# Lab 1 & 2: Getting started with R

## Problem statement:

- Perform basic mathematics operations on variable such as (addition/substation/multiplication division/power/modulo operator). Also perform relational and logical operations.
- Create vector and perform various operations.
- Create numerical vector.
- Operation between scalar and vector.
- Operations between vectors.
- Text variable and vector.
- Create matrices.
- Manage the workspace (setw() and getw()).
- Understand data data frames and list.
- How to import and export the data.
- How to read and write CSV/text files.
- Installation and loading packages.
- Perform various loop and conditional statement.
- How function works in R.
- Practice plotting and charting.
- Understand the factor variable.

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Enrollment Number: 15100007
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 1 & 2
#Task: Getting started with R

#Perform basic mathematics operations on variable such as (addition/substation/multiplication division/power/modulo operator). Also perform relational and logical operations.
x=12
y=6
12+6
12-6
12*6
12/6
12**6
12%%6
x<y
x>y
x == 12
x != 5
x|y
x&y

#Create vector and perform various operations
#Create numerical vector
```

```
x <- c(2,8,3)
y <- c(6,6,1)

#Operation between scalar and vector
x <- c(2,8,3)
z = 1
x + z
x - z

#Operations between vectors
x+y
x>y

#Text variable and vector
s = c ('p','q','r')
length(s)
nchar(s)
t = c(n,s)
t

#Create matrices
B = matrix (c(1,2,3,4,5,6),nrow=3,ncol=2)
B

#Manage the workspace (setw() and getw())
getwd()
setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
getwd()

#Understand data data frames and list.
n = c(1, 2, 3)
s = c("abc","def","ghi")
df = data.frame(n,s)
df

n = c(1, 2, 3)
s = c("abc","def","ghi")
l = list(n,s)
l

#How to import and export the data.
testdata <- read.table ("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab/drug2.csv", header=TRUE,
sep=",")

#How to read and write CSV/text files.
drug = read.csv("drug2.csv")
head(drug)
write.csv(drug, file = "drug3.csv")

#Installation and loading packages.
install.packages("e1071")
```

```
library(e1071)

#Perform various loop and conditional statement.
for(i in 1:10) {
 print(i)
 }


#How function works in R.
add <- function(x,y){
 x= 10
 y= 1
 result <-x+y
 print(result)
}
add()

#Practice plotting and charting.
val <- c(1, 3, 6, 4, 9)
plot(val)

#Understand the factor variable.
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
fdata
```

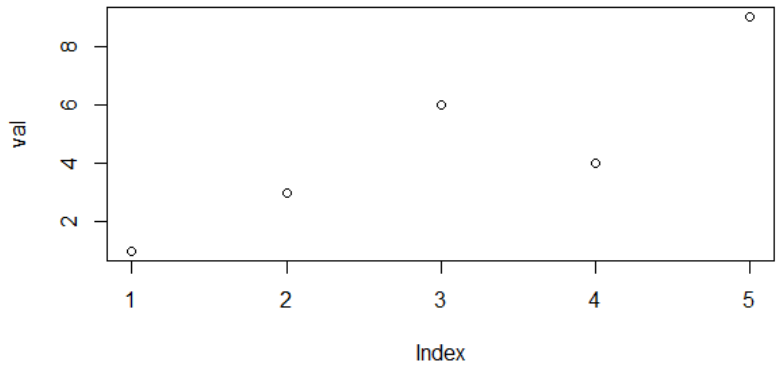
## Output:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Enrollment Number: 15100007
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 1 & 2
#Task: Introduction to R

> #Perform basic mathematics operations on variable such as (addition/substation/multiplication division/power/modulo operator). Also #perform relational and logical operations.
> x=12
> y=6
> 12+6
[1] 18
>
> 12-6
[1] 6
>
> 12*6
[1] 72
>
> 12/6
[1] 2
>
```

```
> 12**6
[1] 2985984
>
> 12%%6
[1] 0
>
> x<y
[1] FALSE
>
> x>y
[1] TRUE
>
> x == 12
[1] TRUE
>
> x != 5
[1] TRUE
>
> x|y
[1] TRUE
>
> x&y
[1] TRUE
>
> #Create vector and perform various operations
> #Create numerical vector
> x <- c(2,8,3)
> y <- c(6,6,1)
>
> #Operation between scalar and vector
> x <- c(2,8,3)
> z = 1
> x + z
[1] 3 9 4
>
> x - z
[1] 1 7 2
>
> #Operations between vectors
> x+y
[1]  8 14  4
>
> x>y
[1] FALSE  TRUE  TRUE
>
> #Text variable and vector
> s = c ('p','q','r')
> length(s)
[1] 3
>
> nchar(s)
[1] 1 1 1
> #Create matrices
> B = matrix (c(1,2,3,4,5,6),nrow=3,ncol=2)
> B
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
>
> #Manage the workspace (setw() and getw())
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab"
>
> #Understand data data frames and list.
> n = c(1, 2, 3)
> s = c("abc","def","ghi")
> df = data.frame(n,s)
> df
  n   s
1 1 abc
2 2 def
3 3 ghi
>
> n = c(1, 2, 3)
> s = c("abc","def","ghi")
> l = list(n,s)
> l
[[1]]
[1] 1 2 3

[[2]]
[1] "abc" "def" "ghi"

> #How to import and export the data.
> testdata <- read.table ("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab/drug2.csv", header=T
RUE, sep=",")

> #How to read and write CSV/text files.
> drug = read.csv("drug2.csv")
> head(drug)
  sex dose response
1   1  0.1    13.75
2   1  0.2    12.90
3   1  0.3    19.26
4   1  0.4    20.34
5   1  0.5    19.97
6   1  0.6    26.80
> write.csv(drug, file = "drug3.csv")
>
> #Installation and loading packages.
> install.packages("e1071")
Installing package into 'C:/Users/Ashish Upadhyay/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/e1071_1.6-8.zip'
Content type 'application/zip' length 894861 bytes (873 KB)
downloaded 873 KB
```

```
package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Ashish Upadhyay\AppData\Local\Temp\RtmpYV25hi\downloaded_packages
> library(e1071)
Warning message:
package 'e1071' was built under R version 3.4.4
>
> #Perform various loop and conditional statement.
> for(i in 1:10) {
+   print(i)
+   }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
>
> #How function works in R.
> add <- function(x,y){
+   x= 10
+   y= 1
+   result <-x+y
+   print(result)
+ }
> add()
[1] 11
>
> #Practice plotting and charting.
> val <- c(1, 3, 6, 4, 9)
> plot(val)
```



```
> #Understand the factor variable.
> data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
> fdata = factor(data)
> fdata
 [1] 1 2 2 3 1 2 3 3 1 2 3 3 1
Levels: 1 2 3
```

# Lab 3: R Functions

## Problem statement:

- Measures of central tendency
- Mean, Median, Trimmed mean, Mode etc.
- Measures of variability.
- Range, Percentile, Interquartile range, variance, standard deviation, mean absolute deviation, Median absolute deviation.

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Enrollment Number: 15100007
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 3
#Task: R functions

#Creating a vector
lst<-c(2,5,7,8,4,8,2,3,9,5,6,4,3,2,2,2)

#Mean, Trimmed mean, Median and Mode calculation
mean(lst)
median(lst)
mean(lst, trim=0.40)
mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
mode(lst)

#Other mathematical functions
range(lst)
quantile(lst)
IQR(lst)
var(lst)
sd(lst)
mad(lst,center = mean(lst))
mad(lst)
```

**Output:**

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Enrollment Number: 15100007
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 3
> #Task: R functions
>
> #Creating a vector
> lst<-c(2,5,7,8,4,8,2,3,9,5,6,4,3,2,2,2)
>
> #Mean, Trimmed mean, Median and Mode calculation
> mean(lst)
[1] 4.5
> median(lst)
[1] 4
> mean(lst, trim=0.40)
[1] 4
> mode <- function(v) {
+   uniqv <- unique(v)
+   uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> mode(lst)
[1] 2
>
> #Other mathematical functions
> range(lst)
[1] 2 9
> quantile(lst)
  0%  25%  50%  75% 100%
2.00 2.00 4.00 6.25 9.00
> IQR(lst)
[1] 4.25
> var(lst)
[1] 6
> sd(lst)
[1] 2.44949
> mad(lst,center = mean(lst))
[1] 3.7065
> mad(lst)
[1] 2.9652
```

# Lab 4: Linear Regression

## Problem statement:

Develop linear regression model (through least square method) on given data set (drug2.csv) as:

- Create a model (A) (Simple linear regression) to predict response with respect to dose.
- Interpret the model summary
- Draw residuals of model to see the normal distribution.
- Improve model (B) by adding more feature (sex) and again investigate residuals graph.
- Validate your model by performing tests (fitted value vs residuals, fitted values vs actual
- Further improve model (C) through moderation i.e. interaction variable and validate model through aforesaid procedure.
- Calculate the RMSE for all models (A, B, C) and represent as a bar chart/histogram.
- Calculate the standard deviation of residuals of all models (A, B, C) and represent as a bar chart/ histogram.

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 4
#Task: Linear Regression Implementation

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
getwd()
drug = read.csv("drug2.csv")
head(drug)
attach(drug)

#Model A
model1 = lm(response~dose)
summary(model1)
err1 = residuals(model1)
hist(err1)
plot(model1$fitted.values,err1)

#Model B
model2 = lm(response~dose+sex)
summary(model2)
err2 = residuals(model2)
hist(err2)
plot(model2$fitted.values,err2)

#Model C - moderation
product = drug$sex * drug$dose
model3 = lm(drug$response~drug$dose+product+drug$sex)
summary(model3)
err3 = residuals(model3)
hist(err3)
plot(model3$fitted.values,err3)
```

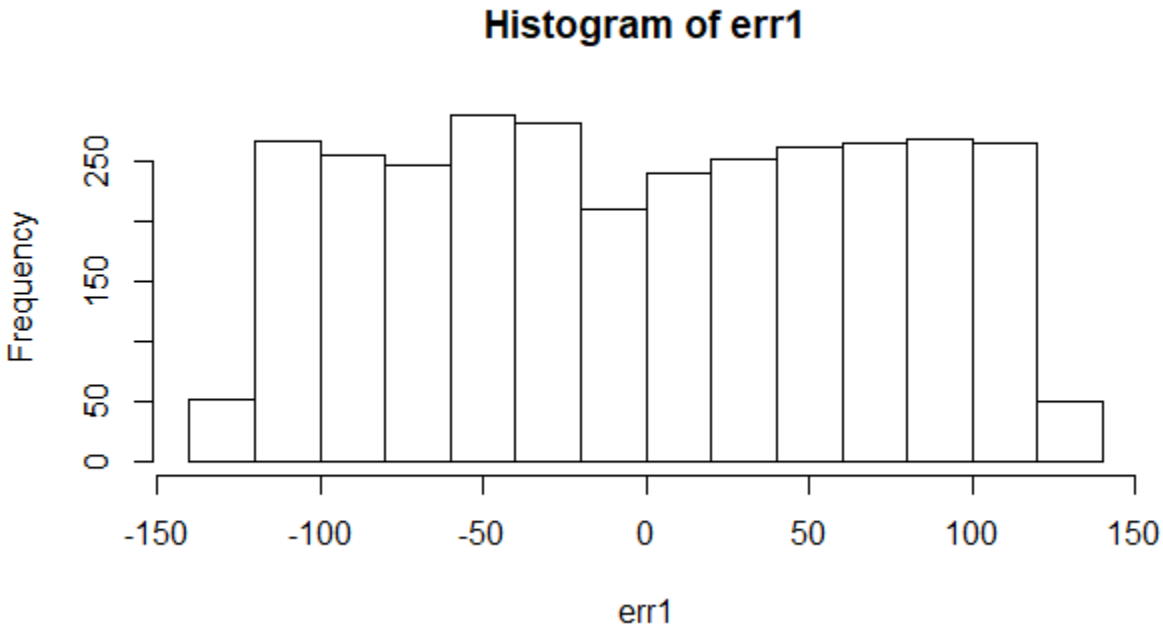
```
plot(model3$fitted.values,drug$response)

#RMSE and Strandard Deviation
pred=predict(model3, drug)
actual= drug$response
diff= actual-pred
head(diff)
rmse= sqrt(sum(diff**2)/nrow(drug))
rmse
err4=residuals(model3)
rmse2= sqrt(sum(err4**2)/nrow(drug))
rmse2
```

## Output:

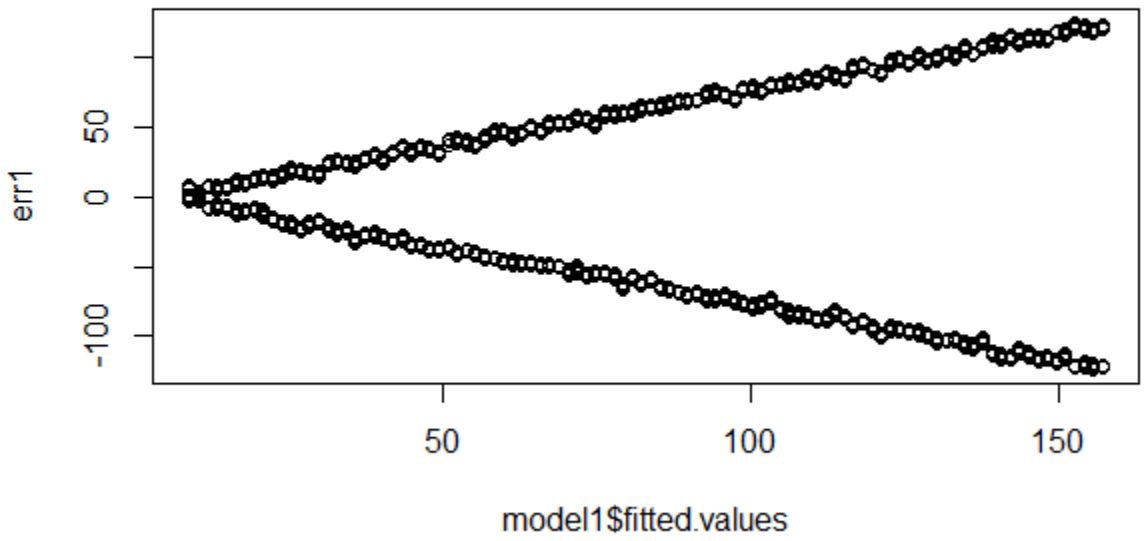
```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 4
> #Task: Linear Regression Implementation
>
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab"
> drug = read.csv("drug2.csv")
> head(drug)
  sex dose response
1   1  0.1    13.75
2   1  0.2    12.90
3   1  0.3    19.26
4   1  0.4    20.34
5   1  0.5    19.97
6   1  0.6    26.80
> attach(drug)
>
> #Model A
> model1 = lm(response~dose)
> summary(model1)

Call:
lm(formula = response ~ dose)

Residuals:
    Min      1Q  Median      3Q     Max
-123.514  -62.764    0.401   63.669  124.707

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.2534     2.5778   2.814  0.00493 **
dose         15.0020     0.4432  33.852  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 72.36 on 3198 degrees of freedom
Multiple R-squared: 0.2638, Adjusted R-squared: 0.2636
F-statistic: 1146 on 1 and 3198 DF, p-value: < 2.2e-16

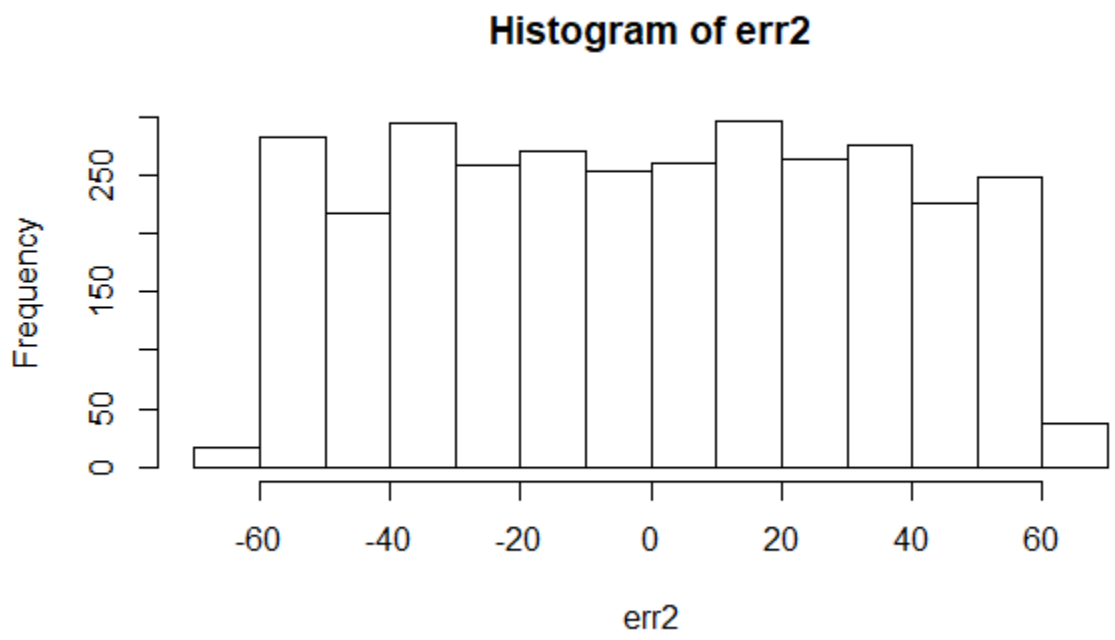
```
> err1 = residuals(model1)
> hist(err1)
```



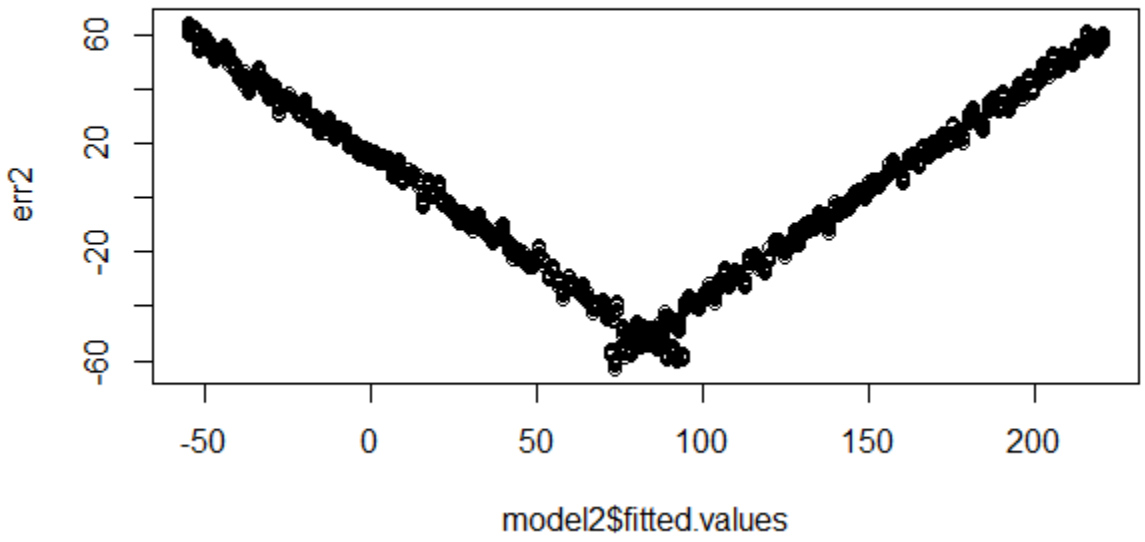

```
> plot(model1$fitted.values,err1)
```



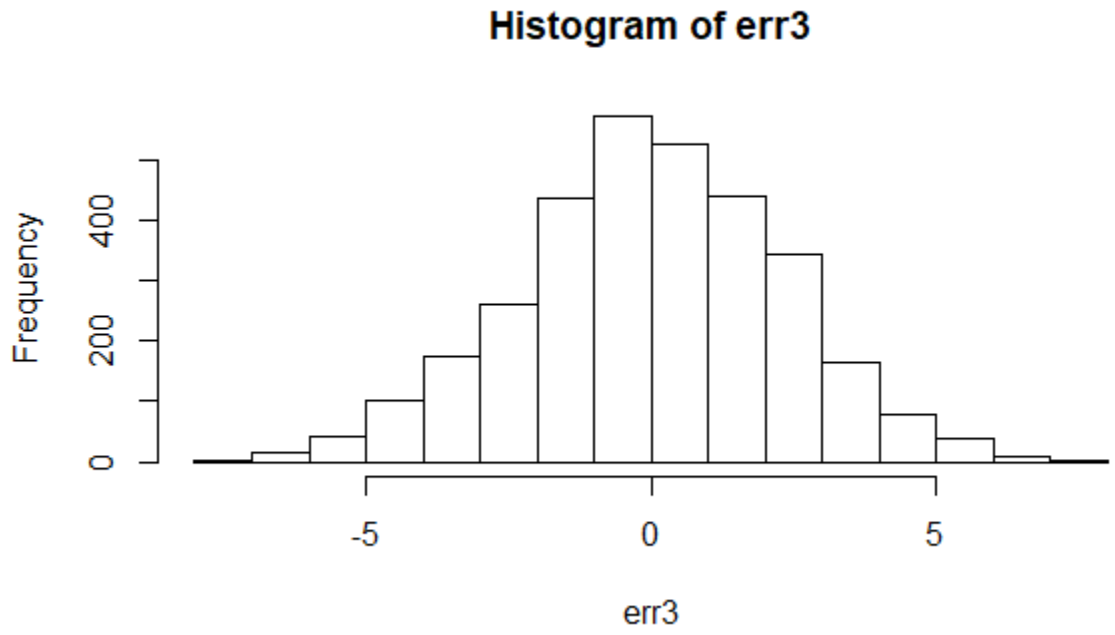
```
> #Model B
> model2 = lm(response~dose+sex)
> summary(model2)

Call:
lm(formula = response ~ dose + sex)

Residuals:
   Min     1Q  Median     3Q    Max
-62.986 -30.350   0.306  29.360  64.009

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -56.1189     1.3881  -40.43   <2e-16 ***
dose         15.0020     0.2138   70.18   <2e-16 ***
sex         126.7445     1.2341  102.70   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 34.91 on 3197 degrees of freedom
Multiple R-squared:  0.8288,      Adjusted R-squared:  0.8286
F-statistic:  7736 on 2 and 3197 DF,  p-value: < 2.2e-16

> err2 = residuals(model2)
> hist(err2)
```



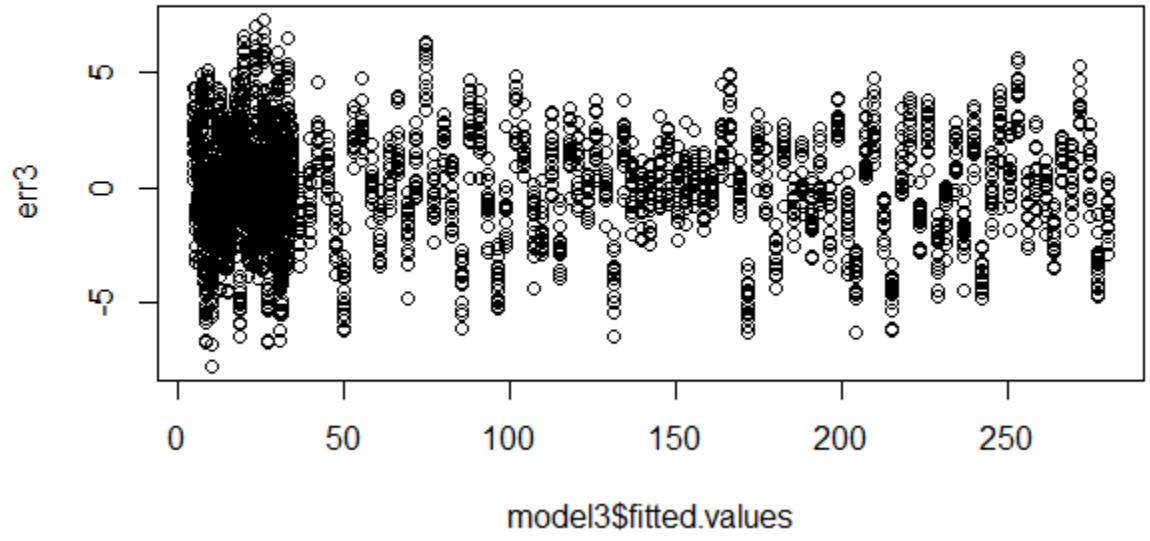

```
> plot(model2$fitted.values,err2)
```



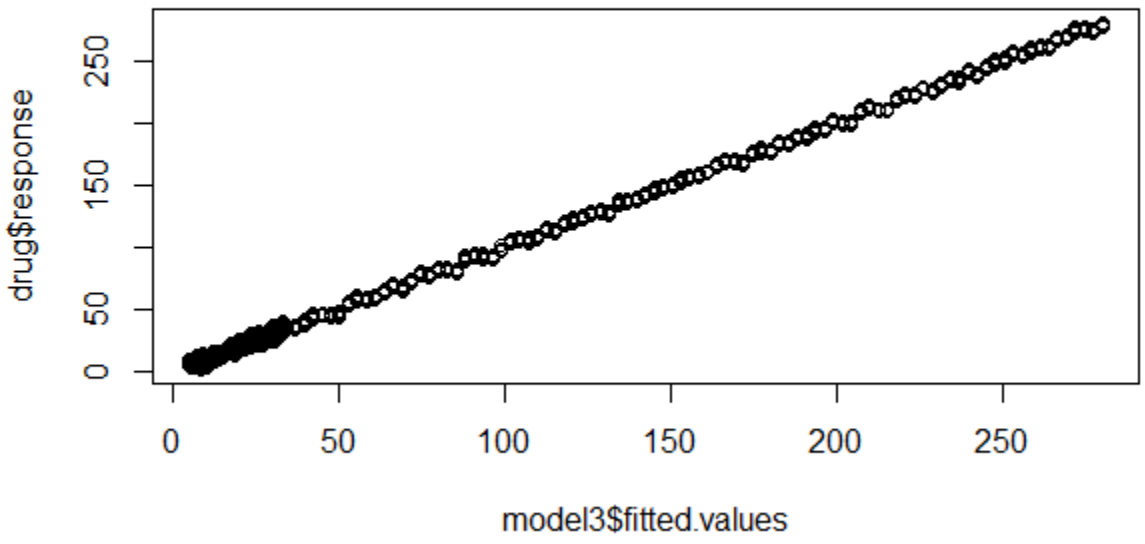

```
> #Model C - moderation
> product = drug$sex * drug$dose
> model3 = lm(drug$response~drug$dose+product+drug$sex)
> summary(model3)

Call:
lm(formula = drug$response ~ drug$dose + product + drug$sex)

Residuals:
   Min     1Q  Median     3Q    Max
-7.6950 -1.4668 -0.0004  1.5996  7.2181

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.78574    0.11658   41.05   <2e-16 ***
drug$dose    2.94171    0.02004  146.77   <2e-16 ***
product     24.12064    0.02834  850.98   <2e-16 ***
drug$sex     4.93530    0.16487   29.93   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.314 on 3196 degrees of freedom
Multiple R-squared:  0.9992,      Adjusted R-squared:  0.9992
F-statistic: 1.415e+06 on 3 and 3196 DF,  p-value: < 2.2e-16

> err3 = residuals(model3)
> hist(err3)
```




```
> plot(model3$fitted.values,err3)
```




```
> plot(model3$fitted.values,drug$response)
```




```
> #RMSE and Strandard Deviation
> pred=predict(model3, drug)
> actual= drug$response
> diff= actual-pred
> head(diff)
       1        2        3        4        5        6
 1.3227276 -2.2335081  1.4202563 -0.2059794 -3.2822150  0.8415493
> rmse= sqrt(sum(diff**2)/nrow(drug))
> rmse
[1] 2.312749
> err4=residuals(model3)
> rmse2= sqrt(sum(err4**2)/nrow(drug))
> rmse2
[1] 2.312749
```

# Lab 5: Polynomial Regression

## Problem statement:

Develop linear regression model (through least square method) where "dmf" as dependent variable with respect to various combination of input variable (flor) on given data set (dmf.csv) as:

- Model A (Dependent variable- flor)
- Model B (Dependent variable- flor, square(flor))
- Model C (Dependent variable- flor, square(flor), 1/sqrt(flor))
- Calculate the RMSE for all models (A, B, C) and represent as a bar chart/histogram.
- Calculate the standard deviation of residuals of all models (A, B, C) and represent as a bar chart/ histogram.
- Validate all three models (A, B, C) through various tests and rank models according to efficiency.
- Perform feature engineering through both forward and backward selection methods.
- Declare most perfect model with justification.

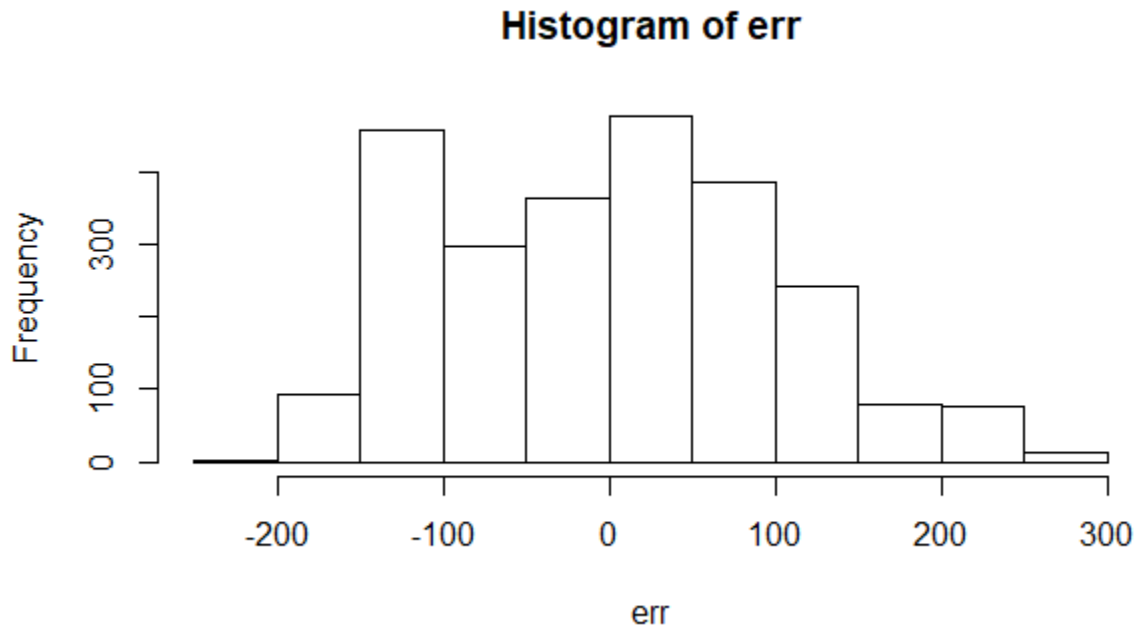
## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 5
#Task: Polynomial Regression Implementation

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
getwd()

dmf = read.csv("dmf.csv")
attach(dmf)

# Basic model: Only flor
model= lm(dmf$dmf~dmf$flor)
summary (model)
err= residuals (model)
hist(err)
plot(model$fitted.values,model$residuals)
plot(model$fitted.values, dmf$dmf)
flor2= dmf$flor^2

# Advance model: flor + flor^2
model2=lm(dmf$dmf~ dmf$flor+flor2)
summary(model2)
err2= residuals(model2)
hist(err2)
plot(model2$fitted.values,model2$residuals)
plot(model2$fitted.values,dmf$dmf)

# More advance model: flor + flor^2 + sqrt(flor)
model3 = lm(dmf$dmf~ dmf$flor+flor2+1/sqrt(flor))
summary(model3)
err3= residuals(model3)
```

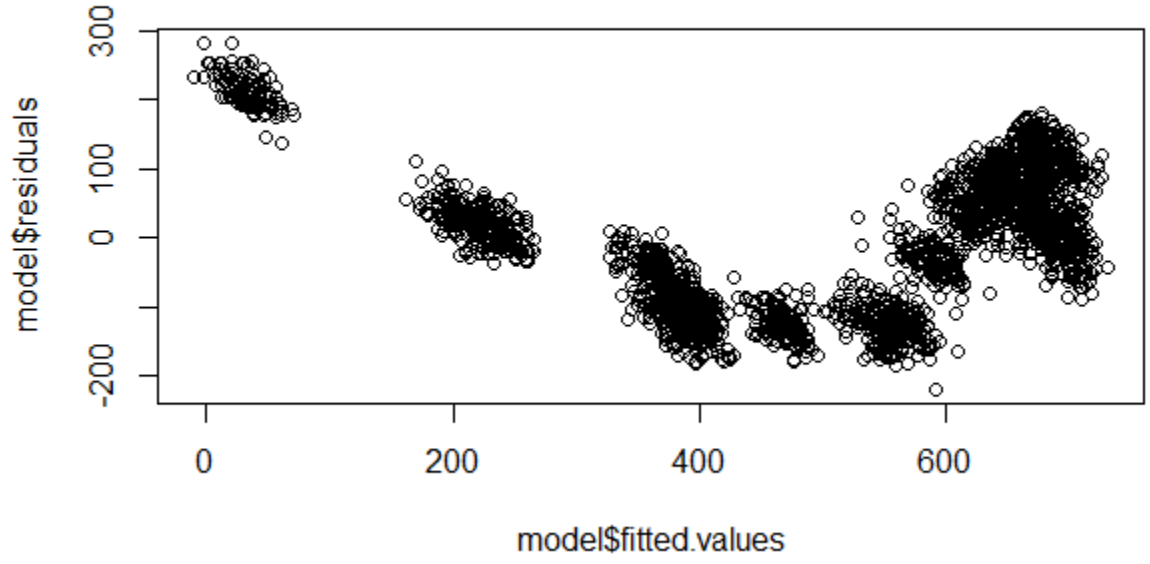
```
hist(err3)
plot(model3$fitted.values,model3$residuals)
plot(model3$fitted.values,dmf$dmf)
```

## Output:

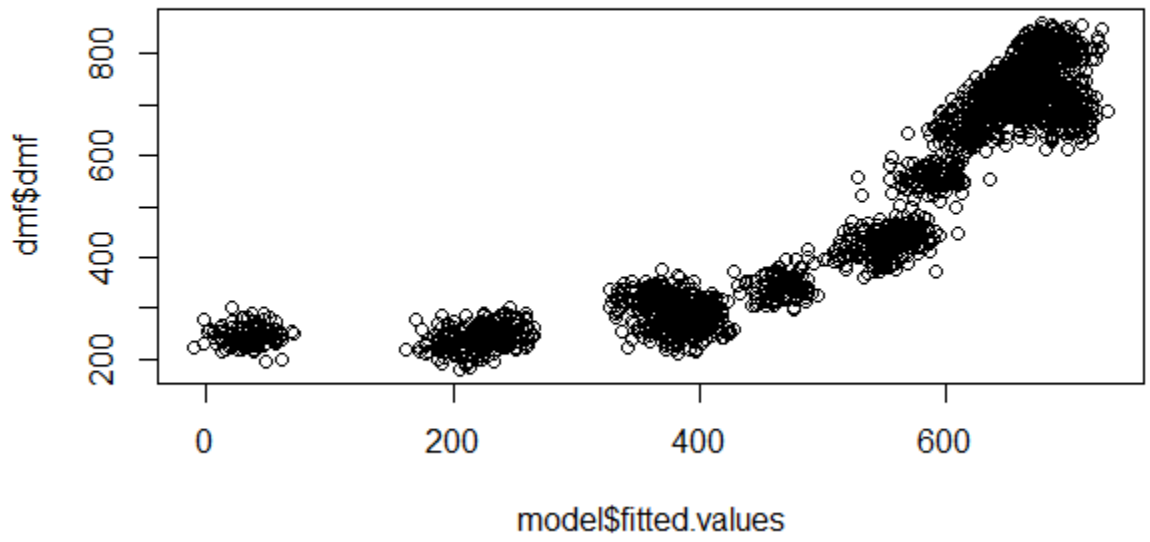
```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 4
> #Task: Ploynomial Regression Implementation
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab"
>
> dmf = read.csv("dmf.csv")
> attach(dmf)
>
> # Basic model: Only flor
> model=lm(dmf$dmf~dmf$flor)
> summary (model)

Call:
lm(formula = dmf$dmf ~ dmf$flor)

Residuals:
    Min      1Q  Median      3Q     Max
-217.943  -91.930    3.935   70.097  281.904

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  730.929     3.158   231.5   <2e-16 ***
dmf$flor    -252.701     2.700   -93.6   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 99.79 on 2479 degrees of freedom
Multiple R-squared:  0.7794,      Adjusted R-squared:  0.7793
F-statistic:  8760 on 1 and 2479 DF,  p-value: < 2.2e-16

> err= residuals (model)
> hist(err)
```



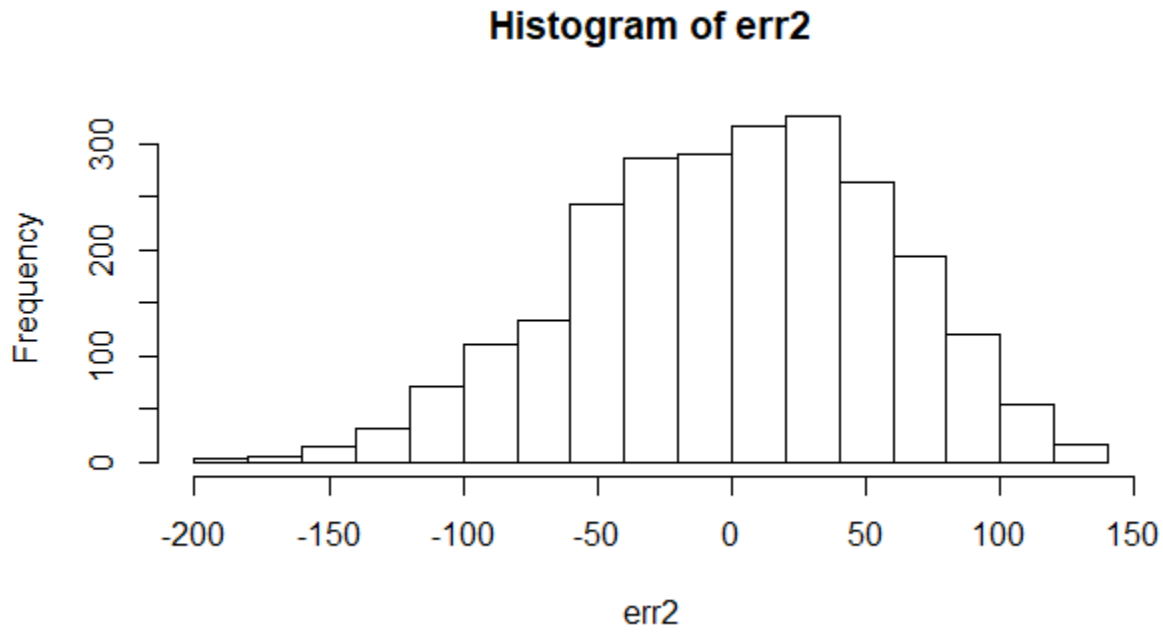

```
> plot(model$fitted.values,model$residuals)
```



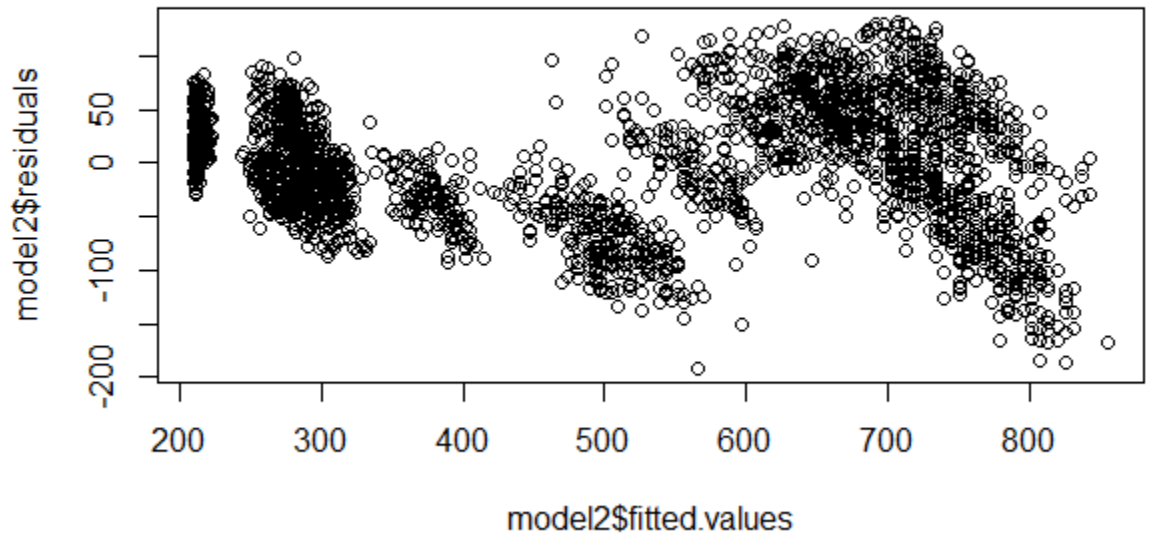

```
> plot(model$fitted.values, dmf$dmf)
```



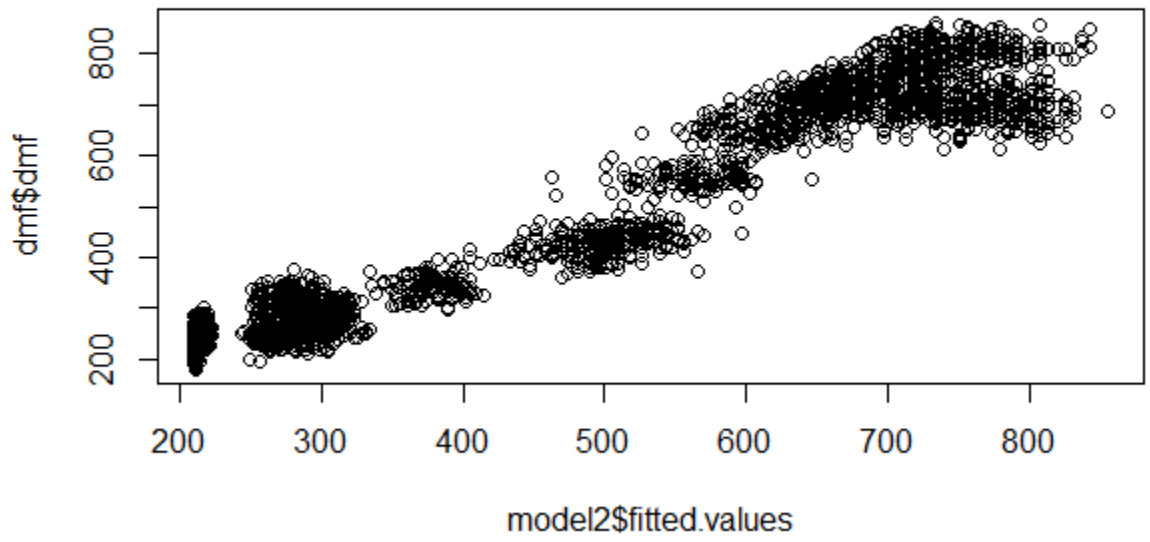

```
> # Advance model: flor + flor^2
> flor2= dmf$flor^2
> model2=lm(dmf$dmf~ dmf$flor+flor2)
> summary(model2)

Call:
lm(formula = dmf$dmf ~ dmf$flor + flor2)

Residuals:
    Min      1Q  Median      3Q     Max
-191.388  -39.457    2.797   42.347  131.543

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  855.126     2.533  337.55   <2e-16 ***
dmf$flor    -604.861     5.231 -115.64   <2e-16 ***
flor2        141.939     2.013   70.53   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57.56 on 2478 degrees of freedom
Multiple R-squared:  0.9267,      Adjusted R-squared:  0.9266
F-statistic: 1.565e+04 on 2 and 2478 DF,  p-value: < 2.2e-16

> err2= residuals(model2)
> hist(err2)
```



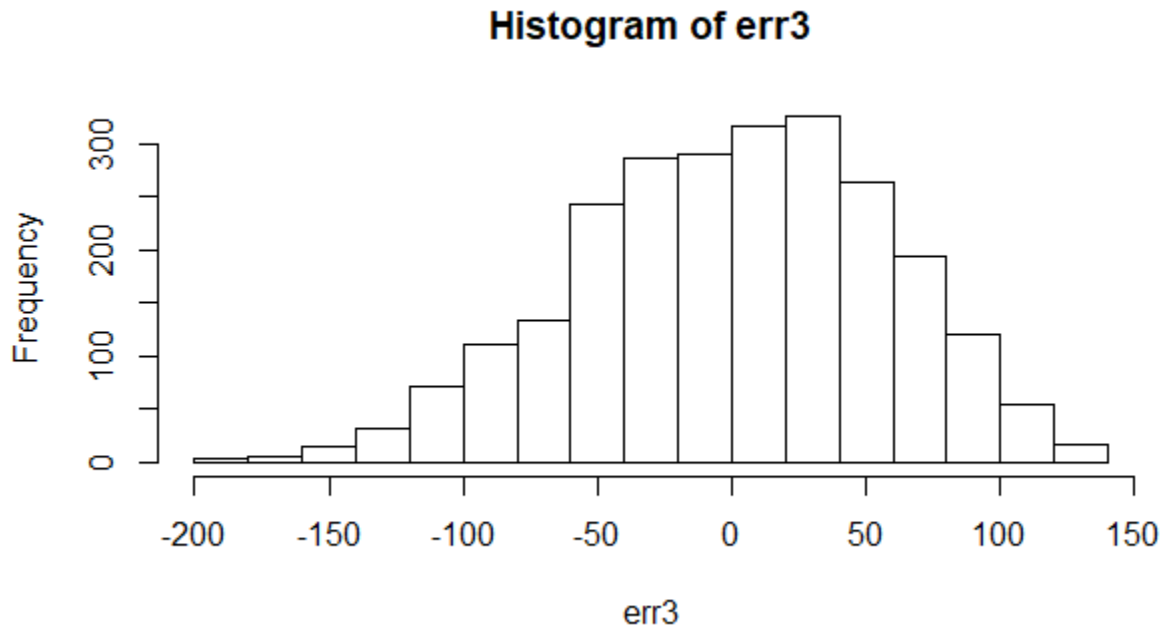

```
> plot(model2$fitted.values,model2$residuals)
```



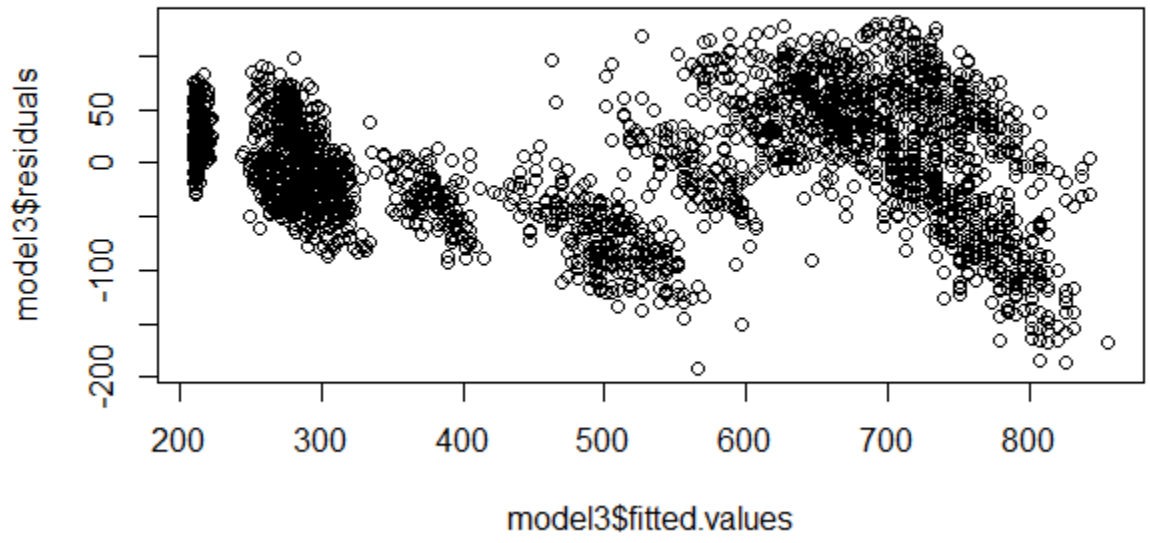

```
> plot(model2$fitted.values,dmf$dmf)
```



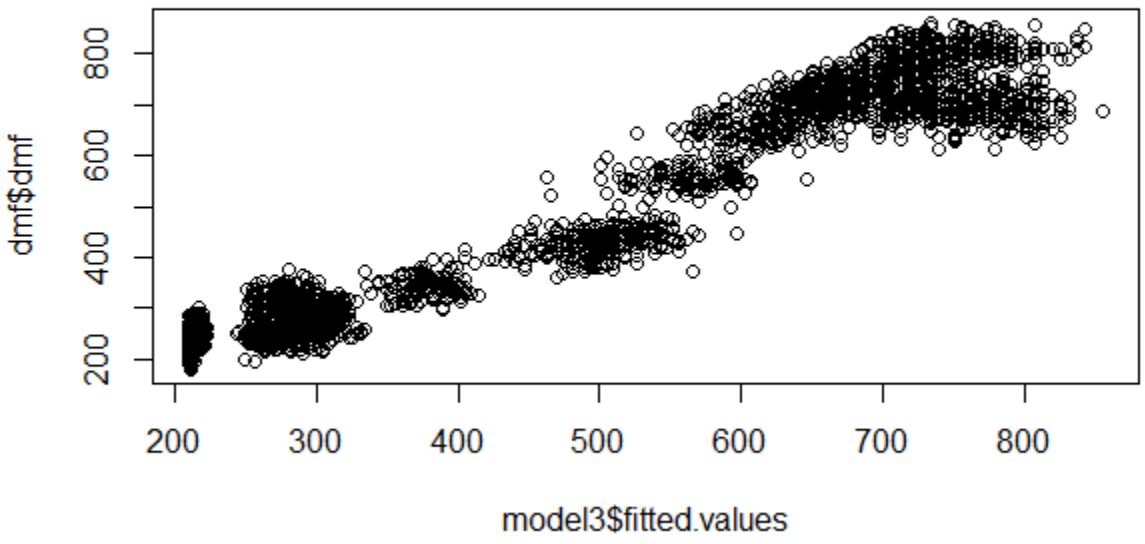

```
>
> # More advance model: flor + flor^2 + sqrt(flor)
> model3 = lm(dmf$dmf~ dmf$flor+flor2+1/sqrt(flor))
> summary(model3)

Call:
lm(formula = dmf$dmf ~ dmf$flor + flor2 + 1/sqrt(flor))

Residuals:
    Min      1Q  Median      3Q     Max
-191.388  -39.457    2.797   42.347  131.543

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  855.126     2.533  337.55   <2e-16 ***
dmf$flor    -604.861     5.231 -115.64   <2e-16 ***
flor2        141.939     2.013   70.53   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57.56 on 2478 degrees of freedom
Multiple R-squared:  0.9267,      Adjusted R-squared:  0.9266
F-statistic: 1.565e+04 on 2 and 2478 DF,  p-value: < 2.2e-16

> err3= residuals(model3)
> hist(err3)
```




```
> plot(model3$fitted.values,model3$residuals)
```




```
> plot(model3$fitted.values,dmf$dmf)
```


dmf$dmf
200
400
600
800
200
300
400
500
600
700
800
model3$fitted.values

# Lab 6: Linear Regression Validation

## Problem statement:

Calculate following parameter of Simple Liner Regression (drug2.csv) by implementing the formula from scratch.

- R2
- Adjusted- R2
- Residual standard error
- S-value (for slope only)
- P-value (for slope only)

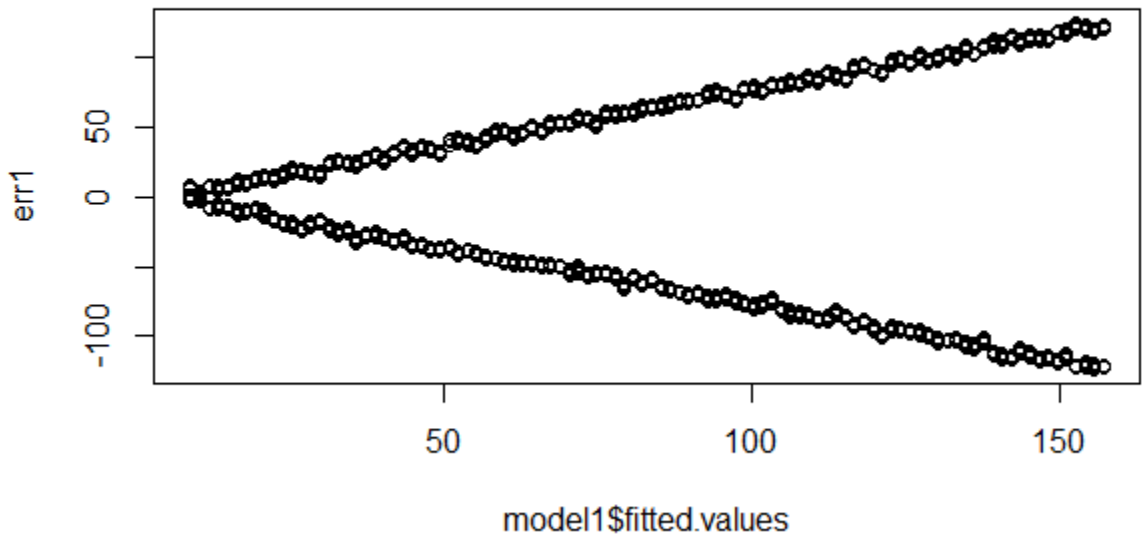
## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 6
#Task: Linear Regression Validation

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()

drug = read.csv("drug2.csv")
attach(drug)
head(drug)
nrow(drug)

model1 = lm(response~dose)
summary(model1)
err1 = residuals(model1)
plot(model1$fitted.values, err1)
hist(err1)
plot(model1$fitted.values, drug$response)
mean <- mean(response)
sst <- sum((response-mean)**2)
sse <- sum((response-model1$fitted.values)**2)
rsq <- 1 - (sse/sst)
fuv <- sse/sst
fuv
rsq
a <- (nrow(drug)-1)*(1-rsq)
b <- nrow(drug)-2
rad <- 1-(a/b)
rad
```

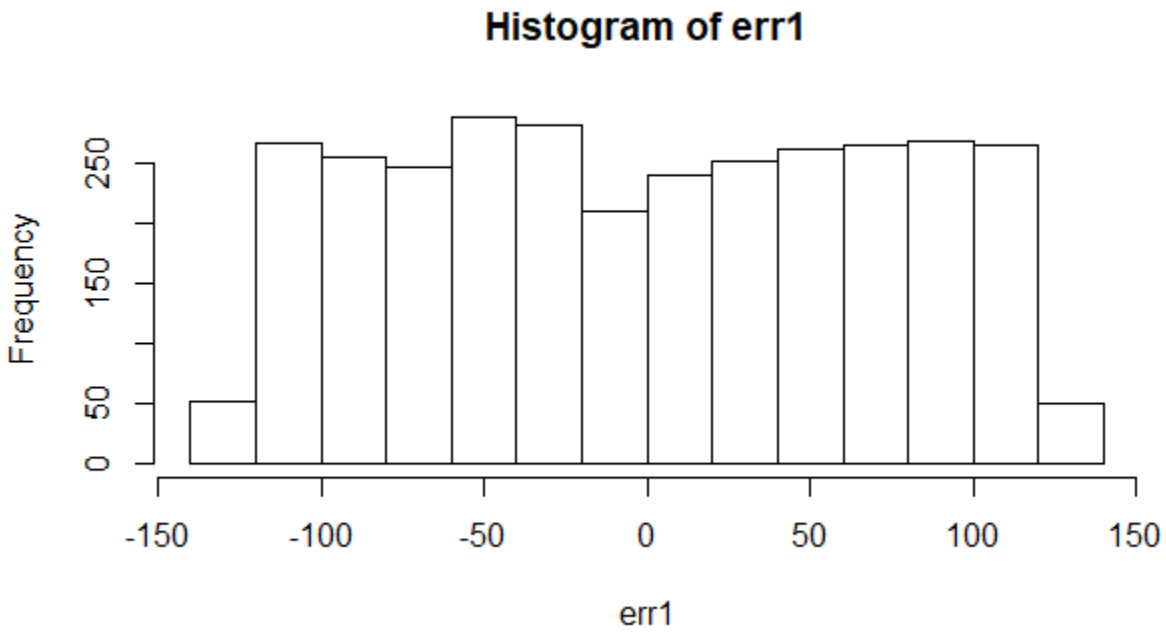
## Output:

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
```

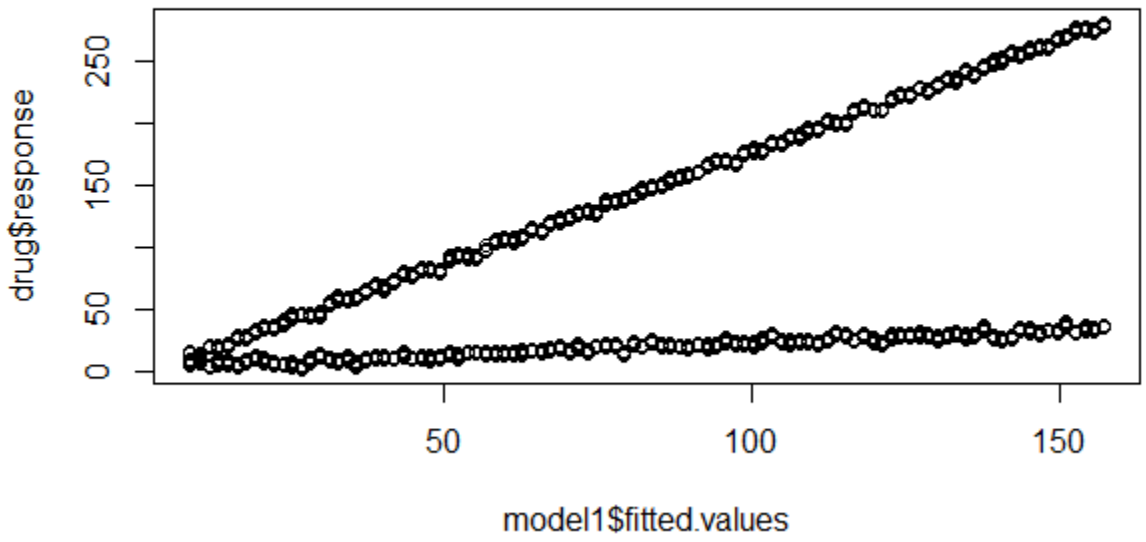
```
> #Subject: Machine Learning Lab 6
> #Task: Linear Regression Validation
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
>
> drug = read.csv("drug2.csv")
> attach(drug)
> head(drug)
  sex dose response
1   1  0.1    13.75
2   1  0.2    12.90
3   1  0.3    19.26
4   1  0.4    20.34
5   1  0.5    19.97
6   1  0.6    26.80
> nrow(drug)
[1] 3200
>
> model1 = lm(response~dose)
> summary(model1)

Call:
lm(formula = response ~ dose)

Residuals:
    Min      1Q  Median      3Q     Max
-123.514  -62.764    0.401   63.669  124.707

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.2534     2.5778   2.814  0.00493 **
dose         15.0020     0.4432  33.852  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 72.36 on 3198 degrees of freedom
Multiple R-squared:  0.2638,      Adjusted R-squared:  0.2636
F-statistic:  1146 on 1 and 3198 DF,  p-value: < 2.2e-16

> err1 = residuals(model1)
> plot(model1$fitted.values, err1)
```




```
> hist(err1)
```




```
> plot(model1$fitted.values, drug$response)
```




```
> mean <- mean(response)
> sst <- sum((response-mean)**2)
> sse <- sum((response-model1$fitted.values)**2)
> rsq <- 1 - (sse/sst)
> fuv <- sse/sst
> fuv
[1] 0.7361931
> rsq
[1] 0.2638069
> a <- (nrow(drug)-1)*(1-rsq)
> b <- nrow(drug)-2
> rad <- 1-(a/b)
> rad
[1] 0.2635767
```

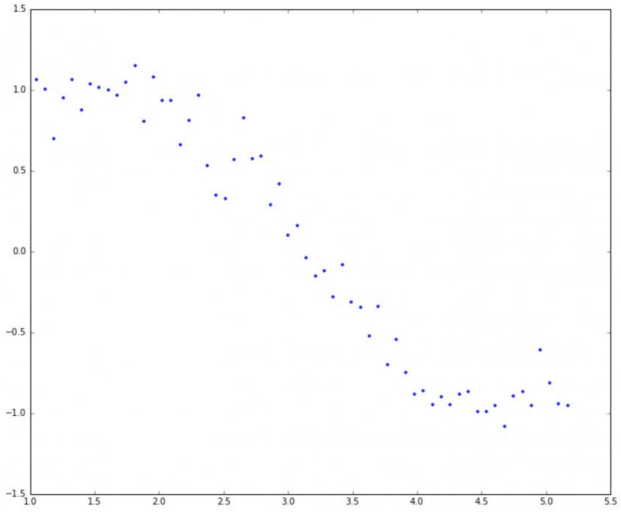
# Lab 6: Ridge/LASSO Regression

## Problem statement:

- Develop Ridge Regression model and try to tune it with varying alpha values. Plot SSE against each value of alpha. [Download suitable dataset with enough features, refer to access R-preloaded dataset]
- Develop LASSO Regression model and try to tune it with varying alpha values. Plot SSE against each value of alpha.
- Demonstrate the program SPARSITY property of LASSO Regression.

## Source Code and Output:

```
"""
Author: Ashish Upadhyay
Branch: Computer Science and Engineering
Semester: 6th
Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
Subject: Machine Learning Lab 7
Task: Ridge/LASSO Regression Implementation
"""

#Importing libraries.
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 12, 10

#Define input array with angles from 60deg to 300deg converted to radians
x = np.array([i*np.pi/180 for i in range(60,300,4)])
np.random.seed(10)  #Setting seed for reproducability
y = np.sin(x) + np.random.normal(0,0.15,len(x))
data = pd.DataFrame(np.column_stack([x,y]),columns=['x','y'])
plt.plot(data['x'],data['y'],'.')
```

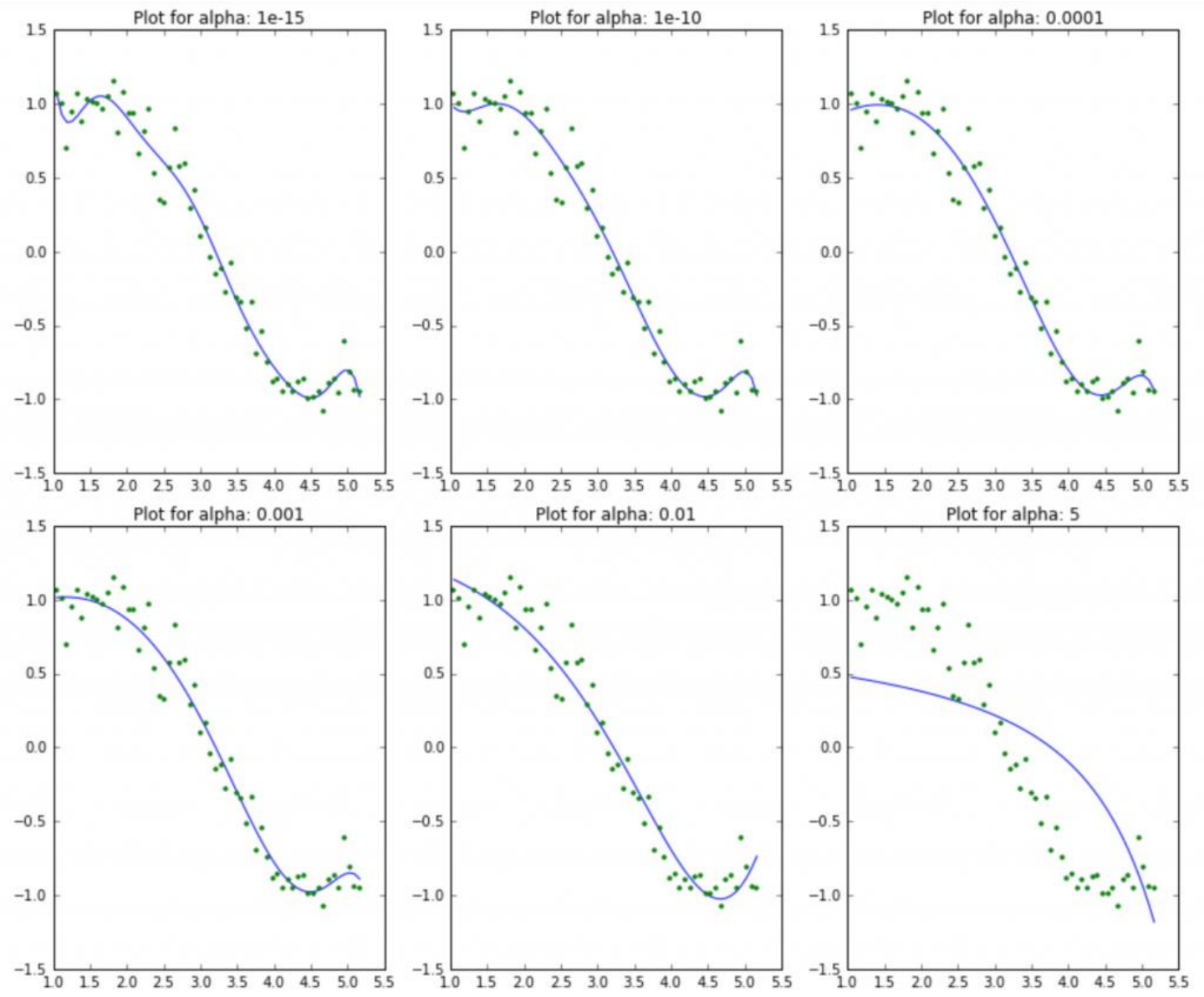


```
for i in range(2,16):  #power of 1 is already there
   colname = 'x_%d'%i      #new var will be x_power
   data[colname] = data['x']**i
print data.head()
```

```
          x         y       x_2       x_3       x_4       x_5       x_6  \
0  1.047198  1.065763  1.096623  1.148381  1.202581  1.259340  1.318778
1  1.117011  1.006086  1.247713  1.393709  1.556788  1.738948  1.942424
2  1.186824  0.695374  1.408551  1.671702  1.984016  2.354677  2.794587
3  1.256637  0.949799  1.579137  1.984402  2.493673  3.133642  3.937850
4  1.326450  1.063496  1.759470  2.333850  3.095735  4.106339  5.446854

        x_7       x_8        x_9       x_10       x_11       x_12       x_13
0  1.381021  1.446202   1.514459   1.585938   1.660790   1.739176   1.821260
1  2.169709  2.423588   2.707173   3.023942   3.377775   3.773011   4.214494
2  3.316683  3.936319   4.671717   5.544505   6.580351   7.809718   9.268760
3  4.948448  6.218404   7.814277   9.819710  12.339811  15.506664  19.486248
4  7.224981  9.583578  12.712139  16.862020  22.366630  29.668222  39.353420

        x_14       x_15
0   1.907219   1.997235
1   4.707635   5.258479
2  11.000386  13.055521
3  24.487142  30.771450
4  52.200353  69.241170
```

```
#Ridge Regression
from sklearn.linear_model import Ridge
def ridge_regression(data, predictors, alpha, models_to_plot={}):
   #Fit the model
   ridgereg = Ridge(alpha=alpha,normalize=True)
   ridgereg.fit(data[predictors],data['y'])
   y_pred = ridgereg.predict(data[predictors])

   #Check if a plot is to be made for the entered alpha
   if alpha in models_to_plot:
      plt.subplot(models_to_plot[alpha])
      plt.tight_layout()
      plt.plot(data['x'],y_pred)
      plt.plot(data['x'],data['y'],'.')
      plt.title('Plot for alpha: %.3g'%alpha)

   #Return the result in pre-defined format
   rss = sum((y_pred-data['y'])**2)
   ret = [rss]
   ret.extend([ridgereg.intercept_])
   ret.extend(ridgereg.coef_)
   return ret
```

```
#Initialize predictors to be set of 15 powers of x
predictors=['x']
predictors.extend(['x_%d'%i for i in range(2,16)])

#Set the different values of alpha to be tested
alpha_ridge = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]

#Initialize the dataframe for storing coefficients.
col = ['rss','intercept'] + ['coef_x_%d'%i for i in range(1,16)]
ind = ['alpha_%.2g'%alpha_ridge[i] for i in range(0,10)]
coef_matrix_ridge = pd.DataFrame(index=ind, columns=col)

models_to_plot = {1e-15:231, 1e-10:232, 1e-4:233, 1e-3:234, 1e-2:235, 5:236}
for i in range(10):
    coef_matrix_ridge.iloc[i,] = ridge_regression(data, predictors, alpha_ridge[i], models_to_plot)
```



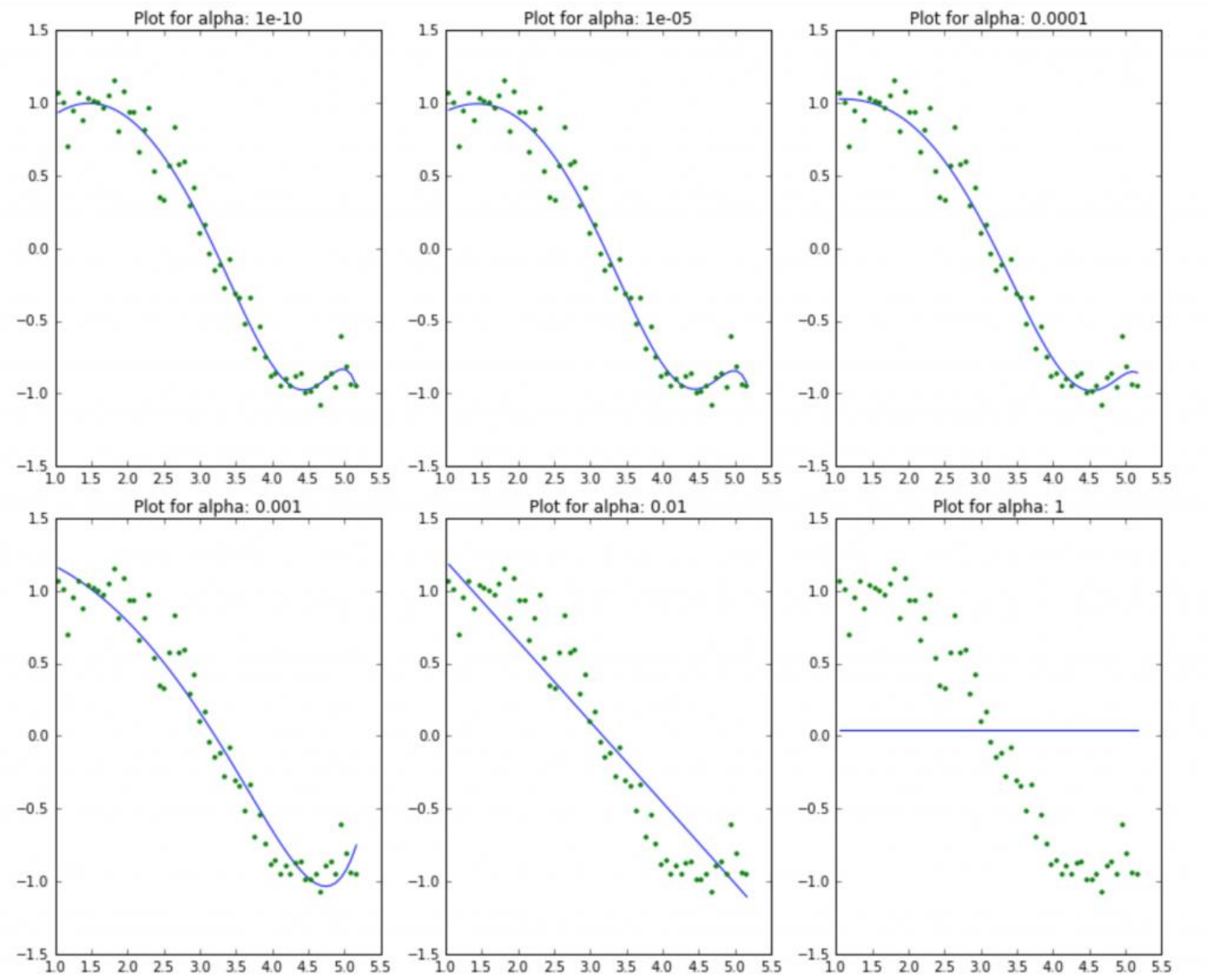
#Set the display format to be scientific for ease of analysis
pd.options.display.float_format = '{:,.2g}'.format
coef_matrix_ridge

| | rss | intercept | coef_x_1 | coef_x_2 | coef_x_3 | coef_x_4 | coef_x_5 | coef_x_6 | coef_x_7 | coef_x_8 | coef_x_9 | coef_x_10 | coef_x_11 | coe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **alpha_1e-15** | 0.87 | 95 | -3e+02 | 3.8e+02 | -2.4e+02 | 66 | 0.96 | -4.8 | 0.64 | 0.15 | -0.026 | -0.0054 | 0.00086 | 0.0 |
| **alpha_1e-10** | 0.92 | 11 | -29 | 31 | -15 | 2.9 | 0.17 | -0.091 | -0.011 | 0.002 | 0.00064 | 2.4e-05 | -2e-05 | -4.2 |
| **alpha_1e-08** | 0.95 | 1.3 | -1.5 | 1.7 | -0.68 | 0.039 | 0.016 | 0.00016 | -0.00036 | -5.4e-05 | -2.9e-07 | 1.1e-06 | 1.9e-07 | 2e- |
| **alpha_0.0001** | 0.96 | 0.56 | 0.55 | -0.13 | -0.026 | -0.0028 | -0.00011 | 4.1e-05 | 1.5e-05 | 3.7e-06 | 7.4e-07 | 1.3e-07 | 1.9e-08 | 1.9 |
| **alpha_0.001** | 1 | 0.82 | 0.31 | -0.087 | -0.02 | -0.0028 | -0.00022 | 1.8e-05 | 1.2e-05 | 3.4e-06 | 7.3e-07 | 1.3e-07 | 1.9e-08 | 1.7 |
| **alpha_0.01** | 1.4 | 1.3 | -0.088 | -0.052 | -0.01 | -0.0014 | -0.00013 | 7.2e-07 | 4.1e-06 | 1.3e-06 | 3e-07 | 5.6e-08 | 9e-09 | 1.1 |
| **alpha_1** | 5.6 | 0.97 | -0.14 | -0.019 | -0.003 | -0.00047 | -7e-05 | -9.9e-06 | -1.3e-06 | -1.4e-07 | -9.3e-09 | 1.3e-09 | 7.8e-10 | 2.4 |
| **alpha_5** | 14 | 0.55 | -0.059 | -0.0085 | -0.0014 | -0.00024 | -4.1e-05 | -6.9e-06 | -1.1e-06 | -1.9e-07 | -3.1e-08 | -5.1e-09 | -8.2e-10 | -1.3 |
| **alpha_10** | 18 | 0.4 | -0.037 | -0.0055 | -0.00095 | -0.00017 | -3e-05 | -5.2e-06 | -9.2e-07 | -1.6e-07 | -2.9e-08 | -5.1e-09 | -9.1e-10 | -1.6 |
| **alpha_20** | 23 | 0.28 | -0.022 | -0.0034 | -0.0006 | -0.00011 | -2e-05 | -3.6e-06 | -6.6e-07 | -1.2e-07 | -2.2e-08 | -4e-09 | -7.5e-10 | -1.4 |

coef_matrix_ridge.apply(lambda x: sum(x.values==0),axis=1)

```
alpha_1e-15      0
alpha_1e-10      0
alpha_1e-08      0
alpha_0.0001     0
alpha_0.001      0
alpha_0.01       0
alpha_1          0
alpha_5          0
alpha_10         0
alpha_20         0
dtype: int64
```

```
#LASSO Rigression

from sklearn.linear_model import Lasso
def lasso_regression(data, predictors, alpha, models_to_plot={}):
    #Fit the model
    lassoreg = Lasso(alpha=alpha,normalize=True, max_iter=1e5)
    lassoreg.fit(data[predictors],data['y'])
    y_pred = lassoreg.predict(data[predictors])

    #Check if a plot is to be made for the entered alpha
    if alpha in models_to_plot:
        plt.subplot(models_to_plot[alpha])
        plt.tight_layout()
        plt.plot(data['x'],y_pred)
        plt.plot(data['x'],data['y'],'.')
        plt.title('Plot for alpha: %.3g'%alpha)

    #Return the result in pre-defined format
    rss = sum((y_pred-data['y'])**2)
    ret = [rss]
    ret.extend([lassoreg.intercept_])
    ret.extend(lassoreg.coef_)
    return ret

#Initialize predictors to all 15 powers of x
predictors=['x']
predictors.extend(['x_%d'%i for i in range(2,16)])

#Define the alpha values to test
alpha_lasso = [1e-15, 1e-10, 1e-8, 1e-5,1e-4, 1e-3,1e-2, 1, 5, 10]

#Initialize the dataframe to store coefficients
col = ['rss','intercept'] + ['coef_x_%d'%i for i in range(1,16)]
ind = ['alpha_%.2g'%alpha_lasso[i] for i in range(0,10)]
coef_matrix_lasso = pd.DataFrame(index=ind, columns=col)

#Define the models to plot
models_to_plot = {1e-10:231, 1e-5:232,1e-4:233, 1e-3:234, 1e-2:235, 1:236}

#Iterate over the 10 alpha values:
for i in range(10):
    coef_matrix_lasso.iloc[i,] = lasso_regression(data, predictors, alpha_lasso[i], models_to_plot)
```




| | rss | intercept | coef_x_1 | coef_x_2 | coef_x_3 | coef_x_4 | coef_x_5 | coef_x_6 | coef_x_7 | coef_x_8 | coef_x_9 | coef_x_10 | coef_x_11 | coe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **alpha_1e-15** | 0.96 | 0.22 | 1.1 | -0.37 | 0.00089 | 0.0016 | -0.00012 | -6.4e-05 | -6.3e-06 | 1.4e-06 | 7.8e-07 | 2.1e-07 | 4e-08 | 5.4 |
| **alpha_1e-10** | 0.96 | 0.22 | 1.1 | -0.37 | 0.00088 | 0.0016 | -0.00012 | -6.4e-05 | -6.3e-06 | 1.4e-06 | 7.8e-07 | 2.1e-07 | 4e-08 | 5.4 |
| **alpha_1e-08** | 0.96 | 0.22 | 1.1 | -0.37 | 0.00077 | 0.0016 | -0.00011 | -6.4e-05 | -6.3e-06 | 1.4e-06 | 7.8e-07 | 2.1e-07 | 4e-08 | 5.3 |
| **alpha_1e-05** | 0.96 | 0.5 | 0.6 | -0.13 | -0.038 | -0 | 0 | 0 | 0 | 7.7e-06 | 1e-06 | 7.7e-08 | 0 | 0 |
| **alpha_0.0001** | 1 | 0.9 | 0.17 | -0 | -0.048 | -0 | -0 | 0 | 0 | 9.5e-06 | 5.1e-07 | 0 | 0 | 0 |
| **alpha_0.001** | 1.7 | 1.3 | -0 | -0.13 | -0 | -0 | -0 | 0 | 0 | 0 | 0 | 0 | 1.5e-08 | 7.5 |
| **alpha_0.01** | 3.6 | 1.8 | -0.55 | -0.00056 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | 0 | 0 | 0 |
| **alpha_1** | 37 | 0.038 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 |
| **alpha_5** | 37 | 0.038 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 |
| **alpha_10** | 37 | 0.038 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 |

HIGH SPARSITY

coef_matrix_lasso.apply(lambda x: sum(x.values==0),axis=1)

```
alpha_1e-15      0
alpha_1e-10      0
alpha_1e-08      0
alpha_1e-05      8
alpha_0.0001    10
alpha_0.001     12
alpha_0.01      13
alpha_1         15
alpha_5         15
alpha_10        15
dtype: int64
```

# Lab 8: Logistic Regression

## Problem statement:

Refer attached bank.csv dataset. Develop a logistic regression classification model as:

- Class variable: repaid
- Independent variable: age and salary
- With summary command observe the results
- Display the probability of each data record
- Calculate and display the assigned class with respect to cut off value 0.5.
- Calculate and display the confusion matrix
- Calculate the accuracy of model
- Calculate the error rate of model
- Calculate the recall of model
- Calculate the precision of model

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 8
#Task: Logistic Regression Implementation - Part I

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()

train <- read.csv("bank.csv")
nrow(train)
head(train)

#install.packages('caTools')
library(caTools)

set.seed(88)
split <- sample.split(train$repaid, SplitRatio = 0.75)

#get training and test data
dresstrain <- subset(train, split == TRUE)
dresstest <- subset(train, split == FALSE)

#Logistic Regression Model
model <- glm (repaid ~ ., data = dresstrain, family = binomial)

#Summary
summary(model)

#Probability
probability <- predict(model, type = 'response')
probability
```

```
#Confusion Matrix (Cut-off value = 0.5)
con_mat <- table(dresstrain$repaid, probability > 0.5)
con_mat

#Accuracy
accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
accuracy

#Precision
precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
precision

#Recall
recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
recall

#Error Rate
error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
error_rate

#F1 score
f1 <- (2 * precision * recall) / (precision + recall)
f1
```

## Output:

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 8
> #Task: Logistic Regression Implementation - Part I
>
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
>
> train <- read.csv("bank.csv")
> nrow(train)
[1] 2952
> head(train)
  age salary repaid
1  70     55      1
2  96     51      1
3  86     71      1
4  87     67      1
5  77     87      1
6  74     87      1
>
> #install.packages('caTools')
> library(caTools)
>
> set.seed(88)
```

```
> split <- sample.split(train$repaid, SplitRatio = 0.75)
>
> #get training and test data
> dresstrain <- subset(train, split == TRUE)
> dresstest <- subset(train, split == FALSE)
> #Logistic Regression Model
> model <- glm (repaid ~ ., data = dresstrain, family = binomial)
>
> #Summary
> summary(model)

Call:
glm(formula = repaid ~ ., family = binomial, data = dresstrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.63965  -0.14935   0.08644   0.34941   3.14588

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -16.044781   0.790730  -20.29   <2e-16 ***
age           0.160963   0.007738   20.80   <2e-16 ***
salary        0.117573   0.006353   18.50   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2756.6  on 2213  degrees of freedom
Residual deviance: 1130.5  on 2211  degrees of freedom
AIC: 1136.5

Number of Fisher Scoring iterations: 7

>
> #Probability
> probability <- predict(model, type = 'response')
> #Confusion Matrix (Cut-off value = 0.5)
> con_mat <- table(dresstrain$repaid, probability > 0.5)
> con_mat

    FALSE TRUE
  0   547  149
  1   102 1416
>
> #Accuracy
> accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> accuracy
[1] 88.66305
>
> #Precision
> precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
> precision
[1] 90.47923
>
```

```
> #Recall
> recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
> recall
[1] 93.28063
>
> #Error Rate
> error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> error_rate
[1] 24.70641
>
> #F1 score
> f1 <- (2 * precision * recall) / (precision + recall)
> f1
[1] 91.85858
```

# Lab 10: Logistic Regression - II

## Problem statement:

Refer attached blooddonation.CSV dataset. Develop a logistic regression classification model as:

- Class variable: donate
- Independent variable: remaining all
- With summary command observe the results
- Display the probability of each data record
- Calculate and display the assigned class with respect to cut off value 0.5.
- Calculate and display the confusion matrix
- Calculate the accuracy of model
- Calculate the error rate of model
- Calculate the recall of model
- Calculate the precision of model
- Construct the ROC curve
- Repeat the steps from (v) to (xi) with cutoff 0.2 and 0.8

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 8
#Task: Logistic Regression Implementation - Part II

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()

train <- read.csv("blooddonation.csv")
nrow(train)
head(train)

#install.packages('caTools')
library(caTools)

set.seed(88)
split <- sample.split(train$donate, SplitRatio = 0.75)

#get training and test data
dresstrain <- subset(train, split == TRUE)
dresstest <- subset(train, split == FALSE)

#Logistic Regression Model
model <- glm (donate ~ ., data = dresstrain, family = binomial)

#Summary
summary(model)

#Probability
```

```
probability <- predict(model, type = 'response')
probability


#Confusion Matrix (Cut-off value = 0.5)
con_mat <- table(dresstrain$donate, probability > 0.5)
con_mat

#Accuracy
accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
accuracy

#Precision
precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
precision

#Recall
recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
recall

#Error Rate
error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
error_rate

#ROCR Curve
#install.packages('ROCR')
library(ROCR)
ROCRpred <- prediction(probability, dresstrain$donate)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))


#Confusion Matrix (Cut-off value = 0.2)
con_mat <- table(dresstrain$donate, probability > 0.2)
con_mat

#Accuracy
accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
accuracy

#Precision
precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
precision

#Recall
recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
recall

#Error Rate
error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
error_rate
```

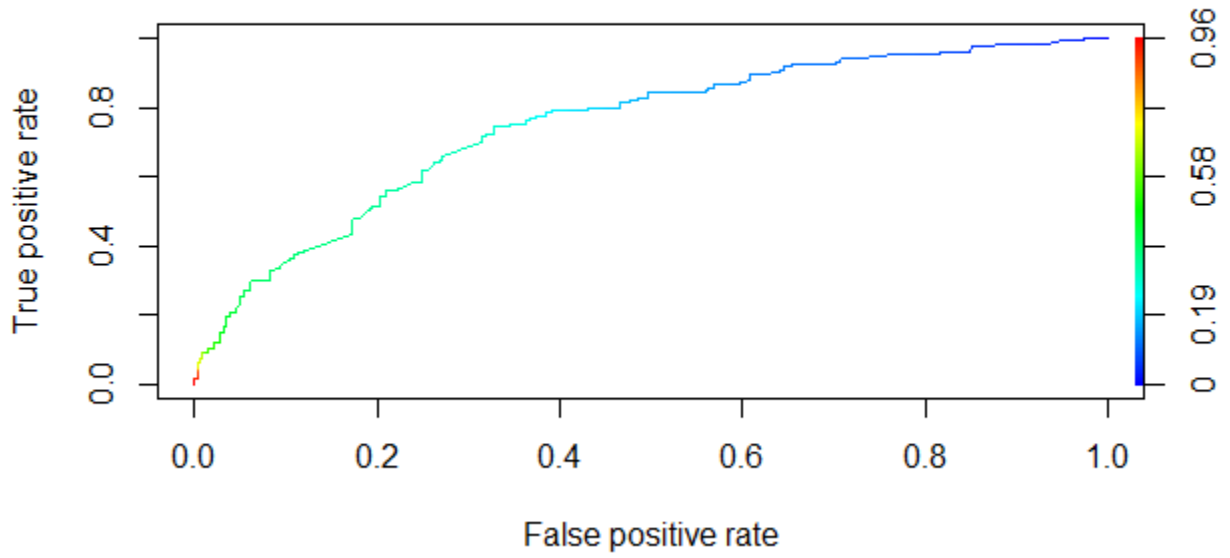
```
#ROCR Curve
#install.packages('ROCR')
library(ROCR)
ROCRpred <- prediction(probability, dresstrain$donate)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))


#Confusion Matrix (Cut-off value = 0.8)
con_mat <- table(dresstrain$donate, probability > 0.8)
con_mat

#Accuracy
accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
accuracy

#Precision
precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
precision

#Recall
recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
recall

#Error Rate
error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
error_rate

#ROCR Curve
#install.packages('ROCR')
library(ROCR)
ROCRpred <- prediction(probability, dresstrain$donate)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```

## Output:

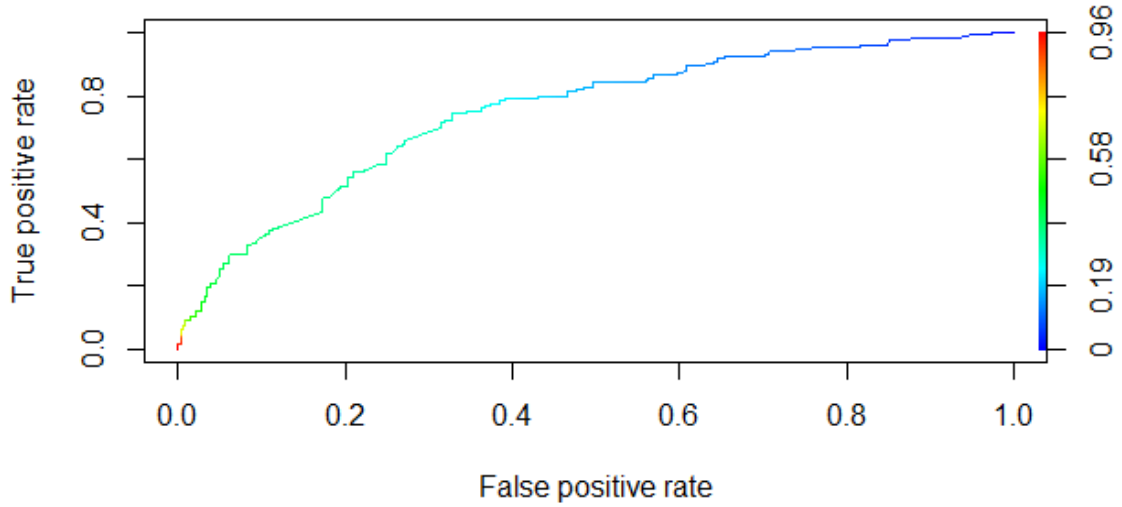
```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 8
> #Task: Logistic Regression Implementation - Part II
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
>
> train <- read.csv("blooddonation.csv")
> nrow(train)
[1] 748
```

```
> head(train)
  Recency..months. Frequency..times. Monetary..c.c..blood. Time..months. donate
1                2                50                12500             98      1
2                0                13                 3250             28      1
3                1                16                 4000             35      1
4                2                20                 5000             45      1
5                1                24                 6000             77      0
6                4                 4                 1000              4      0


> library(caTools)
>
> set.seed(88)
> split <- sample.split(train$donate, SplitRatio = 0.75)
>
> #get training and test data
> dresstrain <- subset(train, split == TRUE)
> dresstest <- subset(train, split == FALSE)
>
> #Logistic Regression Model
> model <- glm (donate ~ ., data = dresstrain, family = binomial)
>
> #Summary
> summary(model)

Call:
glm(formula = donate ~ ., family = binomial, data = dresstrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2778  -0.8168  -0.4871  -0.1833   2.6719

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.44933    0.20315  -2.212  0.02698 *
Recency..months.    -0.10811    0.02035  -5.313 1.08e-07 ***
Frequency..times.    0.11010    0.02684   4.102 4.09e-05 ***
Monetary..c.c..blood.      NA         NA      NA       NA
Time..months.       -0.01696    0.00649  -2.614  0.00896 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 617.38  on 561  degrees of freedom
Residual deviance: 535.65  on 558  degrees of freedom
AIC: 543.65

Number of Fisher Scoring iterations: 5

>
> #Probability
> probability <- predict(model, type = 'response')
> probability
```

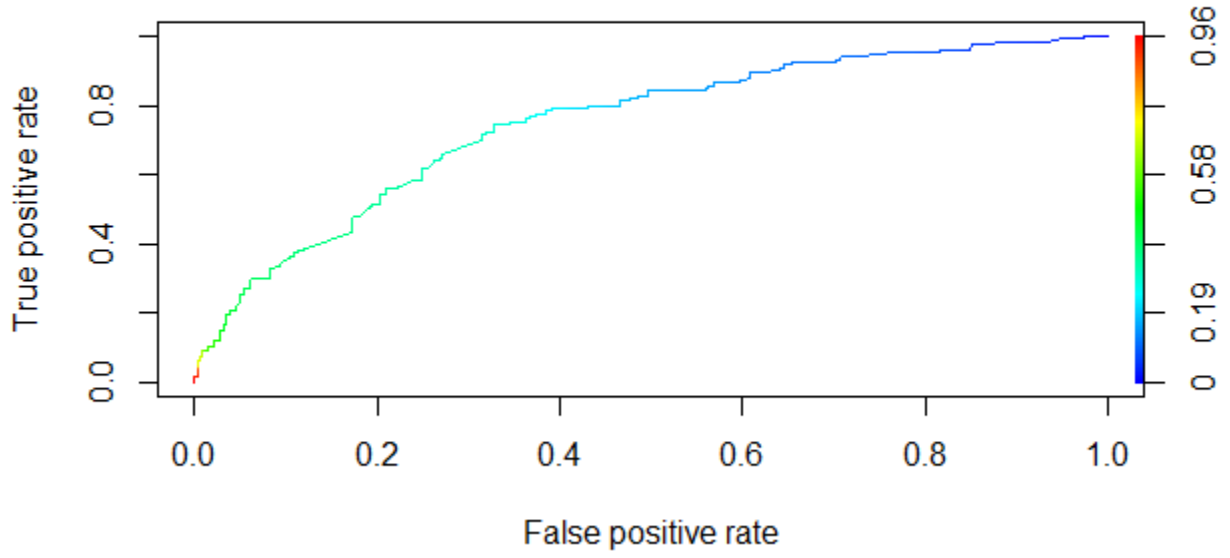
```
> #Confusion Matrix (Cut-off value = 0.5)
> con_mat <- table(dresstrain$donate, probability > 0.5)
> con_mat

   FALSE TRUE
 0   417   11
 1   118   16

> #Accuracy
> accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> accuracy
[1] 77.04626
> #Precision
> precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
> precision
[1] 59.25926
> #Recall
> recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
> recall
[1] 11.9403
> #Error Rate
> error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> error_rate
[1] 74.19929
> #ROCR Curve
> #install.packages('ROCR')
> library(ROCR)
> ROCRpred <- prediction(probability, dresstrain$donate)
> ROCRperf <- performance(ROCRpred, 'tpr','fpr')
> plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```



```
> #Confusion Matrix (Cut-off value = 0.2)
> con_mat <- table(dresstrain$donate, probability > 0.2)
> con_mat

   FALSE TRUE
 0   239  189
 1    27  107
>
> #Accuracy
> accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> accuracy
[1] 61.56584
> #Precision
> precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
> precision
[1] 36.14865
> #Recall
> recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
> recall
[1] 79.85075
> #Error Rate
> error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> error_rate
[1] 42.52669
>
> #ROCR Curve
> #install.packages('ROCR')
> library(ROCR)
> ROCRpred <- prediction(probability, dresstrain$donate)
> ROCRperf <- performance(ROCRpred, 'tpr','fpr')
> plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```



```
> #Confusion Matrix (Cut-off value = 0.8)
> con_mat <- table(dresstrain$donate, probability > 0.8)
> con_mat

   FALSE TRUE
 0   427    1
 1   129    5
> #Accuracy
> accuracy <- ((con_mat[1, 1] + con_mat[2, 2])/(con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> accuracy
[1] 76.86833
> #Precision
> precision <- ((con_mat[2, 2]) / (con_mat[1, 2] + con_mat[2, 2])) * 100
> precision
[1] 83.33333
> #Recall
> recall <- (con_mat[2, 2] / (con_mat[2, 1] + con_mat[2, 2])) * 100
> recall
[1] 3.731343
> #Error Rate
> error_rate <- (con_mat[1, 1] / (con_mat[1, 1] + con_mat[1, 2] + con_mat[2, 1] + con_mat[2, 2])) * 100
> error_rate
[1] 75.97865
> #ROCR Curve
> #install.packages('ROCR')
> library(ROCR)
> ROCRpred <- prediction(probability, dresstrain$donate)
> ROCRperf <- performance(ROCRpred, 'tpr','fpr')
> plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```



# Lab 11: Multinomial Logistic Regression

## Problem statement:

Task1: Refer attached glass_multiclass.CSV dataset. Develop a multinomial logistic regression classification model as:

- Class variable: type
- First convert class variable into categorical variable
- Independent variable: remaining all
- With summary command observe the results
- Partitioned the data set into train and test data
- Observe the dimension of each data set
- Display and observe the summary of model
- Calculate and display confusion matrix for test dataset
- Calculate and display confusion matrix for train dataset
- Calculate the accuracy of model i.e. against both data set (train and test)
- Calculate the error rate of model i.e. against both data set (train and test)

Task-2: Develop four multinomial logistic regression model as:

- model-1: independent variables (RI,Na, Mg)
- model-2: independent variables (RI,Na, Mg, AI, SI, K)
- model-3: independent variables (RI,Na, Mg, AI, SI, K, Ca, Ba, Fe)
- model-3: independent variables (SI, K, Ca, Ba, Fe)
- Find out best model with justifications

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 11
#Task: Multinomial Logistic Regression Implementation

#Task I
setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()

data_set <- read.csv("glass_multiclass.csv")
head(data_set)
nrow(data_set)

names(data_set)
dim(data_set)

summary(data_set)

data_set$Type <- as.factor(data_set$Type)

library(caTools)
set.seed(88)
split <- sample.split(data_set$Type, SplitRatio = 0.75)
```

```
#get training and test data
train <- subset(data_set, split == TRUE)
test <- subset(data_set, split == FALSE)

dim(test)
dim(train)

table(train$Type)
table(test$Type)

#install.packages('nnet')
library("nnet")

#Multinomial Model
model <- multinom(Type ~ ., data = train, maxit = 1000)
summary(model)

predictions_train <- predict(model, train)
con_mat_train_model <- table(predicted = predictions_train, actual = train$Type)
con_mat_train_model
accuracy_train_model <- sum(diag(con_mat_train_model)) / sum(con_mat_train_model)
accuracy_train_model
error_rate_train_model <- 1 - accuracy_train_model
error_rate_train_model

predictions_test <- predict(model, test)
con_mat_test_model <- table(predicted = predictions_test, actual = test$Type)
con_mat_test_model
accuracy_test_model <- sum(diag(con_mat_test_model)) / sum(con_mat_test_model)
accuracy_test_model
error_rate_test_model <- 1 - accuracy_train_model
error_rate_test_model


#Task II
#Model-1
model1 <- multinom(Type ~ RI + Na + Mg, data = train, maxit = 1000)
summary(model1)

predictions_train1 <- predict(model1, train)
con_mat_train_model1 <- table(predicted = predictions_train1, actual = train$Type)
con_mat_train_model1
accuracy_train_model1 <- sum(diag(con_mat_train_model1)) / sum(con_mat_train_model1)
accuracy_train_model1
error_rate_train_model1 <- 1 - accuracy_train_model1
error_rate_train_model1

predictions_test1 <- predict(model1, test)
con_mat_test_model1 <- table(predicted = predictions_test1, actual = test$Type)
con_mat_test_model1
accuracy_test_model1 <- sum(diag(con_mat_test_model1)) / sum(con_mat_test_model1)
```

```
accuracy_test_model1
error_rate_test_model1 <- 1 - accuracy_train_model1
error_rate_test_model1


#Model-2
model2 <- multinom(Type ~ RI + Na + Mg + Al + Si + K, data = train, maxit = 1000)
summary(model2)

predictions_train2 <- predict(model2, train)
con_mat_train_model2 <- table(predicted = predictions_train2, actual = train$Type)
con_mat_train_model2
accuracy_train_model2 <- sum(diag(con_mat_train_model2)) / sum(con_mat_train_model2)
accuracy_train_model2
error_rate_train_model2 <- 1 - accuracy_train_model2
error_rate_train_model2

predictions_test2 <- predict(model2, test)
con_mat_test_model2 <- table(predicted = predictions_test2, actual = test$Type)
con_mat_test_model2
accuracy_test_model2 <- sum(diag(con_mat_test_model2)) / sum(con_mat_test_model2)
accuracy_test_model2
error_rate_test_model2 <- 1 - accuracy_train_model2
error_rate_test_model2


#Model-3
model3 <- multinom(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba +Fe, data = train, maxit = 1000)
summary(model3)

predictions_train3 <- predict(model3, train)
con_mat_train_model3 <- table(predicted = predictions_train3, actual = train$Type)
con_mat_train_model3
accuracy_train_model3 <- sum(diag(con_mat_train_model3)) / sum(con_mat_train_model3)
accuracy_train_model3
error_rate_train_model3 <- 1 - accuracy_train_model3
error_rate_train_model3

predictions_test3 <- predict(model3, test)
con_mat_test_model3 <- table(predicted = predictions_test3, actual = test$Type)
con_mat_test_model3
accuracy_test_model3 <- sum(diag(con_mat_test_model3)) / sum(con_mat_test_model3)
accuracy_test_model3
error_rate_test_model3 <- 1 - accuracy_train_model3
error_rate_test_model3


#Model-4
model4 <- multinom(Type ~ Si + K + Ca + Ba + Fe, data = train, maxit = 1000)
summary(model4)
```

```
predictions_train4 <- predict(model4, train)
con_mat_train_model4 <- table(predicted = predictions_train4, actual = train$Type)
#con_mat_train_model4
accuracy_train_model4 <- sum(diag(con_mat_train_model4)) / sum(con_mat_train_model4)
accuracy_train_model4
error_rate_train_model4 <- 1 - accuracy_train_model4
error_rate_train_model4

predictions_test4 <- predict(model4, test)
con_mat_test_model4 <- table(predicted = predictions_test4, actual = test$Type)
#con_mat_test_model4
accuracy_test_model4 <- sum(diag(con_mat_test_model4)) / sum(con_mat_test_model4)
accuracy_test_model4
error_rate_test_model4 <- 1 - accuracy_train_model4
error_rate_test_model4
```

**Output:**

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 11
> #Task: Multinomial Logistic Regression Implementation
>
>
>
> #Task I
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
>
> data_set <- read.csv("glass_multiclass.csv")
> head(data_set)
       RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
> nrow(data_set)
[1] 214
>
> names(data_set)
 [1] "RI"   "Na"   "Mg"   "Al"   "Si"   "K"    "Ca"   "Ba"   "Fe"   "Type"
> dim(data_set)
[1] 214  10
>
> summary(data_set)
       RI              Na              Mg              Al              Si
 Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290   Min.   :69.81
 1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190   1st Qu.:72.28
 Median :1.518   Median :13.30   Median :3.480   Median :1.360   Median :72.79
 Mean   :1.518   Mean   :13.41   Mean   :2.685   Mean   :1.445   Mean   :72.65
```

```
 3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630   3rd Qu.:73.09
 Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500   Max.   :75.41
       K                Ca               Ba               Fe              Type
 Min.   :0.0000   Min.   : 5.430   Min.   :0.000   Min.   :0.00000   Min.   :1.00
 1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:1.00
 Median :0.5550   Median : 8.600   Median :0.000   Median :0.00000   Median :2.00
 Mean   :0.4971   Mean   : 8.957   Mean   :0.175   Mean   :0.05701   Mean   :2.78
 3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000   3rd Qu.:0.10000   3rd Qu.:3.00
 Max.   :6.2100   Max.   :16.190   Max.   :3.150   Max.   :0.51000   Max.   :7.00
>
> data_set$Type <- as.factor(data_set$Type)
>
> library(caTools)
> set.seed(88)
> split <- sample.split(data_set$Type, SplitRatio = 0.75)
>
> #get training and test data
> train <- subset(data_set, split == TRUE)
> test <- subset(data_set, split == FALSE)
>
> dim(test)
[1] 53 10
> dim(train)
[1] 161  10
>
> table(train$Type)

 1  2  3  5  6  7
52 57 13 10  7 22
> table(test$Type)

 1  2  3  5  6  7
18 19  4  3  2  7
>
> #install.packages('nnet')
> library("nnet")
>
> model <- multinom(Type ~ ., data = train, maxit = 1000)
# weights:  66 (50 variable)
initial  value 288.473275
iter  10 value 176.893686
iter  20 value 124.738149
iter  30 value 111.008787
iter  40 value 105.664700
iter  50 value 102.298369
iter  60 value 100.195435
iter  70 value 99.807182
iter  80 value 99.646948
iter  90 value 99.256888
iter 100 value 98.987365
iter 110 value 98.965893
iter 120 value 98.858676
iter 130 value 98.812843
iter 140 value 97.690144
iter 150 value 95.919478
iter 160 value 95.024378
iter 170 value 95.011113
```

```
iter 180 value 94.129473
iter 190 value 93.815132
iter 200 value 93.699533
iter 210 value 93.642554
iter 220 value 93.623661
iter 230 value 93.574819
iter 240 value 93.531863
iter 250 value 93.474273
iter 260 value 93.302727
iter 270 value 93.180463
iter 280 value 93.160441
iter 290 value 92.970575
iter 300 value 92.835629
iter 310 value 92.782585
iter 320 value 92.775068
iter 330 value 92.757020
iter 340 value 92.674846
iter 350 value 92.570052
iter 360 value 92.334816
iter 370 value 92.321175
iter 380 value 92.302224
iter 390 value 91.971135
iter 400 value 90.833979
iter 410 value 89.977295
iter 420 value 89.941450
iter 430 value 89.928643
iter 440 value 89.919600
iter 450 value 89.911005
iter 460 value 89.828482
iter 470 value 89.602676
iter 480 value 89.470807
iter 490 value 89.454347
iter 500 value 89.360322
iter 510 value 89.277129
iter 520 value 89.272248
iter 530 value 89.267633
iter 540 value 89.257607
iter 550 value 89.192259
iter 560 value 89.152085
iter 570 value 89.077939
iter 580 value 88.812195
iter 590 value 88.751789
iter 600 value 88.731080
final  value 88.680339
converged
> summary(model)
Call:
multinom(formula = Type ~ ., data = train, maxit = 1000)

Coefficients:
  (Intercept)         RI         Na         Mg           Al         Si           K
2  203.026613  171.42504  -4.099733  -5.800582  -0.06659444  -4.795857   -3.128619
3  848.442724 -786.05540   5.433639   4.239372   4.84957130   2.743805    4.141046
5    5.733115   12.99851 -81.581835 -77.628788 222.12921154   6.134533  254.091839
6  -14.058868  -23.50412 -15.226861 -27.438127   9.75009541   8.231017 -368.668129
7 -143.968089  933.95243 -11.264131 -18.845955 -12.60991726 -12.313294  -10.288273
          Ca          Ba           Fe
```

```
2  -4.495158   -6.789358    0.6437630
3   5.646475    2.223870    0.8239636
5  16.994975 -169.437118  272.7346400
6 -28.089356 -227.566859 -319.8861733
7 -17.173076   -6.244567  -95.8431176

Std. Errors:
  (Intercept)         RI         Na         Mg         Al         Si            K
2  0.03920231 0.06701881  0.5768557 0.85545632 1.65179176 0.1511549 2.289055e+00
3  0.06114337 0.09642064  0.8058941 1.12917704 1.76232222 0.2059976 2.938445e+00
5  0.06245625 0.08769877 12.2913630 8.40708139 1.89682777 2.6617805 2.304166e+00
6  0.01613113 0.02450860  0.2228231 0.03882437 0.01984018 1.1730231 4.111865e-10
7  0.10335390 0.16375551  2.0999998 2.96043069 6.87912454 0.7338250 5.774488e+00
         Ca           Ba           Fe
2 0.5632691 2.036229e+00 2.320576e+00
3 0.6197758 4.502377e+00 3.847483e+00
5 5.3353436 7.067090e+00 1.955795e-02
6 0.1572593 3.993370e-10 2.896739e-22
7 2.6269938 5.059777e+00 1.303330e+00

Residual Deviance: 177.3607
AIC: 277.3607
>
> predictions_train <- predict(model, train)
> con_mat_train_model <- table(predicted = predictions_train, actual = train$Type)
> con_mat_train_model
         actual
predicted  1  2  3  5  6  7
        1 34 13  7  0  0  0
        2 15 44  4  0  0  1
        3  3  0  2  0  0  0
        5  0  0  0 10  0  0
        6  0  0  0  0  7  0
        7  0  0  0  0  0 21
> accuracy_train_model <- sum(diag(con_mat_train_model)) / sum(con_mat_train_model)
> accuracy_train_model
[1] 0.7329193
> error_rate_train_model <- 1 - accuracy_train_model
> error_rate_train_model
[1] 0.2670807
>
> predictions_test <- predict(model, test)
> con_mat_test_model <- table(predicted = predictions_test, actual = test$Type)
> con_mat_test_model
         actual
predicted  1  2  3  5  6  7
        1 13  4  2  0  0  0
        2  4 12  2  1  0  1
        3  1  0  0  0  0  0
        5  0  2  0  2  0  2
        6  0  1  0  0  2  0
        7  0  0  0  0  0  4
> accuracy_test_model <- sum(diag(con_mat_test_model)) / sum(con_mat_test_model)
> accuracy_test_model
[1] 0.6226415
> error_rate_test_model <- 1 - accuracy_train_model
> error_rate_test_model
```

```
[1] 0.2670807
>
> #Task II
>
> model1 <- multinom(Type ~ RI + Na + Mg, data = train, maxit = 1000)
# weights:  30 (20 variable)
initial  value 288.473275
iter  10 value 194.980593
iter  20 value 167.160823
iter  30 value 164.649443
iter  40 value 164.541471
iter  50 value 164.530243
iter  60 value 164.515760
iter  70 value 162.209446
iter  80 value 160.494303
iter  90 value 160.466967
iter 100 value 160.364315
iter 110 value 160.327626
iter 120 value 158.513344
iter 130 value 158.119465
iter 140 value 158.089116
iter 150 value 158.057444
iter 160 value 157.957919
iter 170 value 157.170662
iter 180 value 156.788012
iter 190 value 156.751352
iter 200 value 156.741216
iter 210 value 156.370516
iter 220 value 156.156771
iter 230 value 156.153140
iter 240 value 156.143920
iter 250 value 155.737327
iter 260 value 155.498724
iter 270 value 155.482735
iter 280 value 155.476436
iter 290 value 155.414348
iter 300 value 155.397862
iter 310 value 155.369762
iter 320 value 155.368009
iter 330 value 155.227592
iter 340 value 155.162822
iter 350 value 155.134074
iter 360 value 155.131872
iter 370 value 155.090479
iter 380 value 155.066985
iter 390 value 155.051411
iter 400 value 155.049556
iter 410 value 155.020009
iter 420 value 154.988330
iter 430 value 154.969417
iter 440 value 154.968105
iter 450 value 154.962323
iter 460 value 154.927968
iter 470 value 154.915351
iter 480 value 154.912895
iter 490 value 154.904885
iter 500 value 154.884565
```

```
iter 510 value 154.874404
iter 520 value 154.871606
iter 530 value 154.867561
iter 540 value 154.845921
iter 550 value 154.830774
iter 560 value 154.828572
iter 570 value 154.827326
iter 580 value 154.813451
iter 590 value 154.806541
iter 600 value 154.804931
iter 610 value 154.802746
iter 620 value 154.783254
iter 630 value 154.772970
iter 640 value 154.771725
iter 650 value 154.769581
iter 660 value 154.731171
iter 670 value 154.713668
iter 680 value 154.710243
iter 690 value 154.709176
iter 700 value 154.682573
iter 710 value 154.664084
iter 720 value 154.660278
iter 730 value 154.654326
iter 740 value 154.374744
iter 750 value 154.272857
iter 760 value 154.193204
iter 770 value 154.191000
iter 780 value 154.188690
iter 780 value 154.188690
final  value 154.188690
converged
> summary(model1)
Call:
multinom(formula = Type ~ RI + Na + Mg, data = train, maxit = 1000)

Coefficients:
  (Intercept)        RI          Na         Mg
2    293.7176 -192.9106  0.3743614 -1.6682382
3    312.8025 -221.8155  1.7941323 -0.3902145
5    942.8657 -612.5714 -0.3696066 -3.4544177
6    692.9573 -489.6930  4.0657805 -2.4007213
7    804.4952 -554.1998  3.2497388 -3.0097623

Std. Errors:
  (Intercept)       RI        Na        Mg
2    1.796424 2.598848 0.4639528 0.5105102
3    2.806285 4.160141 0.6650216 0.9503897
5    2.396639 3.735993 0.6462813 0.5914569
6    4.128280 6.282890 0.9807628 0.6592936
7    3.704468 5.669132 0.9015172 0.6242835

Residual Deviance: 308.3774
AIC: 348.3774
>
> predictions_train1 <- predict(model1, train)
> con_mat_train_model1 <- table(predicted = predictions_train1, actual = train$Type)
> con_mat_train_model1
```

```
         actual
predicted  1  2  3  5  6  7
        1 37 15  6  0  0  1
        2 14 38  7  4  2  2
        3  1  0  0  0  0  0
        5  0  1  0  5  0  0
        6  0  1  0  0  1  1
        7  0  2  0  1  4 18
> accuracy_train_model1 <- sum(diag(con_mat_train_model1)) / sum(con_mat_train_model1)
> accuracy_train_model1
[1] 0.6149068
> error_rate_train_model1 <- 1 - accuracy_train_model1
> error_rate_train_model1
[1] 0.3850932
>
> predictions_test1 <- predict(model1, test)
> con_mat_test_model1 <- table(predicted = predictions_test1, actual = test$Type)
> con_mat_test_model1
         actual
predicted  1  2  3  5  6  7
        1 13  5  2  0  0  0
        2  5 12  2  1  1  0
        3  0  0  0  0  0  0
        5  0  2  0  2  0  1
        6  0  0  0  0  0  1
        7  0  0  0  0  1  5
> accuracy_test_model1 <- sum(diag(con_mat_test_model1)) / sum(con_mat_test_model1)
> accuracy_test_model1
[1] 0.6037736
> error_rate_test_model1 <- 1 - accuracy_train_model1
> error_rate_test_model1
[1] 0.3850932
>
>
> model2 <- multinom(Type ~ RI + Na + Mg + Al + Si + K, data = train, maxit = 1000)
# weights:  48 (35 variable)
initial  value 288.473275
iter  10 value 188.517305
iter  20 value 135.361327
iter  30 value 128.644801
iter  40 value 121.811953
iter  50 value 121.356234
iter  60 value 120.873503
iter  70 value 120.287512
iter  80 value 120.268778
iter  90 value 120.149085
iter 100 value 119.779554
iter 110 value 119.735794
iter 120 value 119.081482
iter 130 value 118.860893
iter 140 value 118.629484
final  value 118.627486
converged
> summary(model2)
Call:
multinom(formula = Type ~ RI + Na + Mg + Al + Si + K, data = train,
    maxit = 1000)
```

```
Coefficients:
  (Intercept)         RI          Na         Mg         Al         Si          K
2    25.98450   17.27879  0.07518246 -1.4743160  3.9717069 -0.7444438  1.6374824
3    83.43849  -50.87758  1.32275797 -0.4900864  0.3782597 -0.3346317  0.6921906
5   -56.50829  -67.52222 -0.90700945 -4.3176835 10.2027255  2.2149760  6.2701371
6   -75.05828 -117.38643  6.05473558 -0.8943887  2.6219146  2.3445152 -138.3949282
7   -97.40926  -95.56172  4.62460585 -2.7900264  8.8675225  2.3607118  3.3995930

Std. Errors:
  (Intercept)         RI        Na        Mg       Al        Si          K
2  9.58391996 16.9738175 0.6698951 0.5640709 1.297138 0.4380523 1.98893192
3  3.04300834  5.4808904 0.7190405 0.8414779 1.560872 0.1928353 2.47934650
5  1.65510933  2.8672799 1.1272471 0.9762632 2.536777 0.2283417 3.51310366
6  0.07125727  0.1225657 2.0507160 1.4377365 3.892942 0.4584804 0.09775972
7  1.23496880  2.2092538 0.8977652 0.7951621 2.328618 0.1734089 2.68208189

Residual Deviance: 237.255
AIC: 307.255
>
> predictions_train2 <- predict(model2, train)
> con_mat_train_model2 <- table(predicted = predictions_train2, actual = train$Type)
> con_mat_train_model2
         actual
predicted  1  2  3  5  6  7
        1 36 16  7  0  0  0
        2 16 38  6  1  0  2
        3  0  0  0  0  0  0
        5  0  0  0  9  0  0
        6  0  0  0  0  7  1
        7  0  3  0  0  0 19
> accuracy_train_model2 <- sum(diag(con_mat_train_model2)) / sum(con_mat_train_model2)
> accuracy_train_model2
[1] 0.6770186
> error_rate_train_model2 <- 1 - accuracy_train_model2
> error_rate_train_model2
[1] 0.3229814
>
> predictions_test2 <- predict(model2, test)
> con_mat_test_model2 <- table(predicted = predictions_test2, actual = test$Type)
> con_mat_test_model2
         actual
predicted  1  2  3  5  6  7
        1 14  7  2  0  0  0
        2  4 10  2  1  0  1
        3  0  0  0  0  0  0
        5  0  1  0  2  0  1
        6  0  1  0  0  1  0
        7  0  0  0  0  1  5
> accuracy_test_model2 <- sum(diag(con_mat_test_model2)) / sum(con_mat_test_model2)
> accuracy_test_model2
[1] 0.6037736
> error_rate_test_model2 <- 1 - accuracy_train_model2
> error_rate_test_model2
[1] 0.3229814
>
>
```

```
> model3 <- multinom(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba +Fe, data = train, maxit
= 1000)
# weights:  66 (50 variable)
initial  value 288.473275
iter  10 value 176.893686
iter  20 value 124.738149
iter  30 value 111.008787
iter  40 value 105.664700
iter  50 value 102.298369
iter  60 value 100.195435
iter  70 value 99.807182
iter  80 value 99.646948
iter  90 value 99.256888
iter 100 value 98.987365
iter 110 value 98.965893
iter 120 value 98.858676
iter 130 value 98.812843
iter 140 value 97.690144
iter 150 value 95.919478
iter 160 value 95.024378
iter 170 value 95.011113
iter 180 value 94.129473
iter 190 value 93.815132
iter 200 value 93.699533
iter 210 value 93.642554
iter 220 value 93.623661
iter 230 value 93.574819
iter 240 value 93.531863
iter 250 value 93.474273
iter 260 value 93.302727
iter 270 value 93.180463
iter 280 value 93.160441
iter 290 value 92.970575
iter 300 value 92.835629
iter 310 value 92.782585
iter 320 value 92.775068
iter 330 value 92.757020
iter 340 value 92.674846
iter 350 value 92.570052
iter 360 value 92.334816
iter 370 value 92.321175
iter 380 value 92.302224
iter 390 value 91.971135
iter 400 value 90.833979
iter 410 value 89.977295
iter 420 value 89.941450
iter 430 value 89.928643
iter 440 value 89.919600
iter 450 value 89.911005
iter 460 value 89.828482
iter 470 value 89.602676
iter 480 value 89.470807
iter 490 value 89.454347
iter 500 value 89.360322
iter 510 value 89.277129
iter 520 value 89.272248
iter 530 value 89.267633
```

```
iter 540 value 89.257607
iter 550 value 89.192259
iter 560 value 89.152085
iter 570 value 89.077939
iter 580 value 88.812195
iter 590 value 88.751789
iter 600 value 88.731080
final  value 88.680339
converged
> summary(model3)
Call:
multinom(formula = Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba +
    Fe, data = train, maxit = 1000)

Coefficients:
  (Intercept)         RI         Na         Mg          Al         Si          K
2  203.026613  171.42504  -4.099733  -5.800582  -0.06659444  -4.795857   -3.128619
3  848.442724 -786.05540   5.433639   4.239372   4.84957130   2.743805    4.141046
5    5.733115   12.99851 -81.581835 -77.628788 222.12921154   6.134533  254.091839
6  -14.058868  -23.50412 -15.226861 -27.438127   9.75009541   8.231017 -368.668129
7 -143.968089  933.95243 -11.264131 -18.845955 -12.60991726 -12.313294  -10.288273
          Ca          Ba           Fe
2  -4.495158   -6.789358    0.6437630
3   5.646475    2.223870    0.8239636
5  16.994975 -169.437118  272.7346400
6 -28.089356 -227.566859 -319.8861733
7 -17.173076   -6.244567  -95.8431176

Std. Errors:
  (Intercept)         RI         Na         Mg         Al        Si           K
2  0.03920231 0.06701881  0.5768557 0.85545632 1.65179176 0.1511549 2.289055e+00
3  0.06114337 0.09642064  0.8058941 1.12917704 1.76232222 0.2059976 2.938445e+00
5  0.06245625 0.08769877 12.2913630 8.40708139 1.89682777 2.6617805 2.304166e+00
6  0.01613113 0.02450860  0.2228231 0.03882437 0.01984018 1.1730231 4.111865e-10
7  0.10335390 0.16375551  2.0999998 2.96043069 6.87912454 0.7338250 5.774488e+00
          Ca           Ba           Fe
2 0.5632691 2.036229e+00 2.320576e+00
3 0.6197758 4.502377e+00 3.847483e+00
5 5.3353436 7.067090e+00 1.955795e-02
6 0.1572593 3.993370e-10 2.896739e-22
7 2.6269938 5.059777e+00 1.303330e+00

Residual Deviance: 177.3607
AIC: 277.3607
>
> predictions_train3 <- predict(model3, train)
> con_mat_train_model3 <- table(predicted = predictions_train3, actual = train$Type)
> con_mat_train_model3
         actual
predicted  1  2  3  5  6  7
        1 34 13  7  0  0  0
        2 15 44  4  0  0  1
        3  3  0  2  0  0  0
        5  0  0  0 10  0  0
        6  0  0  0  0  7  0
        7  0  0  0  0  0 21
> accuracy_train_model3 <- sum(diag(con_mat_train_model3)) / sum(con_mat_train_model3)
```

```
> accuracy_train_model3
[1] 0.7329193
> error_rate_train_model3 <- 1 - accuracy_train_model3
> error_rate_train_model3
[1] 0.2670807
>
> predictions_test3 <- predict(model3, test)
> con_mat_test_model3 <- table(predicted = predictions_test3, actual = test$Type)
> con_mat_test_model3
         actual
predicted  1  2  3  5  6  7
        1 13  4  2  0  0  0
        2  4 12  2  1  0  1
        3  1  0  0  0  0  0
        5  0  2  0  2  0  2
        6  0  1  0  0  2  0
        7  0  0  0  0  0  4
> accuracy_test_model3 <- sum(diag(con_mat_test_model3)) / sum(con_mat_test_model3)
> accuracy_test_model3
[1] 0.6226415
> error_rate_test_model3 <- 1 - accuracy_train_model3
> error_rate_test_model3
[1] 0.2670807
>
>
> model4 <- multinom(Type ~ Si + K + Ca + Ba + Fe, data = train, maxit = 1000)
# weights:  42 (30 variable)
initial  value 288.473275
iter  10 value 205.965189
iter  20 value 155.373994
iter  30 value 150.953991
iter  40 value 145.643947
iter  50 value 143.170483
iter  60 value 142.438705
iter  70 value 141.691530
iter  80 value 141.064968
iter  90 value 140.177861
iter 100 value 138.642205
iter 110 value 138.399477
iter 120 value 136.553486
iter 130 value 136.513584
iter 140 value 136.300029
iter 150 value 135.732249
iter 160 value 135.716372
iter 170 value 135.311145
iter 180 value 135.104716
final  value 135.083329
converged
> summary(model4)
Call:
multinom(formula = Type ~ Si + K + Ca + Ba + Fe, data = train,
    maxit = 1000)

Coefficients:
  (Intercept)         Si             K          Ca          Ba          Fe
2    5.918628 -0.1850184    4.4373011  0.6122962   1.2768883  -0.3208501
3   46.376637 -0.6368082   -0.3132222 -0.1581747  -1.0258372  -1.2687962
```

```
5  -83.639168  0.6612129   17.9719726  2.5082157  -0.1476698   3.6152259
6 -261.603409  3.8580477 -167.6912440 -1.5795150 -34.5116136 -82.6154120
7 -321.544481  4.4026375    0.1117102 -0.3228178  10.4514431 -34.7591492

Std. Errors:
  (Intercept)         Si           K        Ca           Ba           Fe
2 0.181144854 0.03732824 1.432751530 0.2513852 2.203916e+00 1.996694e+00
3 0.033493868 0.05670796 1.742530391 0.4103424 3.760468e+00 3.395902e+00
5 0.088578009 0.10633811 4.062301615 0.5524033 2.428180e+00 4.237475e+00
6 0.009440163 0.22062122 0.005634213 1.6293712 1.514334e-06 2.760710e-07
7 0.061870446 0.10700022 2.844427034 0.7900601 2.126609e+00 1.150811e-01

Residual Deviance: 270.1667
AIC: 330.1667
>
> predictions_train4 <- predict(model4, train)
> con_mat_train_model4 <- table(predicted = predictions_train4, actual = train$Type)
> #con_mat_train_model4
> accuracy_train_model4 <- sum(diag(con_mat_train_model4)) / sum(con_mat_train_model4)
> accuracy_train_model4
[1] 0.5900621
> error_rate_train_model4 <- 1 - accuracy_train_model4
> error_rate_train_model4
[1] 0.4099379
>
> predictions_test4 <- predict(model4, test)
> con_mat_test_model4 <- table(predicted = predictions_test4, actual = test$Type)
> #con_mat_test_model4
> accuracy_test_model4 <- sum(diag(con_mat_test_model4)) / sum(con_mat_test_model4)
> accuracy_test_model4
[1] 0.5471698
> error_rate_test_model4 <- 1 - accuracy_train_model4
> error_rate_test_model4
[1] 0.4099379
>
```

**NOTE:** The best model is Model-4, because it is having the lowest AIC value.

# Lab 12: Naïve Bayes Classifier

## Problem statement:

Task1: Use iris dataset which is available with R by default data (iris) and perform following operations:

- Develop Naïve Bayes classifier (Dependent variable (Species/Class) and rest of all are independent features)
- Observe the priori probabilities of all available classes.
- Observe the Conditional probabilities of all the classes against each and every independent features.
- Measured and display the confusion matrix
- Calculate the accuracy of model

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 12
#Task: Naive Bayes Implementation

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()
d <- read.csv("iris.csv")
head(d)
nrow(d)
summary(d)

#converting as a factor to class
d$class=factor(d$class)
#Finding structure of iris data
str(d)

# Creating table for class variable
table(d$class)

sample_iris=sample(150,110,replace = FALSE)

#creating training and test dataset
iris_training=d[sample_iris,]
iris_test=d[-sample_iris,]
#creating levels
iris_training_labels=d[sample_iris,]$class
iris_test_labels=d[-sample_iris,]$class

table(iris_training$class)
table(iris_test$class)

library(e1071)
iris_classifier=naiveBayes(class ~ ., data = iris_training)
class(iris_classifier)
print(iris_classifier)
```

```
summary(iris_classifier)
#Evaluvating model performance
iris_test_pred=predict(iris_classifier,iris_test)
iris_test_pred

#install.packages("gmodels")
#library(gmodels)
conf_matrix <- table(iris_test_pred, iris_test$class)
conf_matrix
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
accuracy
```

## Output:

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 12
> #Task: Naive Bayes Implementation
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
> d <- read.csv("iris.csv")
> head(d)
  length_sepal width_sepal length_petal width_petal      class
1          5.1         3.5          1.4         0.2 Iris-setosa
2          4.9         3.0          1.4         0.2 Iris-setosa
3          4.7         3.2          1.3         0.2 Iris-setosa
4          4.6         3.1          1.5         0.2 Iris-setosa
5          5.0         3.6          1.4         0.2 Iris-setosa
6          5.4         3.9          1.7         0.4 Iris-setosa
> nrow(d)
[1] 150
> summary(d)
  length_sepal    width_sepal     length_petal    width_petal
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
         class
 Iris-setosa    :50
 Iris-versicolor:50
 Iris-virginica :50



>
> #converting as a factor to class
> d$class=factor(d$class)
> #Finding structure of iris data
> str(d)
```

```
'data.frame':          150 obs. of  5 variables:
 $ length_sepal: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ width_sepal : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ length_petal: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ width_petal : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ class       : Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...
>
> # Creating table for class variable
> table(d$class)

   Iris-setosa Iris-versicolor  Iris-virginica
            50             50             50
>
> sample_iris=sample(150,110,replace = FALSE)
>
> #creating training and test dataset
> iris_training=d[sample_iris,]
> iris_test=d[-sample_iris,]
> #creating levels
> iris_training_labels=d[sample_iris,]$class
> iris_test_labels=d[-sample_iris,]$class
>
> table(iris_training$class)

   Iris-setosa Iris-versicolor  Iris-virginica
            35             36             39
> table(iris_test$class)

   Iris-setosa Iris-versicolor  Iris-virginica
            15             14             11
>
> library(e1071)
> iris_classifier=naiveBayes(class ~ ., data = iris_training)
> class(iris_classifier)
[1] "naiveBayes"
> print(iris_classifier)

Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
   Iris-setosa Iris-versicolor  Iris-virginica
     0.3181818      0.3272727      0.3545455

Conditional probabilities:
                 length_sepal
Y                     [,1]     [,2]
  Iris-setosa     5.014286 0.3614862
  Iris-versicolor 5.913889 0.5259836
  Iris-virginica  6.684615 0.5828869

                 width_sepal
```

```
Y                 [,1]     [,2]
 Iris-setosa     3.431429 0.3771187
 Iris-versicolor 2.769444 0.3087481
 Iris-virginica  3.007692 0.3351234

             length_petal
Y                 [,1]     [,2]
 Iris-setosa     1.465714 0.1781322
 Iris-versicolor 4.255556 0.4494088
 Iris-virginica  5.617949 0.5305765

             width_petal
Y                 [,1]     [,2]
 Iris-setosa     0.2285714 0.1045197
 Iris-versicolor 1.3277778 0.1830084
 Iris-virginica  2.0461538 0.2798930

> summary(iris_classifier)
      Length Class  Mode
apriori 3     table  numeric
tables  4     -none- list
levels  3     -none- character
call    4     -none- call
> #Evaluvating model performance
> iris_test_pred=predict(iris_classifier,iris_test)
> iris_test_pred
 [1] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
 [6] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
[11] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
[16] Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor
[21] Iris-versicolor Iris-virginica  Iris-versicolor Iris-versicolor Iris-versicolor
[26] Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-virginica
[31] Iris-versicolor Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
[36] Iris-virginica  Iris-versicolor Iris-virginica  Iris-virginica  Iris-virginica
Levels: Iris-setosa Iris-versicolor Iris-virginica
>
> #install.packages("gmodels")
> #library(gmodels)
> conf_matrix <- table(iris_test_pred, iris_test$class)
> conf_matrix

iris_test_pred    Iris-setosa Iris-versicolor Iris-virginica
 Iris-setosa           15            0            0
 Iris-versicolor        0           13            2
 Iris-virginica         0            1            9
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> accuracy
[1] 0.925
```

# Lab 13: k-Nearest Neighbor Classifier

## Problem statement:

Task1: Refer attached dataset wisc_bc_data.csv (Cancer dataset ) as

- Total number of columns: 32
- Column-1 PatientID -- should be discarded
- Column-2- class label (Two classes 'B' and 'M')
- Remaining columns are numerical values representing values against various test
- Convert column-2 (diagnosis) as factor
- Normalized all numeric feature and scale values between 0 to 1
- Create test and train dataset as per your choice
- Develop KNN classification with k=21
- Train KNN model with train dataset
- Test the KNN model with test dataset
- Observe the confusion matrix
- Repeat the same procedure with K=1 to K= number of training dataset and observe the classifier performance

Task-2: Repeat this with different scaling formula (Z-score standardization)

- X_new= (x-Mean(x))/ StdDev (x)

## Source Code:

```
#Author: Ashish Upadhyay
#Branch: Computer Science and Engineering
#Semester: 6th
#Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
#Subject: Machine Learning Lab 13
#Task: k-Nearest Neighbour Implementation

setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
getwd()

data_set <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
stringsAsFactors = FALSE
#head(data_set)
nrow(data_set)
str(data_set)
data_set <- data_set[-1]
str(data_set)
table(data_set$diagnosis)
data_set$diagnosis <- as.factor(data_set$diagnosis)

#Normalization
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# One could also use sequence such as df[1:2]
dfNorm <- as.data.frame(lapply(data_set[2:31], normalize))
head(dfNorm)

data_train <- dfNorm[1:400,]
```

```
data_test <- dfNorm[401:569,]

data_train_labels <- data_set[1:400, 1]
data_test_labels <- data_set[401:569, 1]

#install.packages("class")
library(class)

data_test_pred <- knn(train = data_train, test = data_test,cl = data_train_labels, k=21)
#summary(data_test_pred)

#install.packages("gmodels")
library(gmodels)
CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq=FALSE)

#Z-score standardization
stad <- function(x) {
 return ((x - mean(x)) / sqrt(var(x)))
}
# One could also use sequence such as df[1:2]
dfNorm <- as.data.frame(lapply(data_set[2:31], stad))
head(dfNorm)

data_train <- dfNorm[1:400,]
data_test <- dfNorm[401:569,]

data_train_labels <- data_set[1:400, 1]
data_test_labels <- data_set[401:569, 1]

#install.packages("class")
library(class)

data_test_pred <- knn(train = data_train, test = data_test,cl = data_train_labels, k=21)
#summary(data_test_pred)

#install.packages("gmodels")
library(gmodels)
CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq=FALSE)
```

**Output:**

```
> #Author: Ashish Upadhyay
> #Branch: Computer Science and Engineering
> #Semester: 6th
> #Dr. SP Mukherjee International Institute of Information Technology, Naya Raipur
> #Subject: Machine Learning Lab 13
> #Task: k-Nearest Neighbour Implementation
>
> setwd("C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs")
> getwd()
[1] "C:/Users/Ashish Upadhyay/Documents/Semester6/MachineLearning/Lab Programs"
>
> data_set <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
```

```
> stringsAsFactors = FALSE
> #head(data_set)
> nrow(data_set)
[1] 569
> str(data_set)
'data.frame':  569 obs. of  32 variables:
 $ id                : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862
989 89827 ...
 $ diagnosis         : chr  "B" "B" "B" "B" ...
 $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean         : num  464 346 373 385 712 ...
 $ smoothness_mean   : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean    : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean       : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean     : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean    : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se         : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se        : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se      : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se           : num  17.4 27.1 13.5 26.3 17.7 ...
 $ smoothness_se     : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se    : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ concavity_se      : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se         : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se       : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se      : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst      : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst     : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst   : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst        : num  549 425 471 434 819 ...
 $ smoothness_worst  : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst   : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst      : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst    : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst   : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
> data_set <- data_set[-1]
> str(data_set)
'data.frame':  569 obs. of  31 variables:
 $ diagnosis         : chr  "B" "B" "B" "B" ...
 $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean         : num  464 346 373 385 712 ...
 $ smoothness_mean   : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean    : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean       : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean     : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean    : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se         : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se        : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se      : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se           : num  17.4 27.1 13.5 26.3 17.7 ...
```

```
 $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst       : num  549 425 471 434 819 ...
 $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
> table(data_set$diagnosis)

  B   M
357 212
> data_set$diagnosis <- as.factor(data_set$diagnosis)
>
> #Normalization
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x)))
+ }
> # One could also use sequence such as df[1:2]
> dfNorm <- as.data.frame(lapply(data_set[2:31], normalize))
> head(dfNorm)
  radius_mean texture_mean perimeter_mean  area_mean smoothness_mean
1   0.2526859    0.0906324      0.2422777 0.13599152       0.4529205
2   0.1712812    0.3124789      0.1761454 0.08606575       0.3994764
3   0.1921056    0.2407846      0.1874784 0.09743372       0.4971563
4   0.2034644    0.1244505      0.2018520 0.10235419       0.5756974
5   0.3885182    0.1183632      0.3721927 0.24106045       0.2437483
6   0.2171896    0.3155225      0.2101444 0.11291622       0.2963799
  compactness_mean concavity_mean points_mean symmetry_mean dimension_mean
1        0.1546838     0.09341612  0.18389662     0.4540404      0.2019798
2        0.2923747     0.14964855  0.13131213     0.4353535      0.3148694
3        0.1799276     0.07136832  0.12326044     0.3303030      0.2830666
4        0.2890007     0.10859888  0.23836978     0.3590909      0.2266217
5        0.1532421     0.07949859  0.13205765     0.3338384      0.1154170
6        0.1774124     0.12851453  0.07097416     0.4904040      0.2676917
   radius_se texture_se perimeter_se    area_se smoothness_se compactness_se
1 0.04508419 0.06749470   0.04301937 0.01985065     0.2152497     0.07170968
2 0.12275937 0.18493635   0.12594826 0.03791198     0.1957032     0.25203533
3 0.03085280 0.22692716   0.02756443 0.01258503     0.1171092     0.05334665
4 0.08216549 0.21720297   0.05154785 0.03647380     0.3248802     0.24580166
5 0.02418975 0.01155852   0.02737596 0.02039231     0.1121460     0.09461652
6 0.06333514 0.23864038   0.06827498 0.02521115     0.1891763     0.13682519
  concavity_se points_se symmetry_se dimension_se radius_worst texture_worst
1   0.04250000 0.2350824   0.1598188   0.04675041    0.1981501    0.09648188
2   0.08469697 0.2585717   0.3821410   0.08371682    0.1405194    0.29104478
3   0.02666667 0.1417503   0.1308324   0.04502301    0.1593739    0.38432836
4   0.05522727 0.3722296   0.1114144   0.08800077    0.1419424    0.09994670
5   0.03916667 0.1734230   0.1208420   0.03013280    0.2942014    0.09888060
6   0.11229798 0.1666793   0.1519390   0.08444233    0.1828531    0.39872068
```

```
  perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
1       0.1820808 0.08943669        0.4446279        0.09635106      0.09920128
2       0.1388017 0.05888714        0.3310440        0.21752966      0.15303514
3       0.1470193 0.07034015        0.4340619        0.11730749      0.08522364
4       0.1300862 0.06114825        0.4327412        0.15029446      0.06924121
5       0.2693859 0.15579532        0.2735918        0.14204771      0.10878594
6       0.1793914 0.08240759        0.3548174        0.16145181      0.20447284
  points_worst symmetry_worst dimension_worst
1    0.3227148      0.2487680      0.08310376
2    0.2723711      0.2710428      0.13662600
3    0.2553608      0.2824759      0.15590975
4    0.2959107      0.1058545      0.08395645
5    0.2810309      0.1817465      0.08277581
6    0.2290034      0.2897694      0.18234291
> 
> data_train <- dfNorm[1:400,]
> data_test <- dfNorm[401:569,]
> 
> data_train_labels <- data_set[1:400, 1]
> data_test_labels <- data_set[401:569, 1]
> 
> #install.packages("class")
> library(class)
> 
> data_test_pred <- knn(train = data_train, test = data_test,cl = data_train_labels, k=21
)
> #summary(data_test_pred)
> 
> #install.packages("gmodels")
> library(gmodels)
> CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  169


                 | data_test_pred 
data_test_labels |         B |         M | Row Total | 
-----------------|-----------|-----------|-----------|
               B |       106 |         0 |       106 | 
                 |     1.000 |     0.000 |     0.627 | 
                 |     0.972 |     0.000 |           | 
                 |     0.627 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
               M |         3 |        60 |        63 | 
                 |     0.048 |     0.952 |     0.373 | 
                 |     0.028 |     1.000 |           | 
                 |     0.018 |     0.355 |           | 
```

```
-----------------|-----------|-----------|-----------|
    Column Total |       109 |        60 |       169 |
                 |     0.645 |     0.355 |           |
-----------------|-----------|-----------|-----------|


> 
> #Z-score standardization
> stad <- function(x) {
+   return ((x - mean(x)) / sqrt(var(x)))
+ }
> # One could also use sequence such as df[1:2]
> dfNorm <- as.data.frame(lapply(data_set[2:31], stad))
> head(dfNorm)
  radius_mean texture_mean perimeter_mean  area_mean smoothness_mean
1  -0.5128453  -1.60418301     -0.5399006 -0.5421468       0.4578825
2  -1.0009202  -0.07896900     -0.9337442 -0.8766033       0.0369535
3  -0.8760638  -0.57187353     -0.8662517 -0.8004484       0.8062867
4  -0.8079604  -1.37168088     -0.7806514 -0.7674858       1.4248817
5   0.3015589  -1.41353126      0.2337944  0.1617181      -1.1895712
6  -0.7256686  -0.05804381     -0.7312666 -0.6967299      -0.7750414
  compactness_mean concavity_mean points_mean symmetry_mean dimension_mean
1       -0.6538379     -0.6137661 -0.30717196     0.5376080    -0.45997776
2        0.1961461     -0.3127117 -0.57983238     0.4026419     0.29919003
3       -0.4980044     -0.7318045 -0.62158190    -0.3560868     0.08532000
4        0.1753178     -0.5324814 -0.02471844    -0.1481659    -0.29426389
5       -0.6627373     -0.6882771 -0.57596668    -0.3305526    -1.04210082
6       -0.5135309     -0.4258580 -0.89269604     0.8002448    -0.01807412
   radius_se  texture_se perimeter_se    area_se smoothness_se compactness_se
1 -0.6100407 -0.99928403   -0.5915654 -0.5035518    0.33439304     -0.7637928
2  0.1634542 -0.03598928    0.2789225 -0.2909823    0.14288710      0.5769353
3 -0.7517580  0.30843301   -0.7537927 -0.5890632   -0.62713327     -0.9003226
4 -0.2407825  0.22867206   -0.5020436 -0.3079088    1.40849153      0.5305878
5 -0.8181090 -1.45809077   -0.7557711 -0.4971769   -0.67575913     -0.5934796
6 -0.4282964  0.40450870   -0.3264623 -0.4404624    0.07894077     -0.2796565
  concavity_se   points_se symmetry_se dimension_se radius_worst texture_worst
1  -0.49902890  0.09948696  -0.1575418   -0.5846041   -0.5729467    -1.6330626
2   0.05453788  0.30045012   1.7538168   -0.1802309   -0.9081255    -0.4453479
3  -0.70674066 -0.69901746  -0.4067442   -0.6035000   -0.7984682     0.1241043
4  -0.33206441  1.27285250  -0.5736857   -0.1333690   -0.8998495    -1.6119115
5  -0.54275769 -0.42804133  -0.4926344   -0.7663830   -0.0143154    -1.6184195
6   0.41662554 -0.48573721  -0.2252861   -0.1722946   -0.6619139     0.2119626
  perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
1     -0.60385945 -0.5822061        0.2685394       -0.81141409     -0.70935407
2     -0.86247083 -0.8005226       -0.4847751       -0.01757408     -0.38628525
3     -0.81336741 -0.7186759        0.1984636       -0.67412871     -0.79323693
4     -0.91455023 -0.7843640        0.1897041       -0.45803135     -0.88915098
5     -0.08217273 -0.1079870       -0.8658121       -0.51205569     -0.65183440
6     -0.61992966 -0.6324382       -0.3271047       -0.38493959     -0.07759635
  points_worst symmetry_worst dimension_worst
1   -0.3148560    -0.11921566     -0.89893012
2   -0.5377296     0.06343283     -0.44713458
3   -0.6130350     0.15718162     -0.28435531
4   -0.4335191    -1.29107549     -0.89173240
5   -0.4993923    -0.66877751     -0.90169847
6   -0.7297203     0.21698688     -0.06122589
> 
```

```
> data_train <- dfNorm[1:400,]
> data_test <- dfNorm[401:569,]
>
> data_train_labels <- data_set[1:400, 1]
> data_test_labels <- data_set[401:569, 1]
>
> #install.packages("class")
> library(class)
>
> data_test_pred <- knn(train = data_train, test = data_test,cl = data_train_labels, k=21
)
> #summary(data_test_pred)
>
> #install.packages("gmodels")
> library(gmodels)
> CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  169


                 | data_test_pred
data_test_labels |         B |         M | Row Total |
-----------------|-----------|-----------|-----------|
               B |       106 |         0 |       106 |
                 |     1.000 |     0.000 |     0.627 |
                 |     0.938 |     0.000 |           |
                 |     0.627 |     0.000 |           |
-----------------|-----------|-----------|-----------|
               M |         7 |        56 |        63 |
                 |     0.111 |     0.889 |     0.373 |
                 |     0.062 |     1.000 |           |
                 |     0.041 |     0.331 |           |
-----------------|-----------|-----------|-----------|
    Column Total |       113 |        56 |       169 |
                 |     0.669 |     0.331 |           |
-----------------|-----------|-----------|-----------|
```