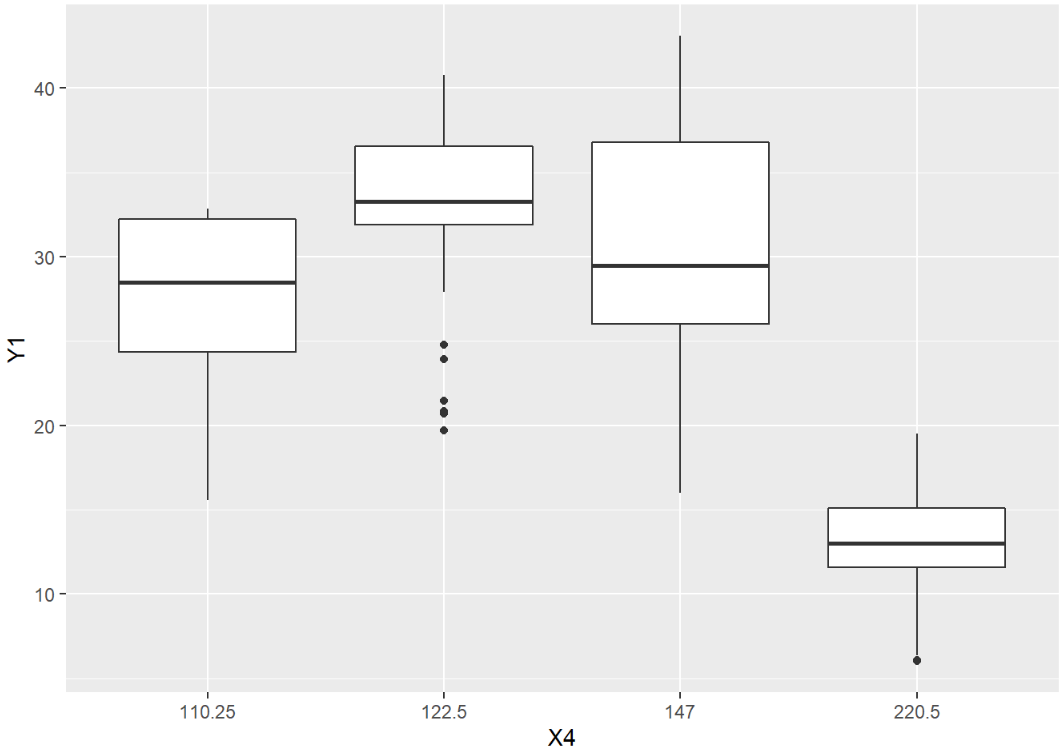
```
ggplot(data, aes(X3,Y1)) + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



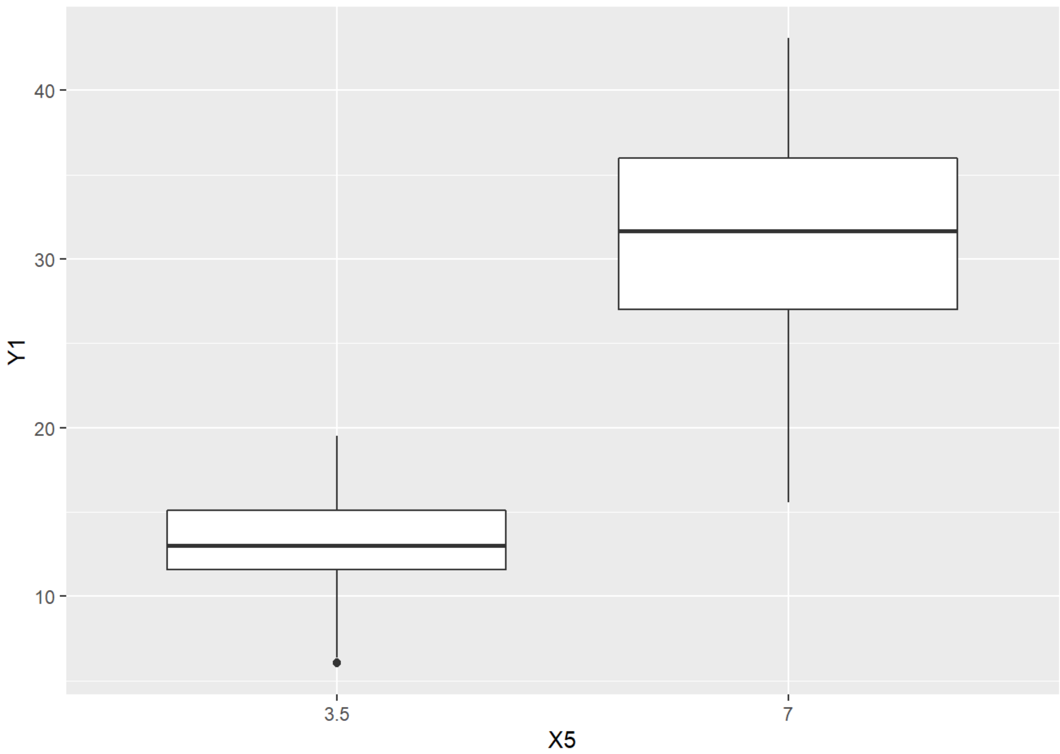
From this we can see that how well the X3 variable fits in Y1.

```
ggplot(data, aes(X4,Y1)) + geom_boxplot()
```



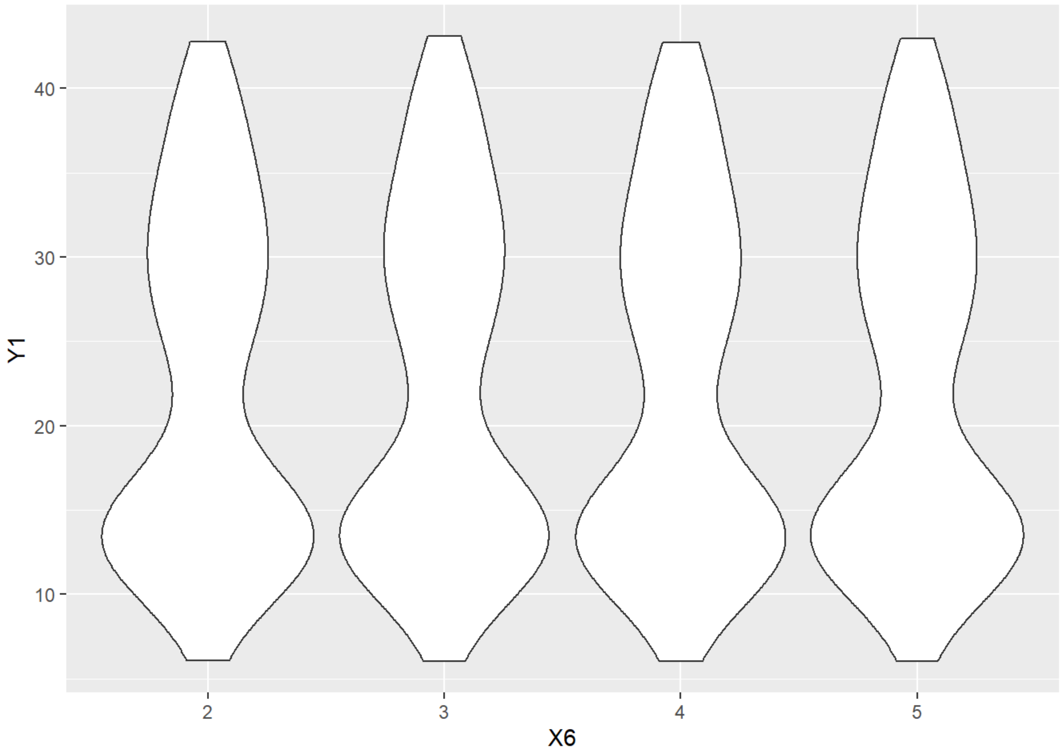
Distribution of X4 w.r.t Y1 where X4 = 122.5 has many outliers.

```
ggplot(data, aes(X5,Y1)) + geom_boxplot()
```



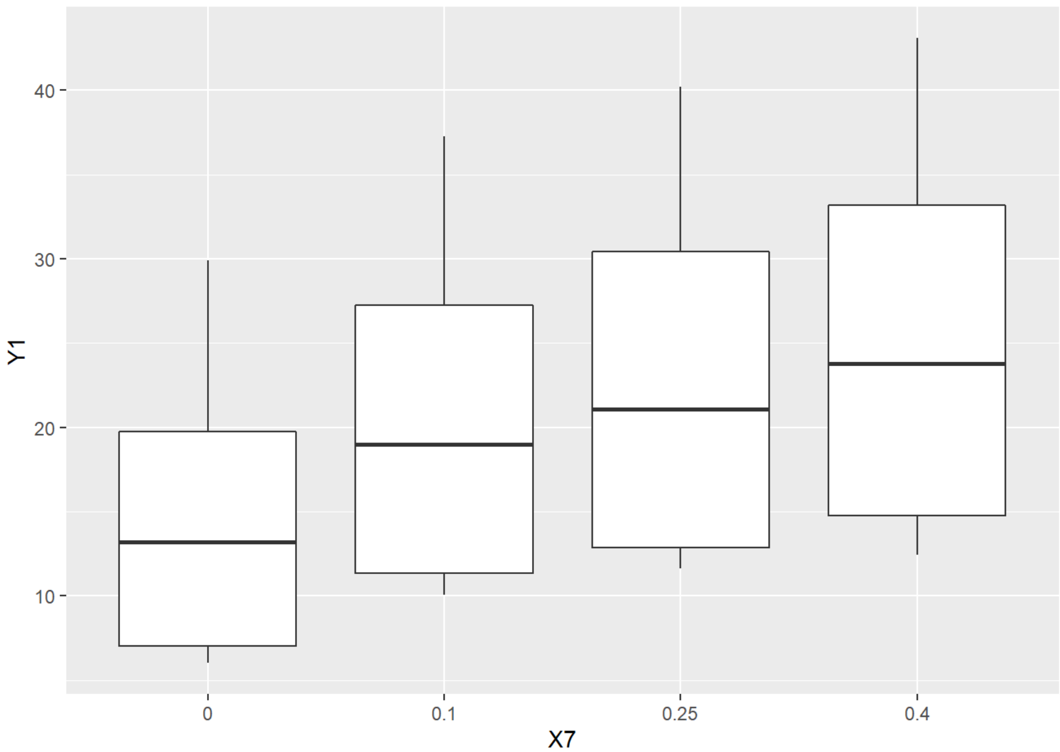
Distribution of X5 w.r.t Y1 where X5 = 7 plays a major role.

```
ggplot(data, aes(X6,Y1)) + geom_violin()
```



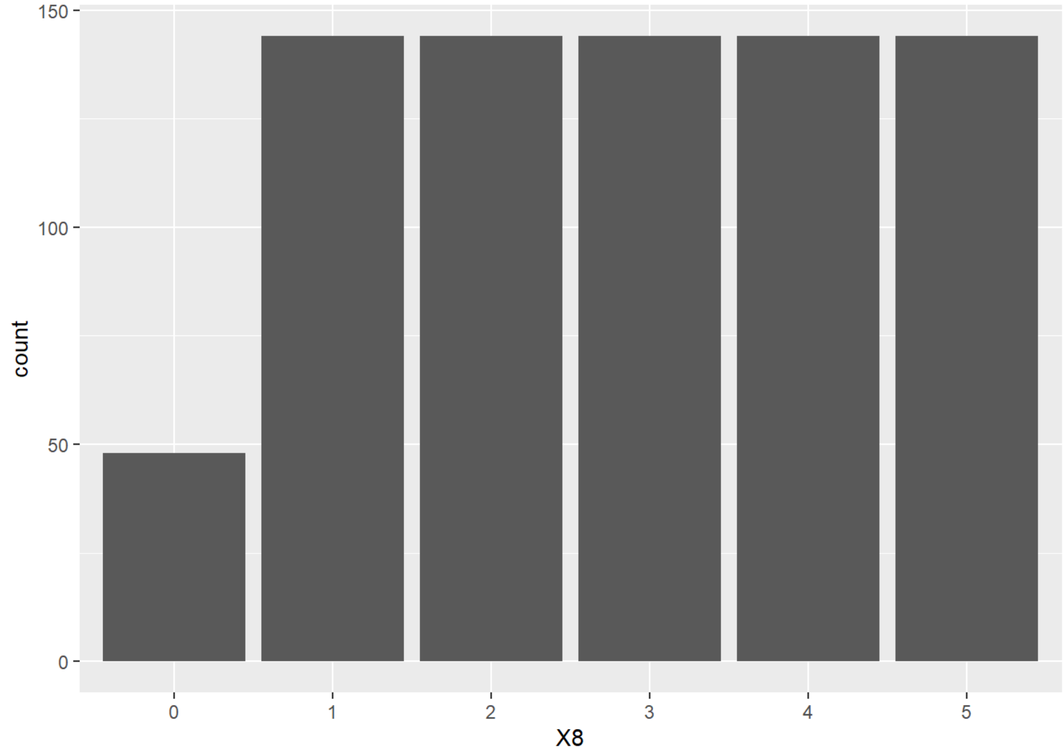
This plot shows that X6 has equal and same distrbution over its all four values.

```
ggplot(data, aes(X7,Y1)) + geom_boxplot()
```



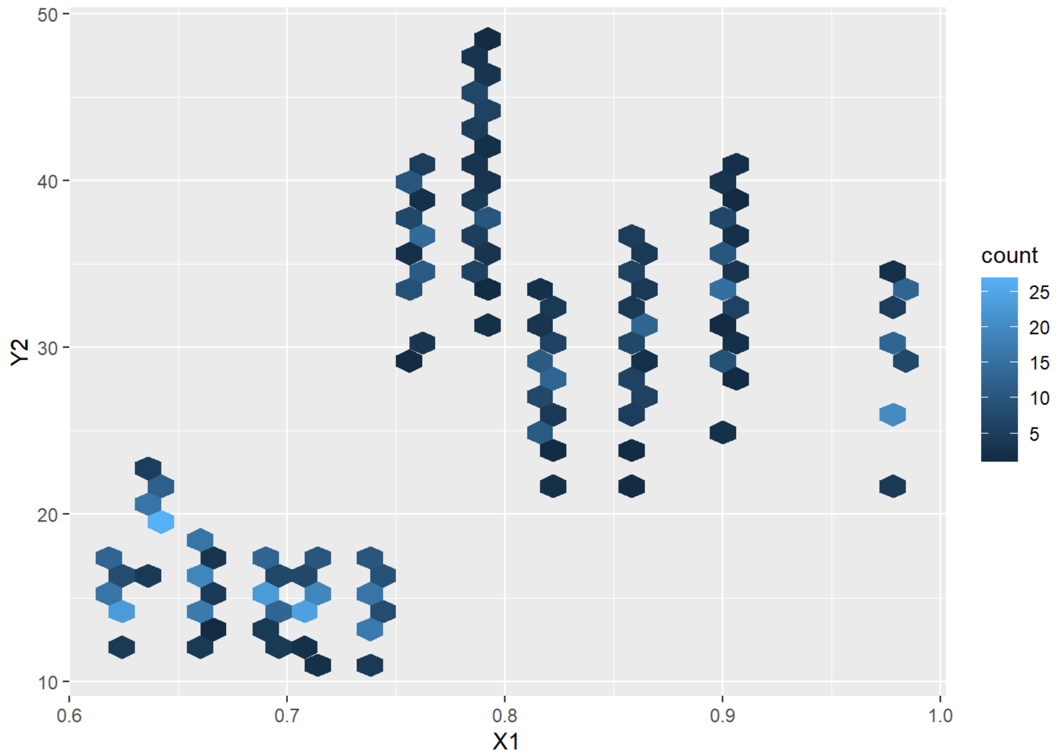
This also shows equal distribution over its all 4 values.

```
ggplot(data, aes(X8)) + geom_bar()
```



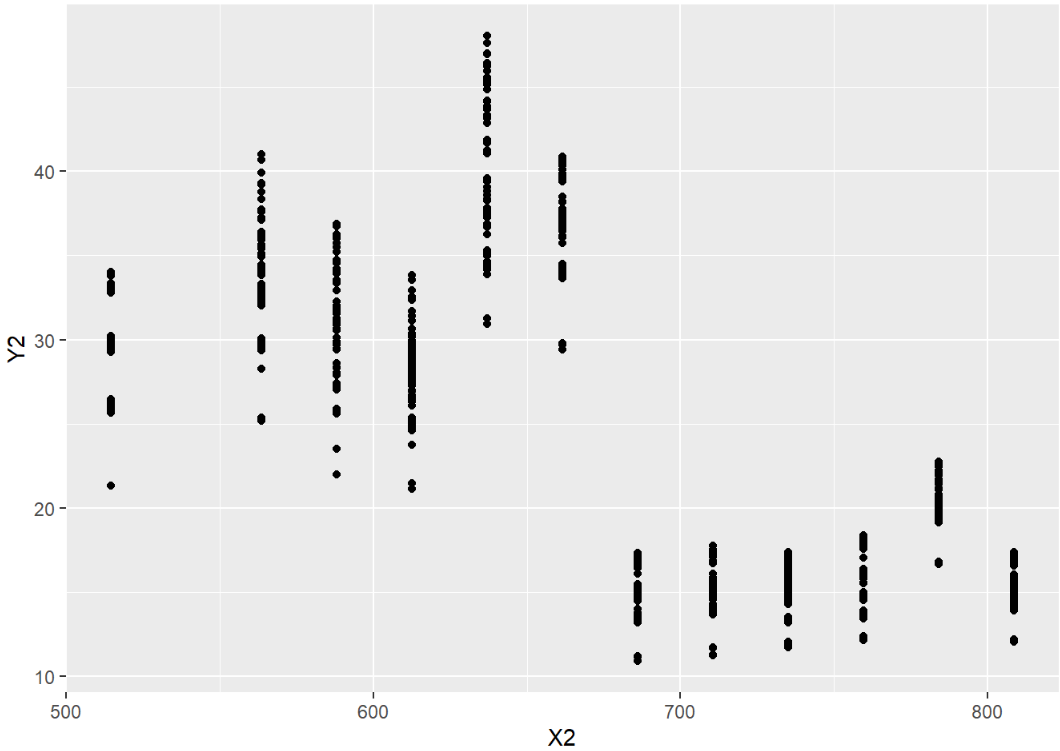
Data visualization w.r.t Y2

```
ggplot(data, aes(X1,Y2)) + geom_hex()
```



This plot gives the impression that when X1 is less than 0.75 , Y1 is less than 20. X1 is greater than 0.75, Y1 is greater than 20.

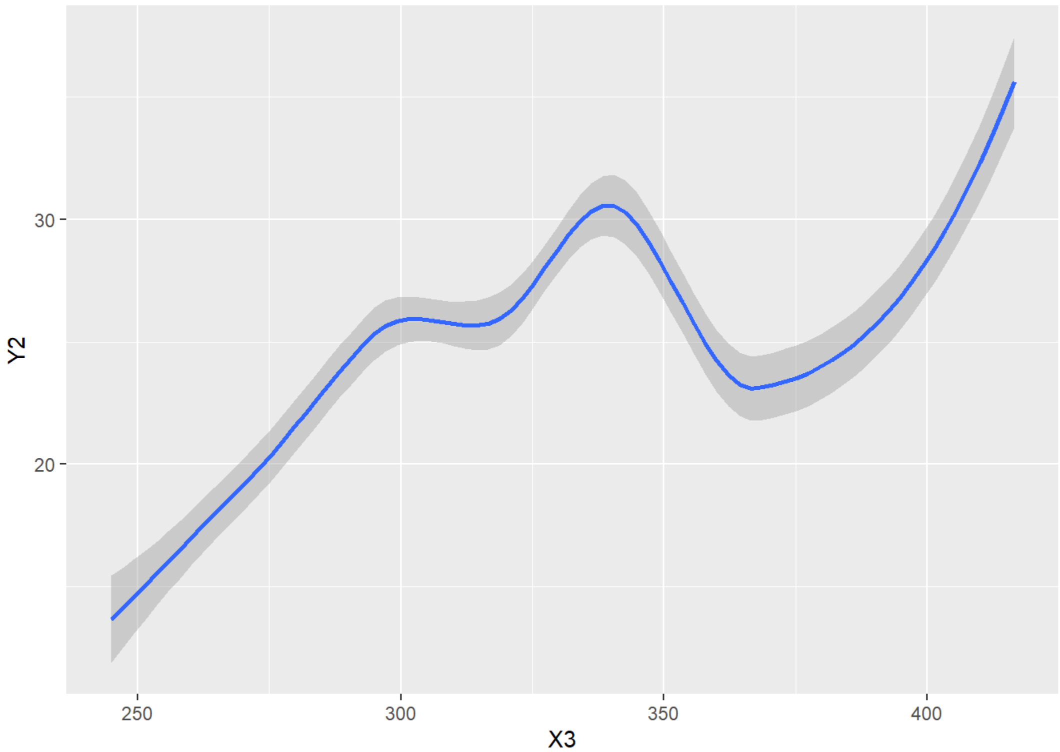
```
ggplot(data, aes(X2,Y2)) + geom_point()
```



This graph shows the reverse of of the previous graph.

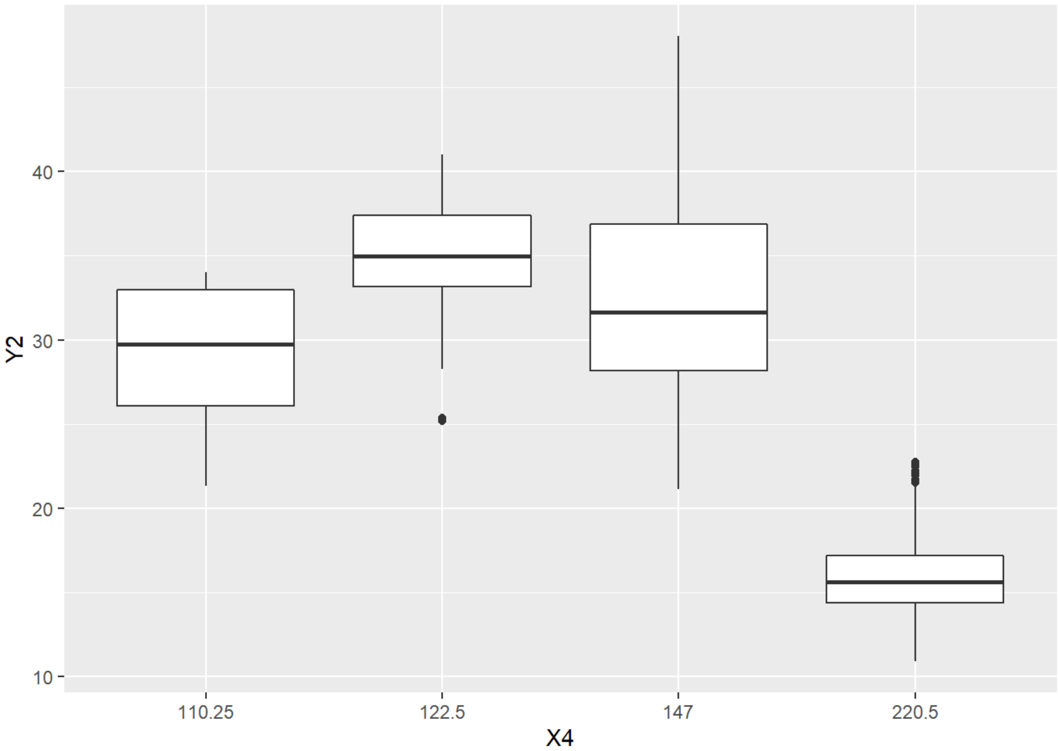
```
ggplot(data, aes(X3,Y2)) + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



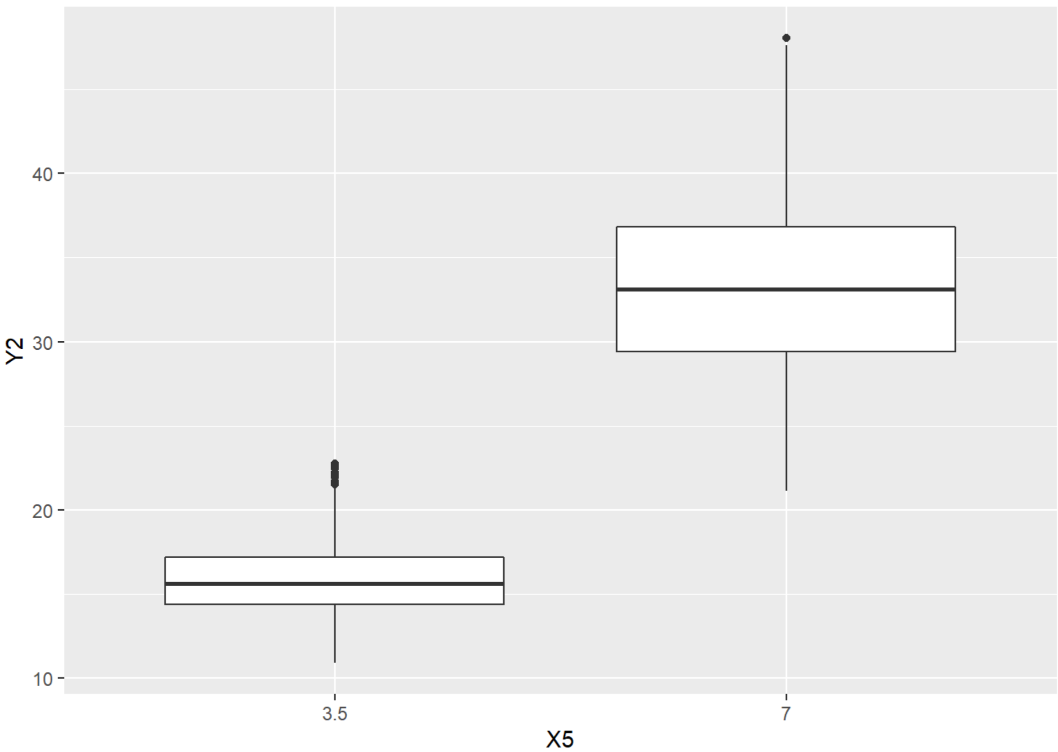
From this we can see that how well the X3 variable fits in Y1.

```
ggplot(data, aes(X4,Y2)) + geom_boxplot()
```



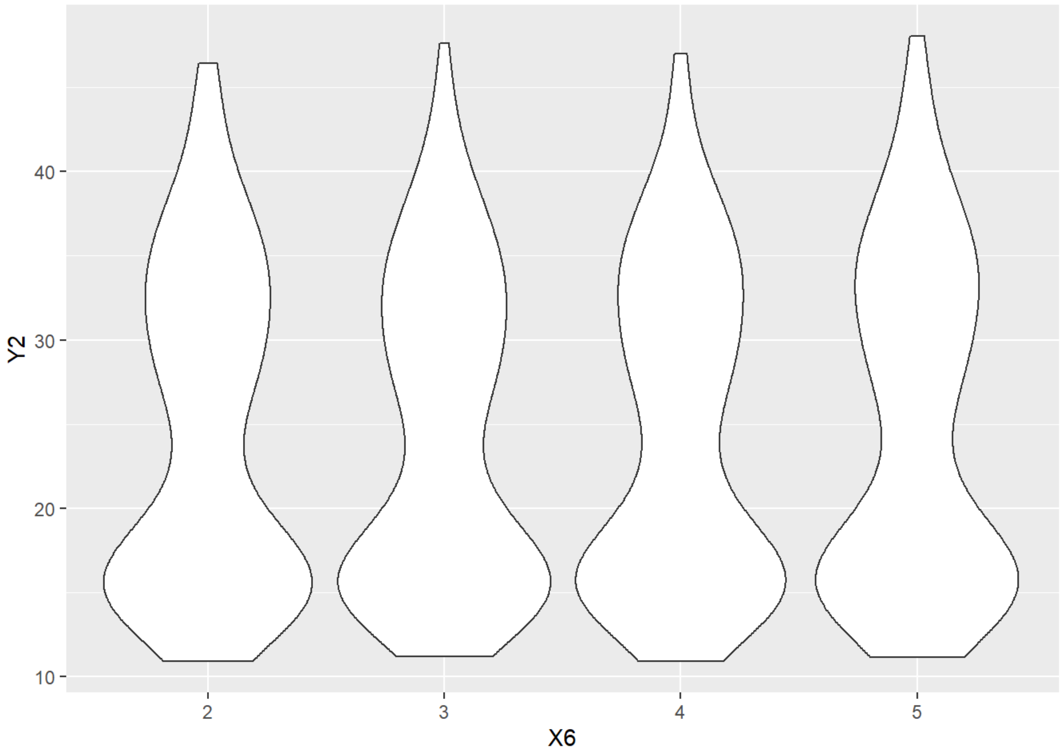
Distribution of X4 w.r.t Y1 where X4 147 is the value which is taken by many Y2.

```
ggplot(data, aes(X5,Y2)) + geom_boxplot()
```



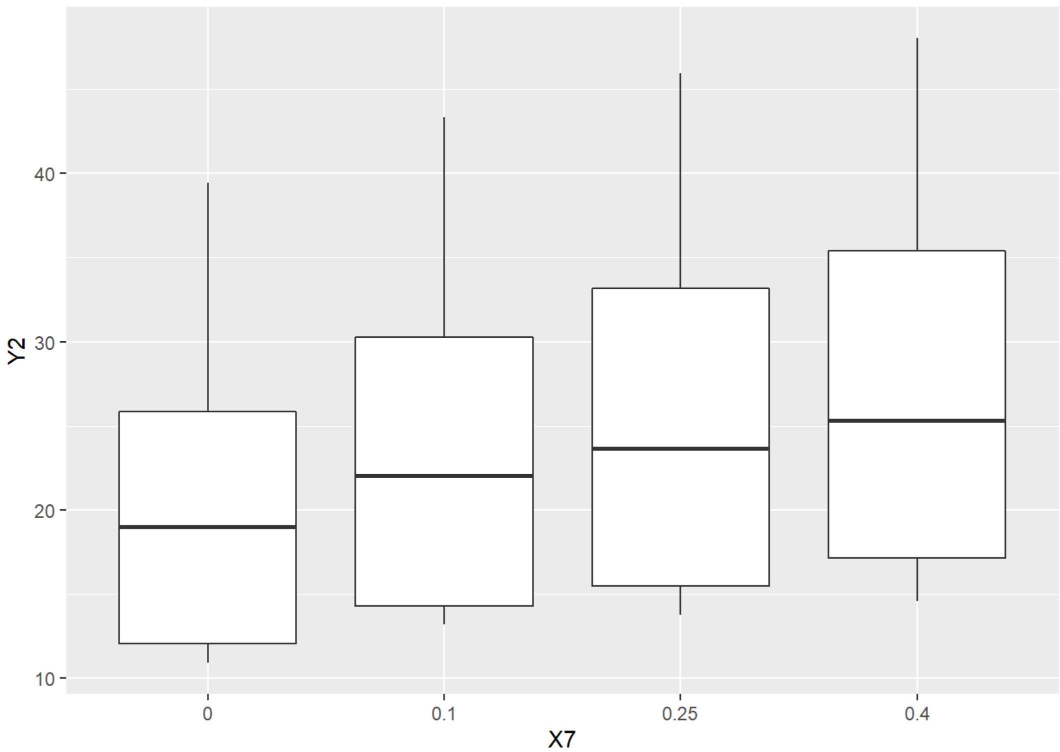
Distribution of X5 w.r.t Y1 where X5 = 7 plays a major role.

```
ggplot(data, aes(X6,Y2)) + geom_violin()
```



This plot shows that X6 has equal and same distrbution over its all four values.

```
ggplot(data, aes(X7,Y2)) + geom_boxplot()
```



This also shows equal distribution over its all 4 values.

Splitting the data

Subsetting and splitting for Y1

```
datay1 <- subset(data,select = -Y2) #select all columns except Y2
set.seed(111)
split <- sample.split(datay1 , SplitRatio = 0.7)
trainy1 <- subset(datay1 ,split == T)
testy1 <- subset(datay1 ,split == F)
```

Subsetting and splitting for Y2

```
datay2 <- subset(data,select = -Y1) #select all columns except Y1
set.seed(111)
split <- sample.split(datay2 , SplitRatio = 0.7)
trainy2 <- subset(datay2 ,split == T)
testy2 <- subset(datay2 ,split == F)
```

Model Making

For Y1

Using linear regression model for Y1 as it is not a categorical variable. Firstly, training the model with all the variables.

```
modely1 <- lm(Y1~X1+X2+X3+X4+X5+X6+X7+X8 , data = trainy1)
summary(modely1)
```

```
##
## Call:
## lm(formula = Y1 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data = trainy1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.935 -1.304 -0.184  1.114  8.129
##
## Coefficients: (3 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.063e+02  2.290e+01   4.640 4.47e-06 ***
## X1          -5.311e+01  1.424e+01  -3.729 0.000214 ***
## X2          -1.378e-01  1.879e-02  -7.333 9.26e-13 ***
## X3           1.213e-01  4.657e-03  26.043  < 2e-16 ***
## X4122.5      2.388e+00  7.419e-01   3.219 0.001369 **
## X4147        4.998e+00  5.909e-01   8.459 3.05e-16 ***
## X4220.5             NA         NA      NA       NA
## X57                 NA         NA      NA       NA
## X63          1.172e+00  3.394e-01   3.451 0.000605 ***
## X64          2.886e-01  3.403e-01   0.848 0.396824
## X65         -8.793e-02  3.380e-01  -0.260 0.794827
## X70.1        6.431e+00  5.857e-01  10.980  < 2e-16 ***
## X70.25       8.705e+00  5.847e-01  14.887  < 2e-16 ***
## X70.4        1.143e+01  5.863e-01  19.502  < 2e-16 ***
## X81          3.136e-01  3.902e-01   0.804 0.422020
## X82          2.686e-01  3.902e-01   0.688 0.491470
## X83          9.053e-04  3.902e-01   0.002 0.998150
## X84          9.692e-02  3.902e-01   0.248 0.803950
## X85                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.703 on 495 degrees of freedom
## Multiple R-squared:  0.9296, Adjusted R-squared:  0.9274
## F-statistic: 435.5 on 15 and 495 DF,  p-value: < 2.2e-16
```

From the summary it is clear that for X8 the model is not confident so we can remove them for better accuracy. Also the singular values has been taken care by the lm() function itself.

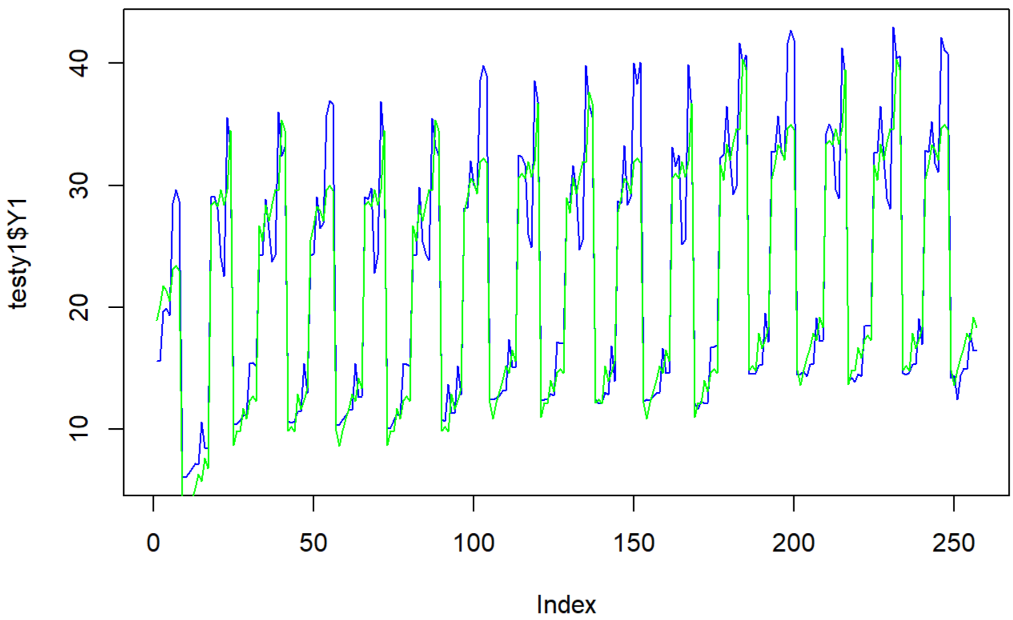
```
modely1 <- lm(Y1~X1+X2+X3+X4+X5+X6+X7 , data = trainy1)
summary(modely1)
```

```
##
## Call:
## lm(formula = Y1 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7, data = trainy1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9731 -1.2938 -0.1948  1.0732  8.0905
##
## Coefficients: (2 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 106.254426  22.834886   4.653 4.19e-06 ***
## X1          -53.108747  14.199858  -3.740 0.000205 ***
## X2           -0.137816   0.018740  -7.354 7.95e-13 ***
## X3            0.121290   0.004644  26.117  < 2e-16 ***
## X4122.5       2.388378   0.739728   3.229 0.001325 **
## X4147         4.998139   0.589169   8.483 2.50e-16 ***
## X4220.5             NA         NA      NA       NA
## X57                 NA         NA      NA       NA
## X63           1.171534   0.338465   3.461 0.000584 ***
## X64           0.288607   0.339346   0.850 0.395465
## X65          -0.087933   0.336994  -0.261 0.794252
## X70.1         6.568309   0.529419  12.407  < 2e-16 ***
## X70.25        8.839862   0.529470  16.696  < 2e-16 ***
## X70.4        11.569394   0.529599  21.846  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.696 on 499 degrees of freedom
## Multiple R-squared:  0.9294, Adjusted R-squared:  0.9278
## F-statistic: 597.1 on 11 and 499 DF,  p-value: < 2.2e-16
```

The accuracy of model has no significant effect but a little improvement in accuracy so we can consider this.

```
predy1 <- predict(modely1 , newdata = testy1) #predicton of values

plot(testy1$Y1 , type ="l" , col="blue") #actual test values in blue color
lines(predy1 , col="green") #predicted test values in green color
```



For Y2

Using linear regression model for Y2 as it is not a categorical variable. Firstly, training the model with all the variables.

```r
modely2 <- lm(Y2~X1+X2+X3+X4+X5+X6+X7+X8 , data = trainy2)
summary(modely2)
```

```
##
## Call:
## lm(formula = Y2 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data = trainy2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7292 -1.6205 -0.2893  1.2383 12.1313
##
## Coefficients: (3 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 114.016196  25.406997   4.488 8.97e-06 ***
## X1          -53.556281  15.799324  -3.390 0.000755 ***
## X2           -0.132571   0.020851  -6.358 4.65e-10 ***
## X3            0.103225   0.005167  19.977  < 2e-16 ***
## X4122.5       2.940177   0.823050   3.572 0.000388 ***
## X4147         5.417255   0.655533   8.264 1.30e-15 ***
## X4220.5             NA         NA      NA       NA
## X57                 NA         NA      NA       NA
## X63           0.904850   0.376589   2.403 0.016639 *
## X64           0.417225   0.377570   1.105 0.269684
## X65           0.251892   0.374953   0.672 0.502027
## X70.1         3.478519   0.649768   5.353 1.32e-07 ***
## X70.25        5.295389   0.648741   8.163 2.74e-15 ***
## X70.4         7.350671   0.650497  11.300  < 2e-16 ***
## X81           0.330225   0.432950   0.763 0.445987
## X82           0.209271   0.432908   0.483 0.629020
## X83           0.091173   0.432951   0.211 0.833298
## X84           0.416683   0.432950   0.962 0.336305
## X85                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.999 on 495 degrees of freedom
## Multiple R-squared:  0.901,  Adjusted R-squared:  0.898
## F-statistic: 300.4 on 15 and 495 DF,  p-value: < 2.2e-16
```

From the summary it is clear that for X8 the model is not confident so we can remove them for better accuracy. Also the singular values has been taken care by the lm() function itself.

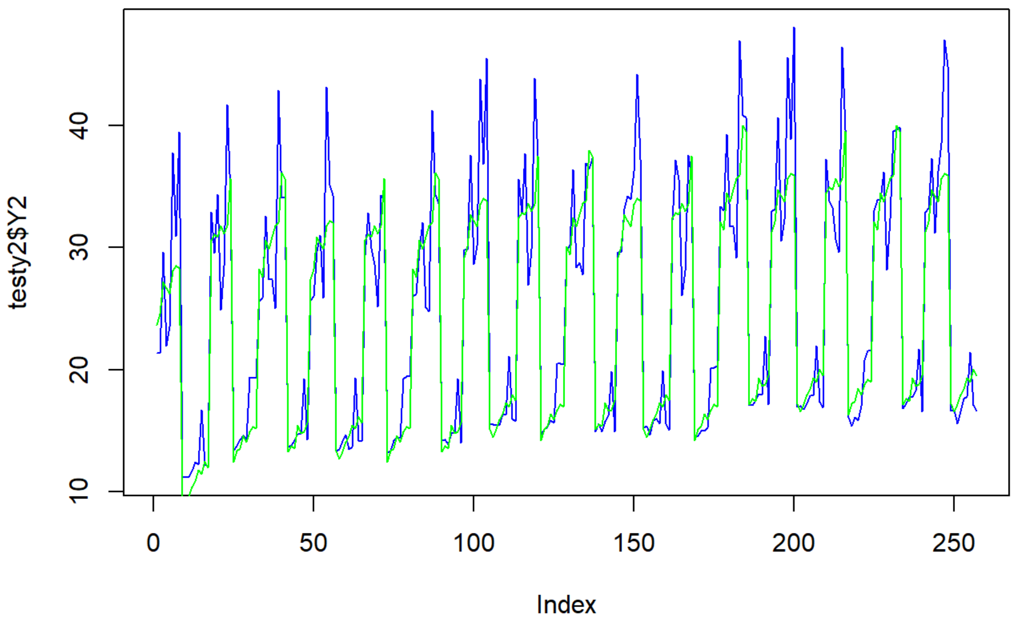
```r
modely2 <- lm(Y2~X1+X2+X3+X4+X5+X6+X7 , data = trainy2)
summary(modely2)
```

```
##
## Call:
## lm(formula = Y2 ~ X1 + X2 + X3 + X4 + X5 + X6 + X7, data = trainy2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.730 -1.606 -0.307  1.251 12.134
##
## Coefficients: (2 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 114.022647  25.336407   4.500 8.45e-06 ***
## X1          -53.560326  15.755428  -3.399 0.000729 ***
## X2           -0.132577   0.020793  -6.376 4.15e-10 ***
## X3            0.103226   0.005153  20.033  < 2e-16 ***
## X4122.5       2.940037   0.820764   3.582 0.000374 ***
## X4147         5.417092   0.653711   8.287 1.08e-15 ***
## X4220.5             NA         NA      NA       NA
## X57                 NA         NA      NA       NA
## X63           0.904835   0.375543   2.409 0.016340 *
## X64           0.417225   0.376521   1.108 0.268350
## X65           0.251891   0.373911   0.674 0.500836
## X70.1         3.690768   0.587415   6.283 7.24e-10 ***
## X70.25        5.501504   0.587472   9.365  < 2e-16 ***
## X70.4         7.560722   0.587616  12.867  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.991 on 499 degrees of freedom
## Multiple R-squared:  0.9008, Adjusted R-squared:  0.8986
## F-statistic: 411.8 on 11 and 499 DF,  p-value: < 2.2e-16
```

The accuracy of model has no significant effect but a little improvement in accuracy so we can consider this.

```
predy2 <- predict(modely2 , newdata = testy2 ) #predicton of values

plot(testy2$Y2 , type ="l" , col="blue") #actual test values in blue color
lines(predy2 , col="green") #predicted test values in green color
```



Calculation of Root Mean Square Error (RMSE)

For Y1

The RMSE for testing dataset is

```
 rmse(testy1$Y1 , predy1)
```

```
## [1] 3.128755
```

The RMSE fo training dataset is

```
predicttesty1 <- predict(modely1 ,data = testy1 )
rmse(trainy1$Y1 , predicttesty1)
```

```
## [1] 2.6638
```

For Y2

The RMSE for testing dataset is

```
rmse(testy2$Y2 , predy2)
```

```
## [1] 3.64019
```

The RMSE fo training dataset is

```
predicttesty2 <- predict(modely2 ,data = testy2 )
rmse(trainy2$Y2 , predicttesty2)
```

```
## [1] 2.955615
```

Result

We have used Linear regression model to predict values of Y1 and Y2 with the RMSE as follow:-

Variable	RMSE Testing dataset	RMSE Training dataset	Absolute Difference in RMSE
Y1	3.1287	2.6638	0.4649
Y2	3.6401	2.9556	0.6845

Since the RMSE error difference is very negligible so our model is trained well.