Michael Dang – 16257750

COMP-SCI 5565

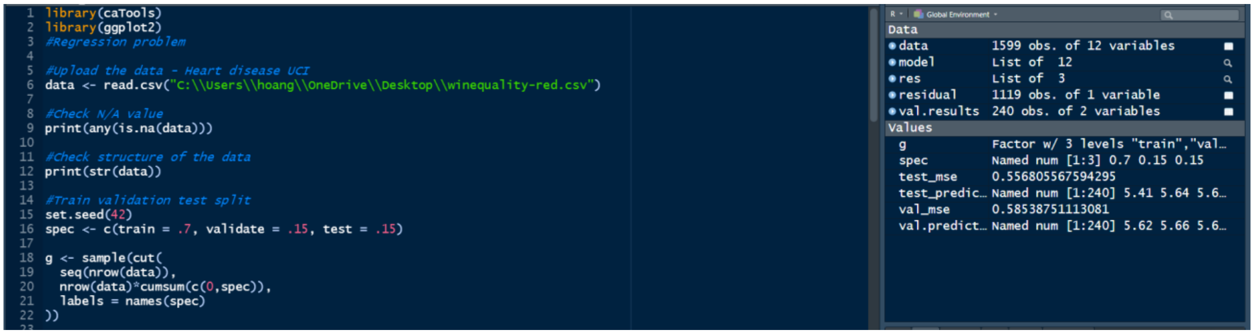
Final Project

12/16/2023

1. Regression

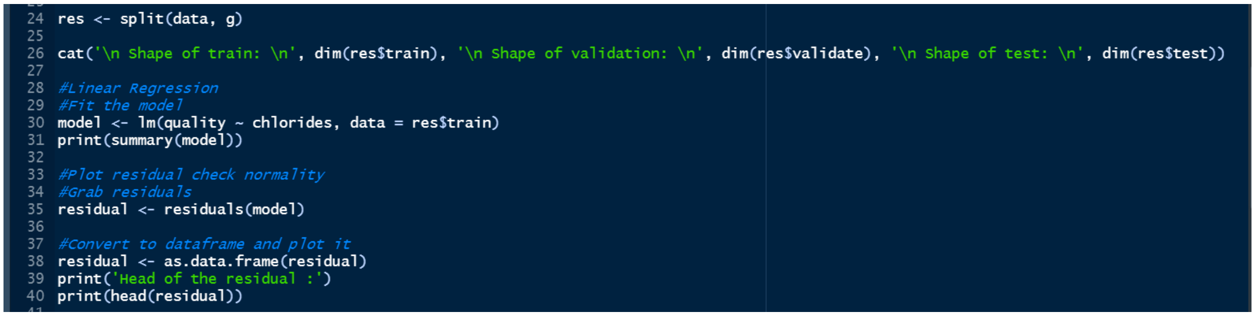
Wine quality dataset

<u>Linear Regression</u>

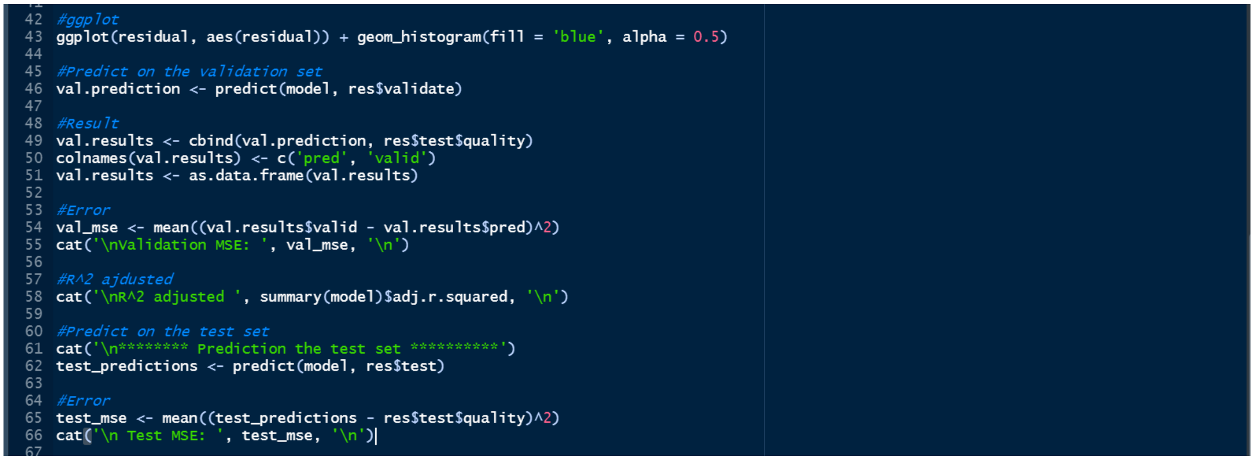


```
library(caTools)
library(ggplot2)
#Regression problem

#Upload the data - Heart disease UCI
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\winequality-red.csv")

#Check N/A value
print(any(is.na(data)))

#Check structure of the data
print(str(data))

#Train validation test split
set.seed(42)
spec <- c(train = .7, validate = .15, test = .15)

g <- sample(cut(
  seq(nrow(data)),
  nrow(data)*cumsum(c(0,spec)),
  labels = names(spec)
))
```

```
Data
data          1599 obs. of 12 variables
model         List of  12
res           List of  3
residual      1119 obs. of 1 variable
val.results   240 obs. of 2 variables
Values
g             Factor w/ 3 levels "train","val…
spec          Named num [1:3] 0.7 0.15 0.15
test_mse      0.556805567594295
test_predic…  Named num [1:240] 5.41 5.64 5.6…
val_mse       0.58538751113081
val.predict…  Named num [1:240] 5.62 5.66 5.6…
```

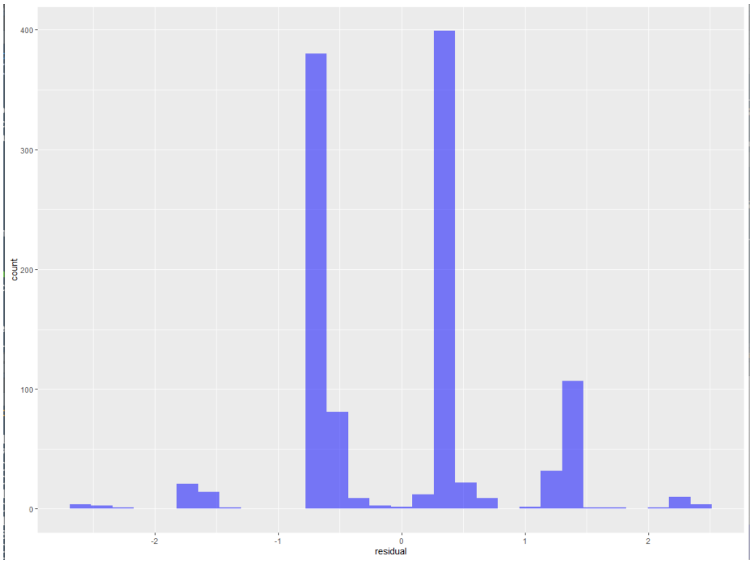


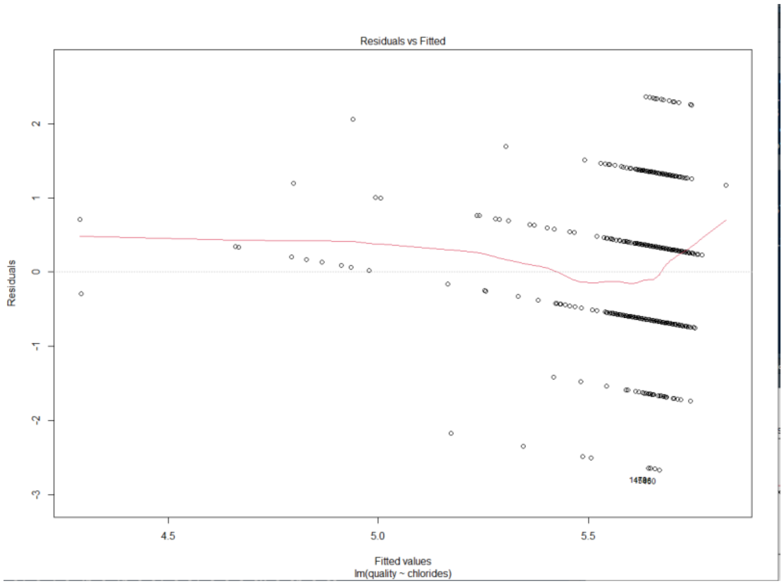
```
res <- split(data, g)

cat('\n Shape of train: \n', dim(res$train), '\n Shape of validation: \n', dim(res$validate), '\n Shape of test: \n', dim(res$test))

#Linear Regression
#Fit the model
model <- lm(quality ~ chlorides, data = res$train)
print(summary(model))

#Plot residual check normality
#Grab residuals
residual <- residuals(model)

#Convert to dataframe and plot it
residual <- as.data.frame(residual)
print('Head of the residual :')
print(head(residual))
```

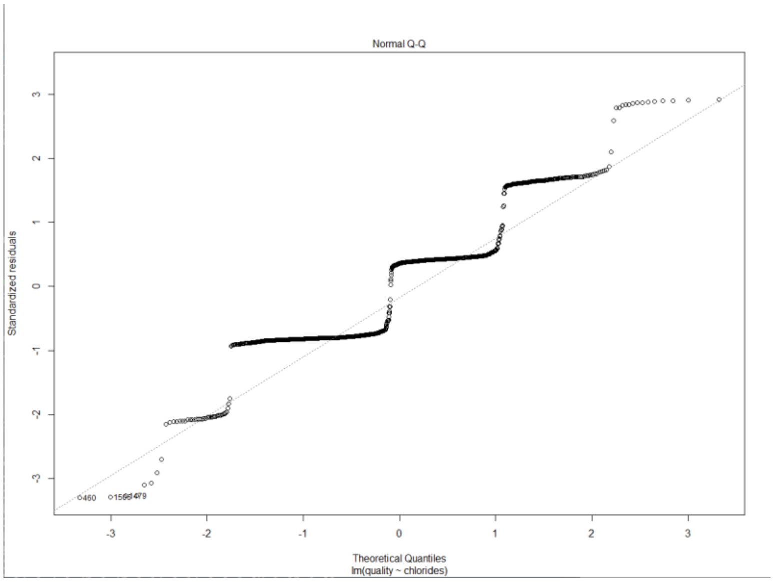


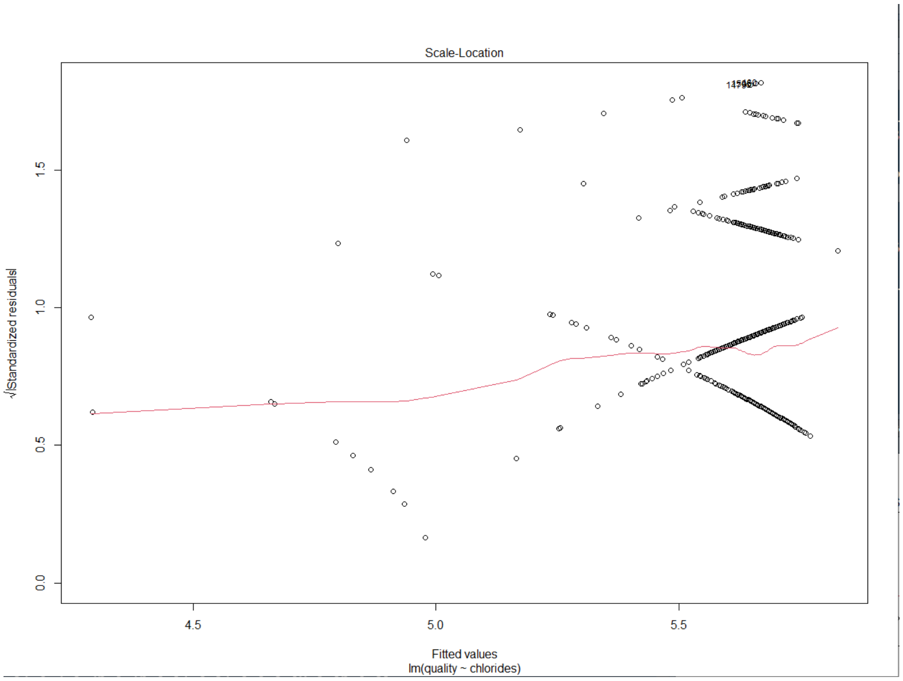
```
#ggplot
ggplot(residual, aes(residual)) + geom_histogram(fill = 'blue', alpha = 0.5)

#Predict on the validation set
val.prediction <- predict(model, res$validate)

#Result
val.results <- cbind(val.prediction, res$test$quality)
colnames(val.results) <- c('pred', 'valid')
val.results <- as.data.frame(val.results)

#Error
val_mse <- mean((val.results$valid - val.results$pred)^2)
cat('\nValidation MSE: ', val_mse, '\n')

#R^2 ajdusted
cat('\nR^2 adjusted ', summary(model)$adj.r.squared, '\n')

#Predict on the test set
cat('\n******** Prediction the test set **********')
test_predictions <- predict(model, res$test)

#Error
test_mse <- mean((test_predictions - res$test$quality)^2)
cat('\n Test MSE: ', test_mse, '\n')
```

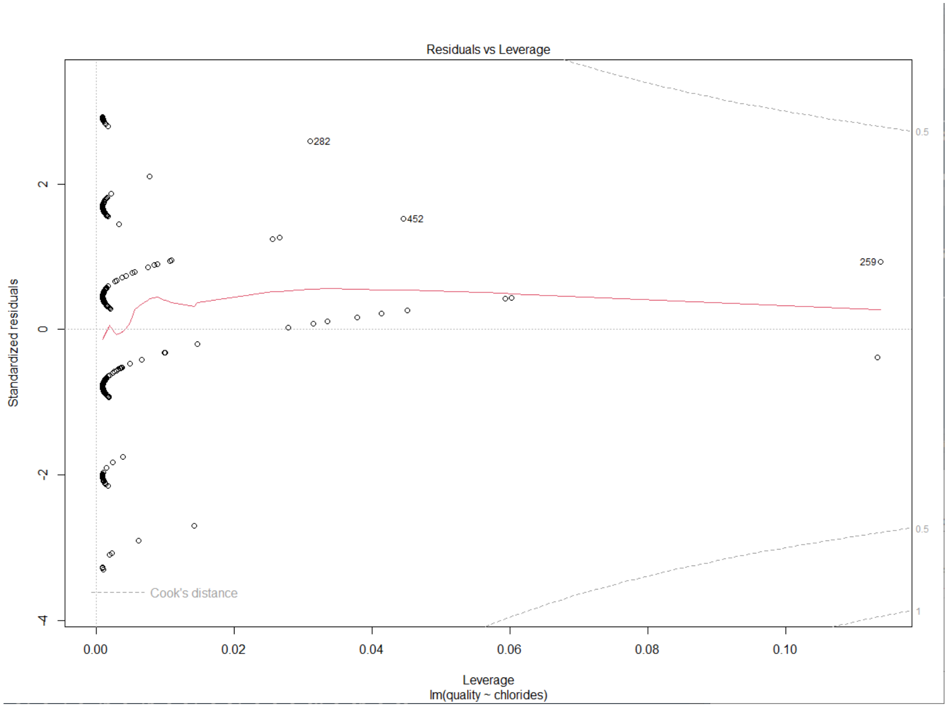
We want a histogram of our residuals to be normally distributed, something with a strong bimodal distribution may be a warning that our data was not a good fit for linear regression.





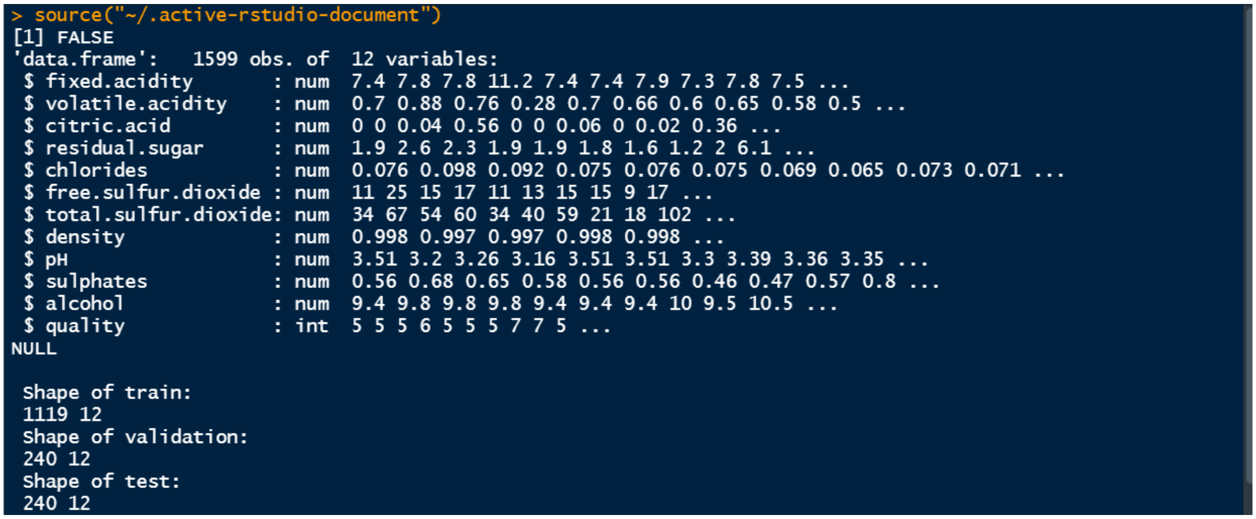






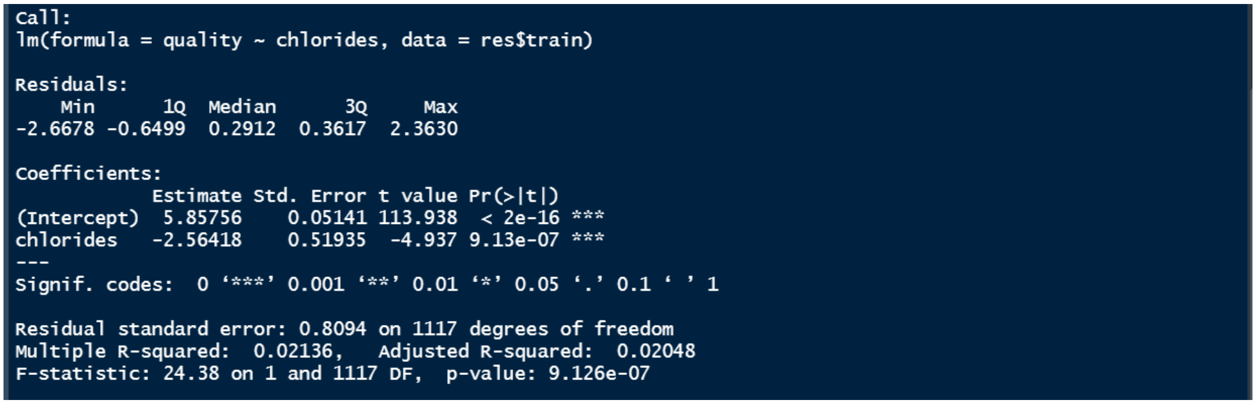
Looks like the data is following a normal distribution.

Code output



```
> source("~/.active-rstudio-document")
[1] FALSE
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
NULL

 Shape of train:
 1119 12
 Shape of validation:
 240 12
 Shape of test:
 240 12
```

Model summary

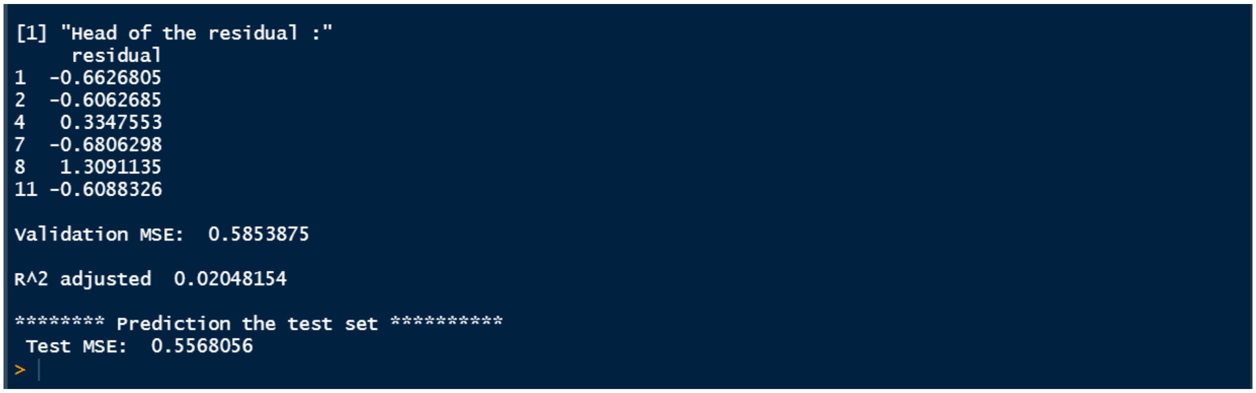


```
Call:
lm(formula = quality ~ chlorides, data = res$train)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.6678 -0.6499  0.2912  0.3617  2.3630 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.85756    0.05141 113.938  < 2e-16 ***
chlorides   -2.56418    0.51935  -4.937 9.13e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8094 on 1117 degrees of freedom
Multiple R-squared:  0.02136,	Adjusted R-squared:  0.02048 
F-statistic: 24.38 on 1 and 1117 DF,  p-value: 9.126e-07
```

Head of the residual column and MSE for validation and test set.

Notice: I’m using the R^2 adjusted instead of R^2 because it is more accurate

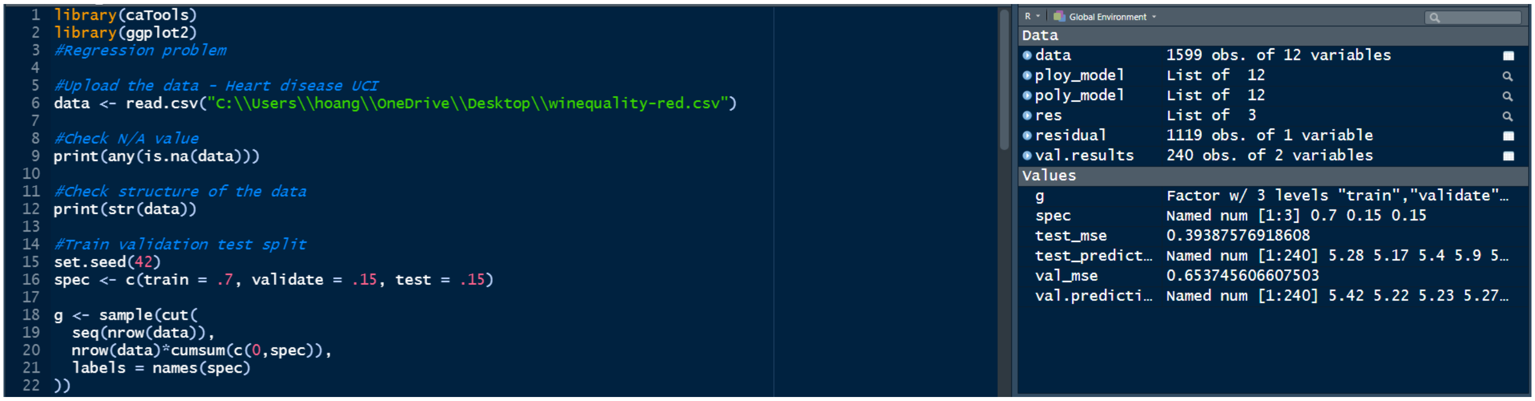
(My statistics professor said so)



```
[1] "Head of the residual :"
     residual
1  -0.6626805
2  -0.6062685
4   0.3347553
7  -0.6806298
8   1.3091135
11 -0.6088326

Validation MSE:  0.5853875

R^2 adjusted  0.02048154

******** Prediction the test set **********
 Test MSE:  0.5568056
> 
```

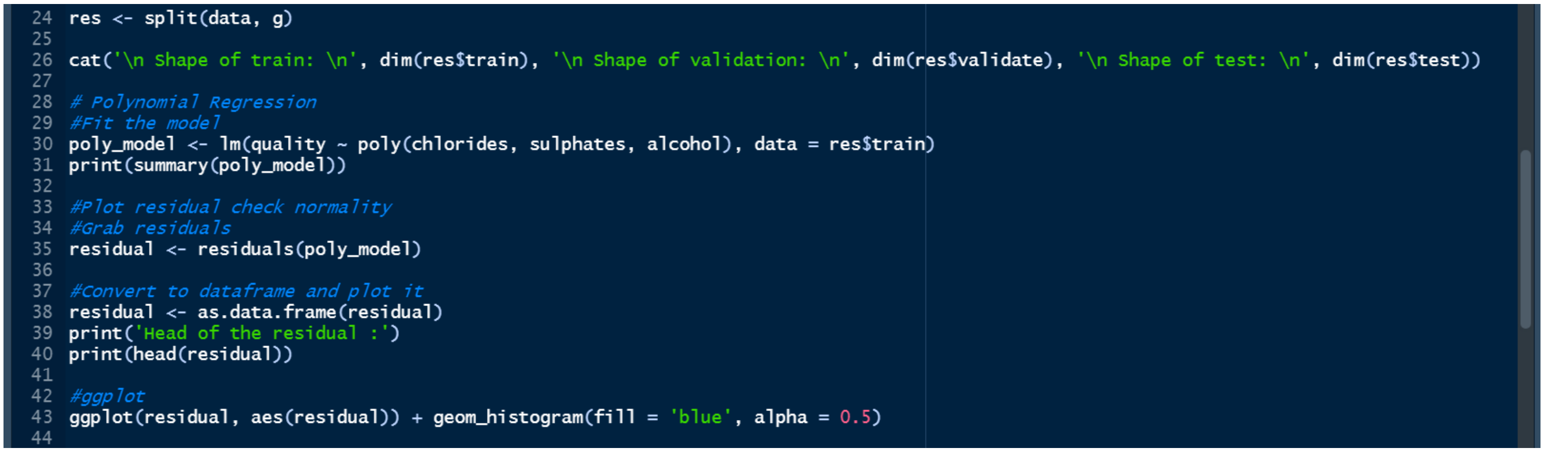
Interpreting the simple linear regression result shows that the regression model shows good MSE but a bad R-square value. Indicate that the model has a low error when predicting values but there is very little correlation between the variables.

<u>Polynomial Regression</u>

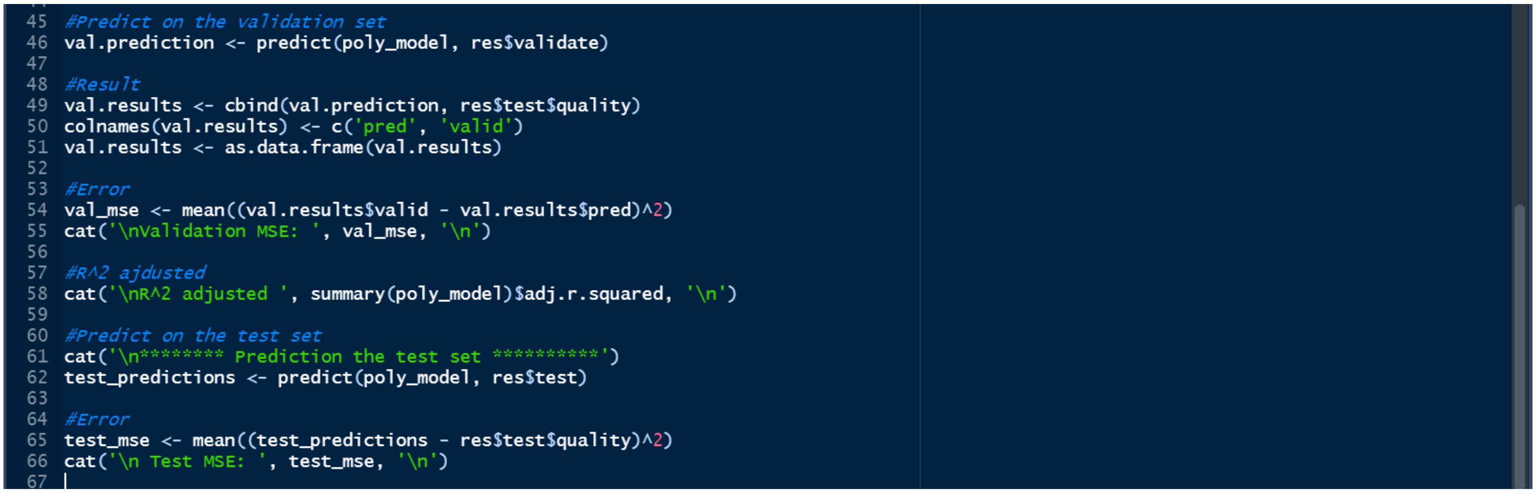


```
library(caTools)
library(ggplot2)
#Regression problem

#Upload the data - Heart disease UCI
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\winequality-red.csv")

#Check N/A value
print(any(is.na(data)))

#Check structure of the data
print(str(data))

#Train validation test split
set.seed(42)
spec <- c(train = .7, validate = .15, test = .15)

g <- sample(cut(
  seq(nrow(data)),
  nrow(data)*cumsum(c(0,spec)),
  labels = names(spec)
))
```

```
Data
data            1599 obs. of 12 variables
ploy_model      List of  12
poly_model      List of  12
res             List of  3
residual        1119 obs. of 1 variable
val.results     240 obs. of 2 variables
Values
g               Factor w/ 3 levels "train","validate"…
spec            Named num [1:3] 0.7 0.15 0.15
test_mse        0.39387576918608
test_predict…   Named num [1:240] 5.28 5.17 5.4 5.9 5…
val_mse         0.653745606607503
val.predicti…   Named num [1:240] 5.42 5.22 5.23 5.27…
```

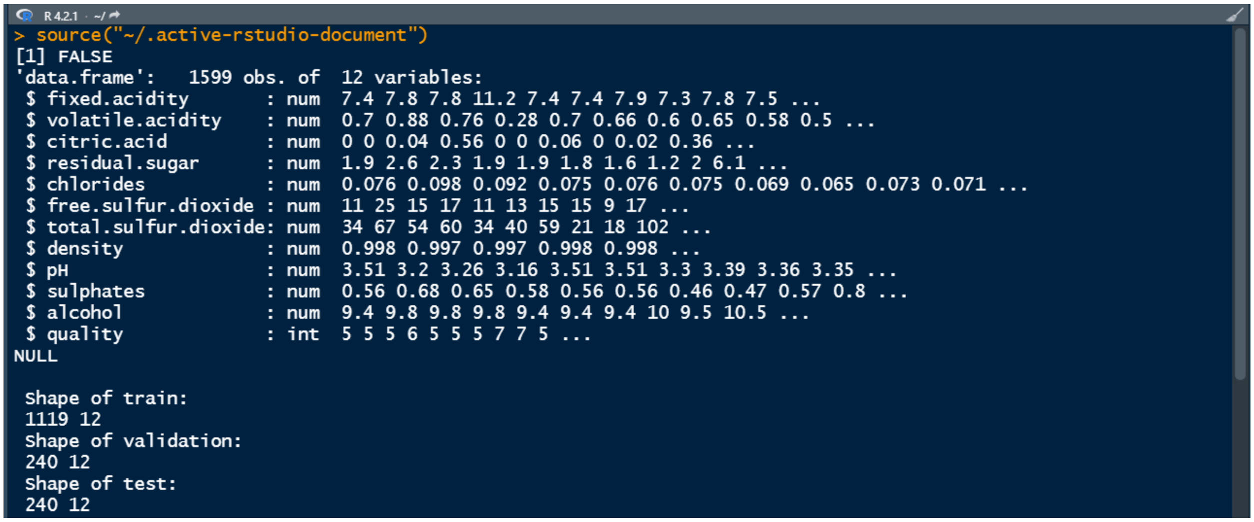


```r
res <- split(data, g)

cat('\n Shape of train: \n', dim(res$train), '\n Shape of validation: \n', dim(res$validate), '\n Shape of test: \n', dim(res$test))

# Polynomial Regression
#Fit the model
poly_model <- lm(quality ~ poly(chlorides, sulphates, alcohol), data = res$train)
print(summary(poly_model))

#Plot residual check normality
#Grab residuals
residual <- residuals(poly_model)

#Convert to dataframe and plot it
residual <- as.data.frame(residual)
print('Head of the residual :')
print(head(residual))

#ggplot
ggplot(residual, aes(residual)) + geom_histogram(fill = 'blue', alpha = 0.5)

```



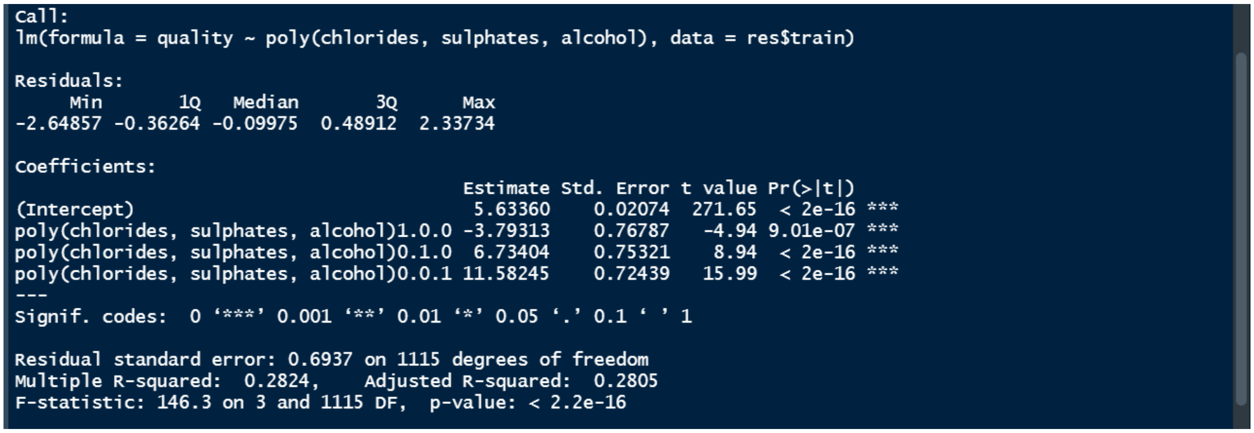
```r
#Predict on the validation set
val.prediction <- predict(poly_model, res$validate)

#Result
val.results <- cbind(val.prediction, res$test$quality)
colnames(val.results) <- c('pred', 'valid')
val.results <- as.data.frame(val.results)

#Error
val_mse <- mean((val.results$valid - val.results$pred)^2)
cat('\nValidation MSE: ', val_mse, '\n')

#R^2 ajdusted
cat('\nR^2 adjusted ', summary(poly_model)$adj.r.squared, '\n')

#Predict on the test set
cat('\n******** Prediction the test set **********')
test_predictions <- predict(poly_model, res$test)

#Error
test_mse <- mean((test_predictions - res$test$quality)^2)
cat('\n Test MSE: ', test_mse, '\n')

```

Code output



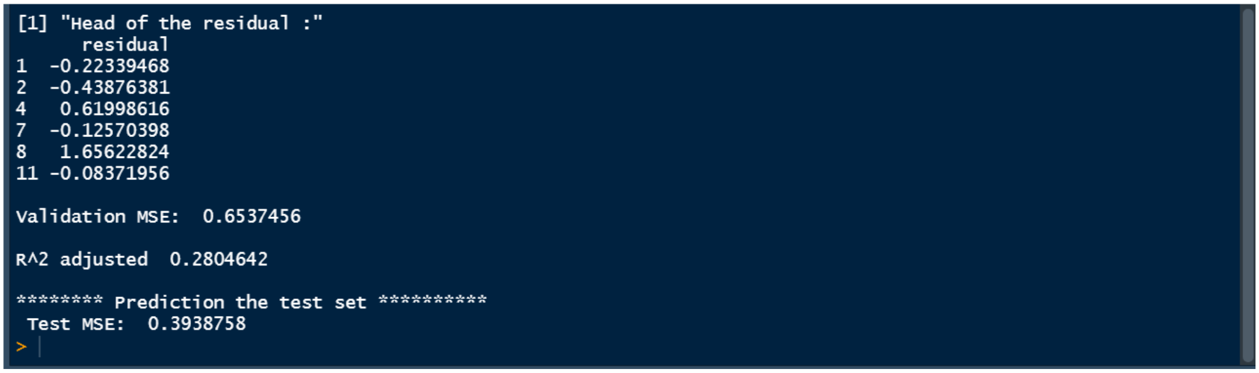
```
> source("~/.active-rstudio-document")
[1] FALSE
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
NULL

 Shape of train: 
 1119 12 
 Shape of validation: 
 240 12 
 Shape of test: 
 240 12
```

Model summary



```
Call:
lm(formula = quality ~ poly(chlorides, sulphates, alcohol), data = res$train)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.64857 -0.36264 -0.09975  0.48912  2.33734 

Coefficients:
                                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)                               5.63360    0.02074  271.65  < 2e-16 ***
poly(chlorides, sulphates, alcohol)1.0.0 -3.79313    0.76787   -4.94 9.01e-07 ***
poly(chlorides, sulphates, alcohol)0.1.0  6.73404    0.75321    8.94  < 2e-16 ***
poly(chlorides, sulphates, alcohol)0.0.1 11.58245    0.72439   15.99  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6937 on 1115 degrees of freedom
Multiple R-squared:  0.2824,	Adjusted R-squared:  0.2805 
F-statistic: 146.3 on 3 and 1115 DF,  p-value: < 2.2e-16
```

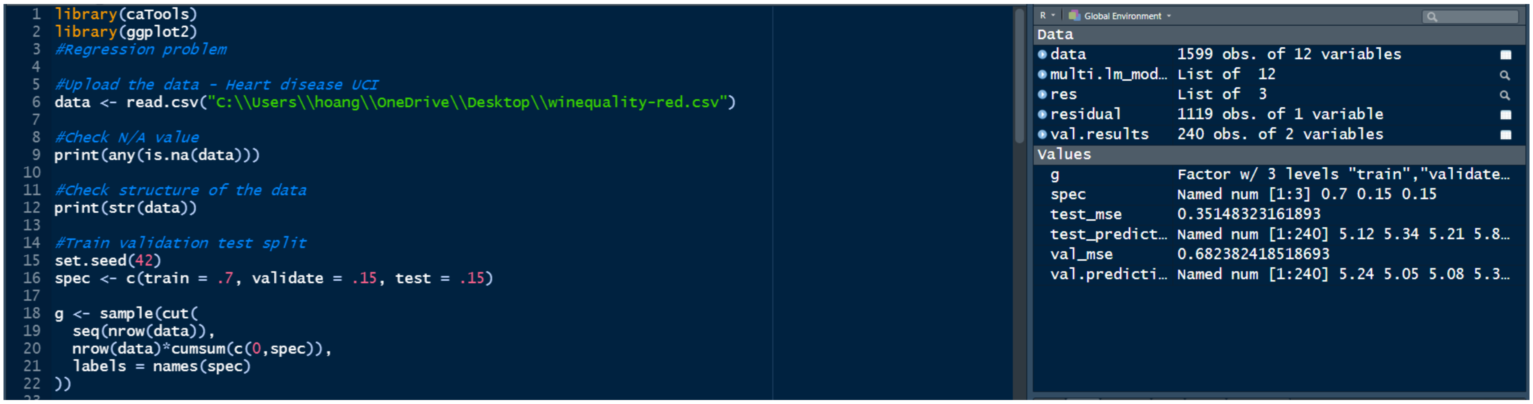
Model result



```
[1] "Head of the residual :"
      residual
1  -0.22339468
2  -0.43876381
4   0.61998616
7  -0.12570398
8   1.65622824
11 -0.08371956

Validation MSE:  0.6537456

R^2 adjusted  0.2804642

******** Prediction the test set **********
 Test MSE:  0.3938758
> 
```

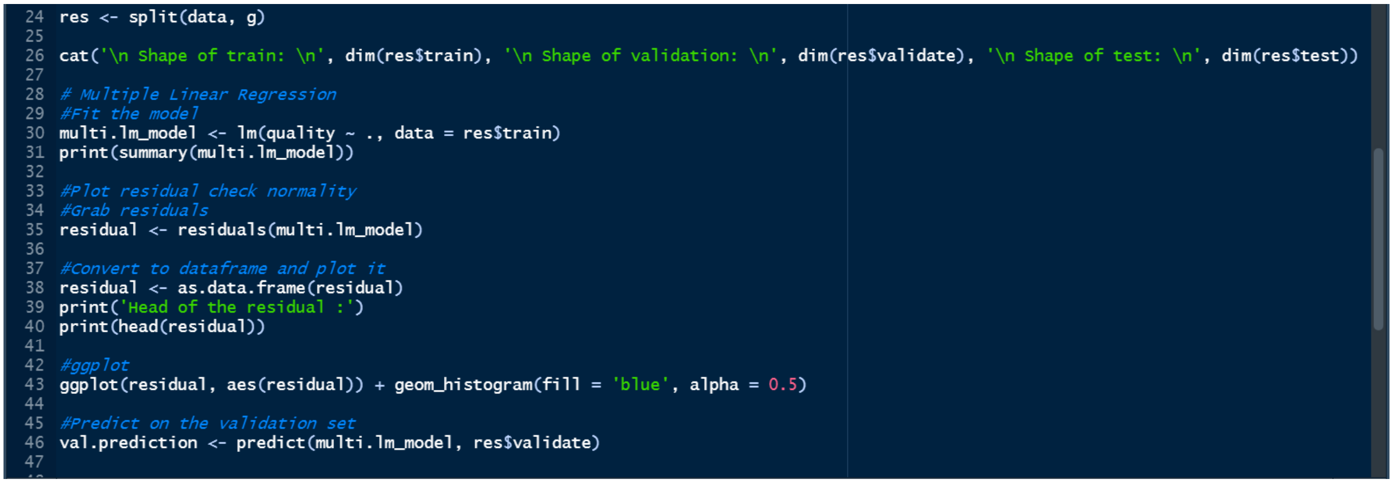
Polynomial regression has better MSE and slightly better R^2 adjusted than simple linear regression. The error on the test set performs much better, therefore the model is not overfitting.

Multiple Linear Regression

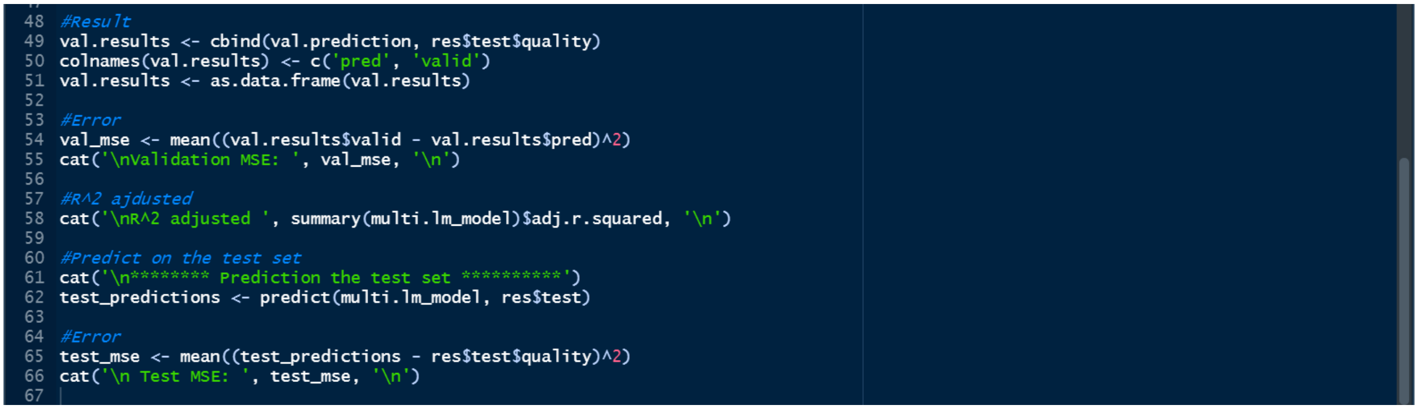


```
library(caTools)
library(ggplot2)
#Regression problem

#Upload the data - Heart disease UCI
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\winequality-red.csv")

#Check N/A value
print(any(is.na(data)))

#Check structure of the data
print(str(data))

#Train validation test split
set.seed(42)
spec <- c(train = .7, validate = .15, test = .15)

g <- sample(cut(
  seq(nrow(data)),
  nrow(data)*cumsum(c(0,spec)),
  labels = names(spec)
))
```

```
Data
data            1599 obs. of 12 variables
multi.lm_mod…   List of  12
res             List of  3
residual        1119 obs. of 1 variable
val.results     240 obs. of 2 variables
Values
g               Factor w/ 3 levels "train","validate…
spec            Named num [1:3] 0.7 0.15 0.15
test_mse        0.35148323161893
test_predict…   Named num [1:240] 5.12 5.34 5.21 5.8…
val_mse         0.682382418518693
val.predicti…   Named num [1:240] 5.24 5.05 5.08 5.3…
```

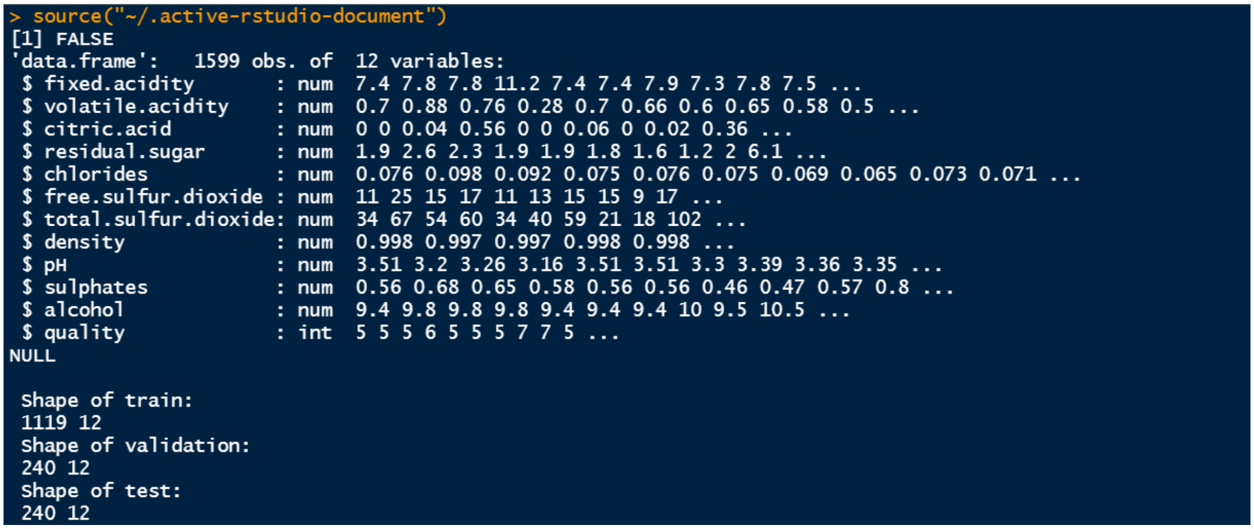


```r
res <- split(data, g)

cat('\n Shape of train: \n', dim(res$train), '\n Shape of validation: \n', dim(res$validate), '\n Shape of test: \n', dim(res$test))

# Multiple Linear Regression
#Fit the model
multi.lm_model <- lm(quality ~ ., data = res$train)
print(summary(multi.lm_model))

#Plot residual check normality
#Grab residuals
residual <- residuals(multi.lm_model)

#Convert to dataframe and plot it
residual <- as.data.frame(residual)
print('Head of the residual :')
print(head(residual))

#ggplot
ggplot(residual, aes(residual)) + geom_histogram(fill = 'blue', alpha = 0.5)

#Predict on the validation set
val.prediction <- predict(multi.lm_model, res$validate)

```

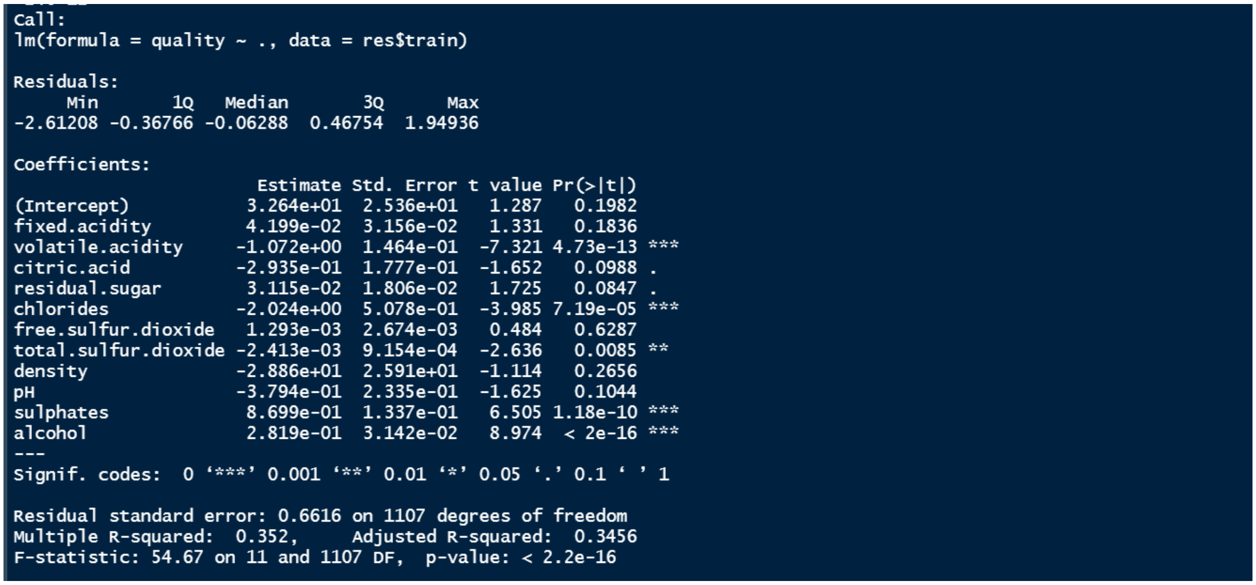


```r
#Result
val.results <- cbind(val.prediction, res$test$quality)
colnames(val.results) <- c('pred', 'valid')
val.results <- as.data.frame(val.results)

#Error
val_mse <- mean((val.results$valid - val.results$pred)^2)
cat('\nValidation MSE: ', val_mse, '\n')

#R^2 ajdusted
cat('\nR^2 adjusted ', summary(multi.lm_model)$adj.r.squared, '\n')

#Predict on the test set
cat('\n******** Prediction the test set **********')
test_predictions <- predict(multi.lm_model, res$test)

#Error
test_mse <- mean((test_predictions - res$test$quality)^2)
cat('\n Test MSE: ', test_mse, '\n')

```

Code output

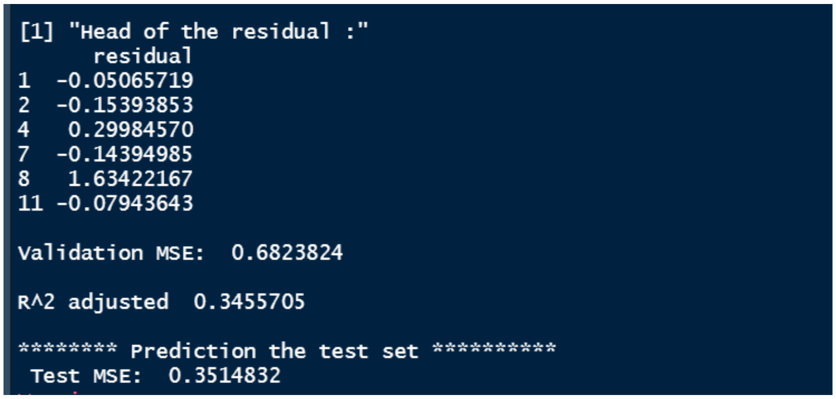


```
> source("~/.active-rstudio-document")
[1] FALSE
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
NULL

 Shape of train: 
 1119 12 
 Shape of validation: 
 240 12 
 Shape of test: 
 240 12
```



```
Call:
lm(formula = quality ~ ., data = res$train)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.61208 -0.36766 -0.06288  0.46754  1.94936 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           3.264e+01  2.536e+01   1.287   0.1982    
fixed.acidity         4.199e-02  3.156e-02   1.331   0.1836    
volatile.acidity     -1.072e+00  1.464e-01  -7.321 4.73e-13 ***
citric.acid          -2.935e-01  1.777e-01  -1.652   0.0988 .  
residual.sugar        3.115e-02  1.806e-02   1.725   0.0847 .  
chlorides            -2.024e+00  5.078e-01  -3.985 7.19e-05 ***
free.sulfur.dioxide   1.293e-03  2.674e-03   0.484   0.6287    
total.sulfur.dioxide -2.413e-03  9.154e-04  -2.636   0.0085 ** 
density              -2.886e+01  2.591e+01  -1.114   0.2656    
pH                   -3.794e-01  2.335e-01  -1.625   0.1044    
sulphates             8.699e-01  1.337e-01   6.505 1.18e-10 ***
alcohol               2.819e-01  3.142e-02   8.974  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6616 on 1107 degrees of freedom
Multiple R-squared:  0.352,	Adjusted R-squared:  0.3456 
F-statistic: 54.67 on 11 and 1107 DF,  p-value: < 2.2e-16
```

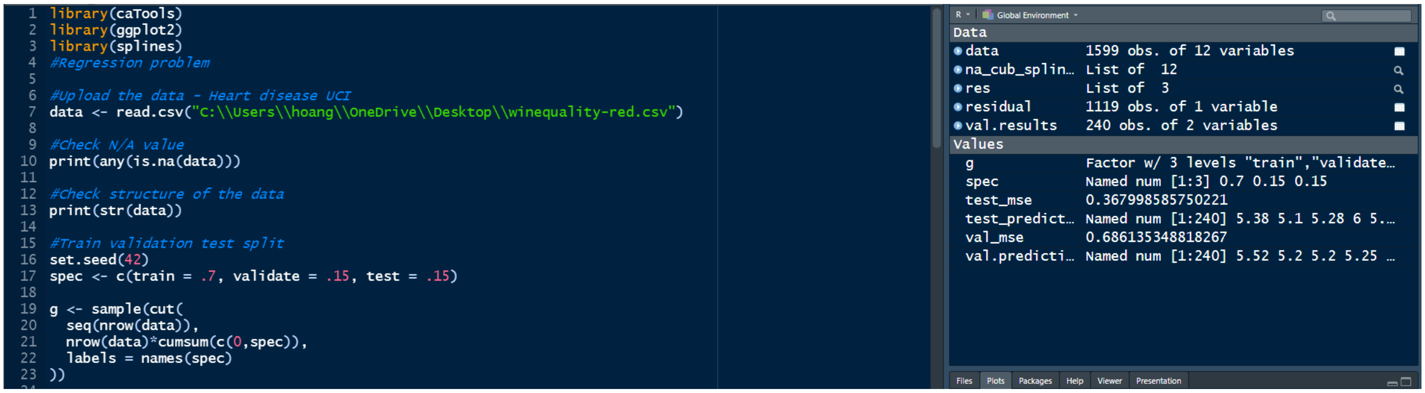
Error



```
[1] "Head of the residual :"
      residual
1  -0.05065719
2  -0.15393853
4   0.29984570
7  -0.14394985
8   1.63422167
11 -0.07943643

Validation MSE:  0.6823824

R^2 adjusted  0.3455705

******** Prediction the test set **********
 Test MSE:  0.3514832
```

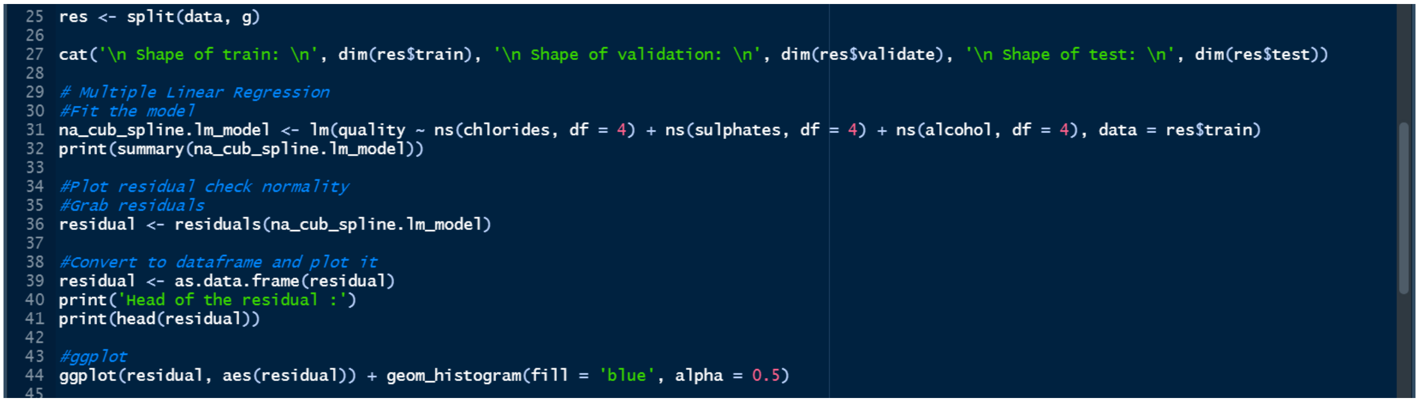
Multiple Linear Regression is performed much better than simple linear regression and polynomial regression. Under both MSE and R_squared adjusted. Again, the validation error is higher than the test error indicating the model is not overfitting.

Natural Cubic Spline

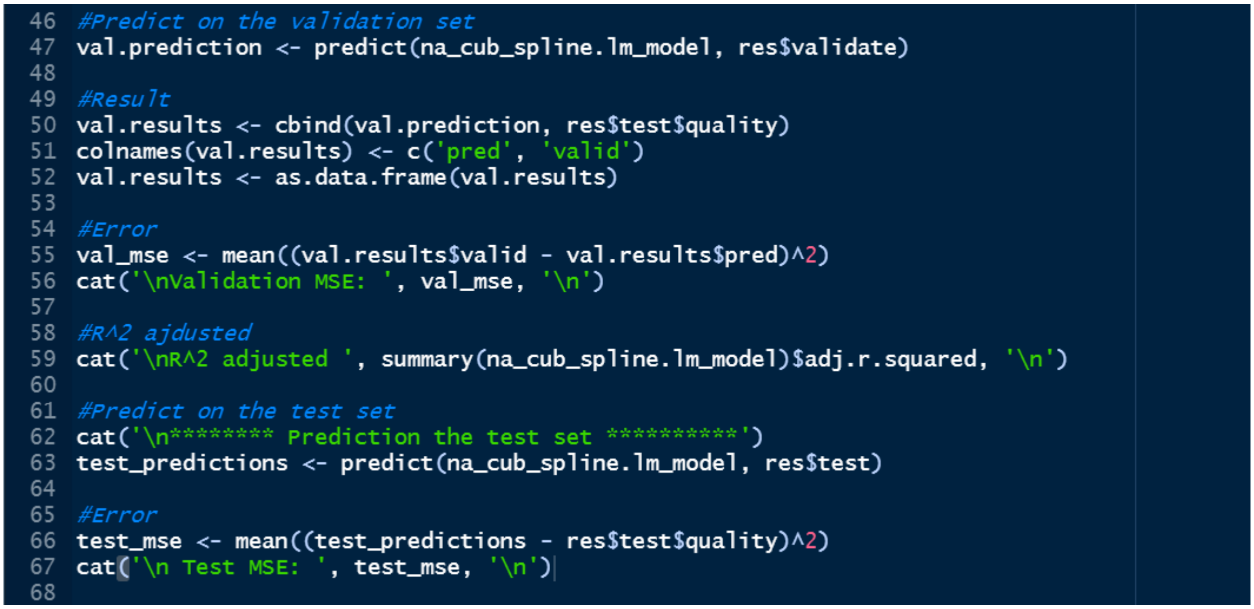


```
library(caTools)
library(ggplot2)
library(splines)
#Regression problem

#Upload the data - Heart disease UCI
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\winequality-red.csv")

#Check N/A value
print(any(is.na(data)))

#Check structure of the data
print(str(data))

#Train validation test split
set.seed(42)
spec <- c(train = .7, validate = .15, test = .15)

g <- sample(cut(
  seq(nrow(data)),
  nrow(data)*cumsum(c(0,spec)),
  labels = names(spec)
))
```

```
Data
data              1599 obs. of 12 variables
na_cub_splin…     List of  12
res               List of  3
residual          1119 obs. of 1 variable
val.results       240 obs. of 2 variables
Values
g                 Factor w/ 3 levels "train","validate…
spec              Named num [1:3] 0.7 0.15 0.15
test_mse          0.367998585750221
test_predict…     Named num [1:240] 5.38 5.1 5.28 6 5.…
val_mse           0.686135348818267
val.predicti…     Named num [1:240] 5.52 5.2 5.2 5.25 …
```

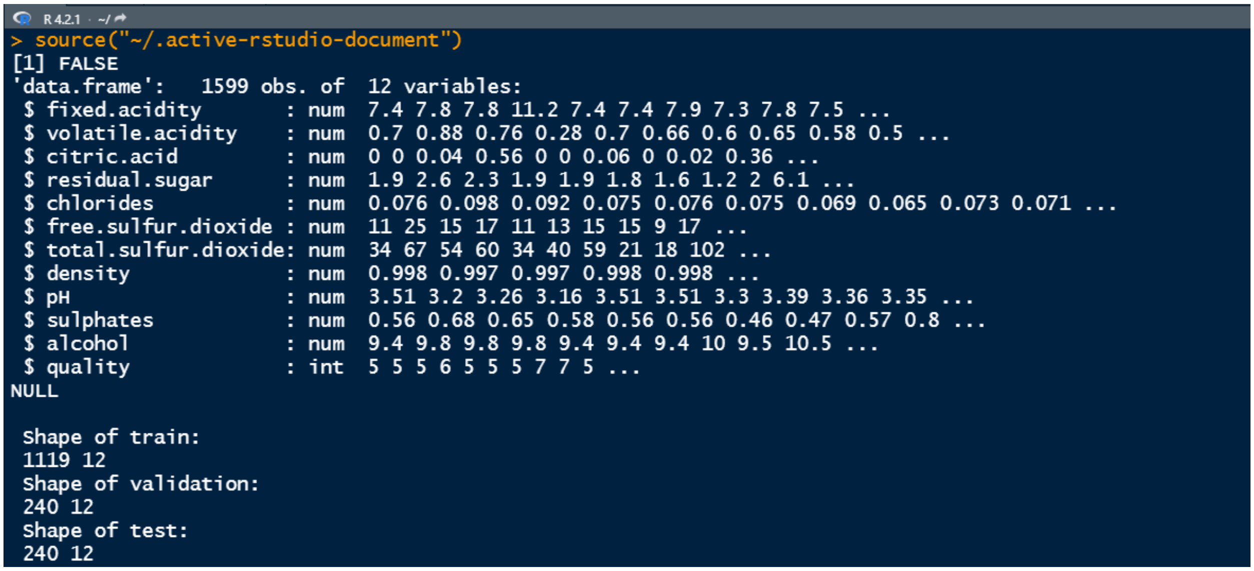


```r
res <- split(data, g)

cat('\n Shape of train: \n', dim(res$train), '\n Shape of validation: \n', dim(res$validate), '\n Shape of test: \n', dim(res$test))

# Multiple Linear Regression
#Fit the model
na_cub_spline.lm_model <- lm(quality ~ ns(chlorides, df = 4) + ns(sulphates, df = 4) + ns(alcohol, df = 4), data = res$train)
print(summary(na_cub_spline.lm_model))

#Plot residual check normality
#Grab residuals
residual <- residuals(na_cub_spline.lm_model)

#Convert to dataframe and plot it
residual <- as.data.frame(residual)
print('Head of the residual :')
print(head(residual))

#ggplot
ggplot(residual, aes(residual)) + geom_histogram(fill = 'blue', alpha = 0.5)
```



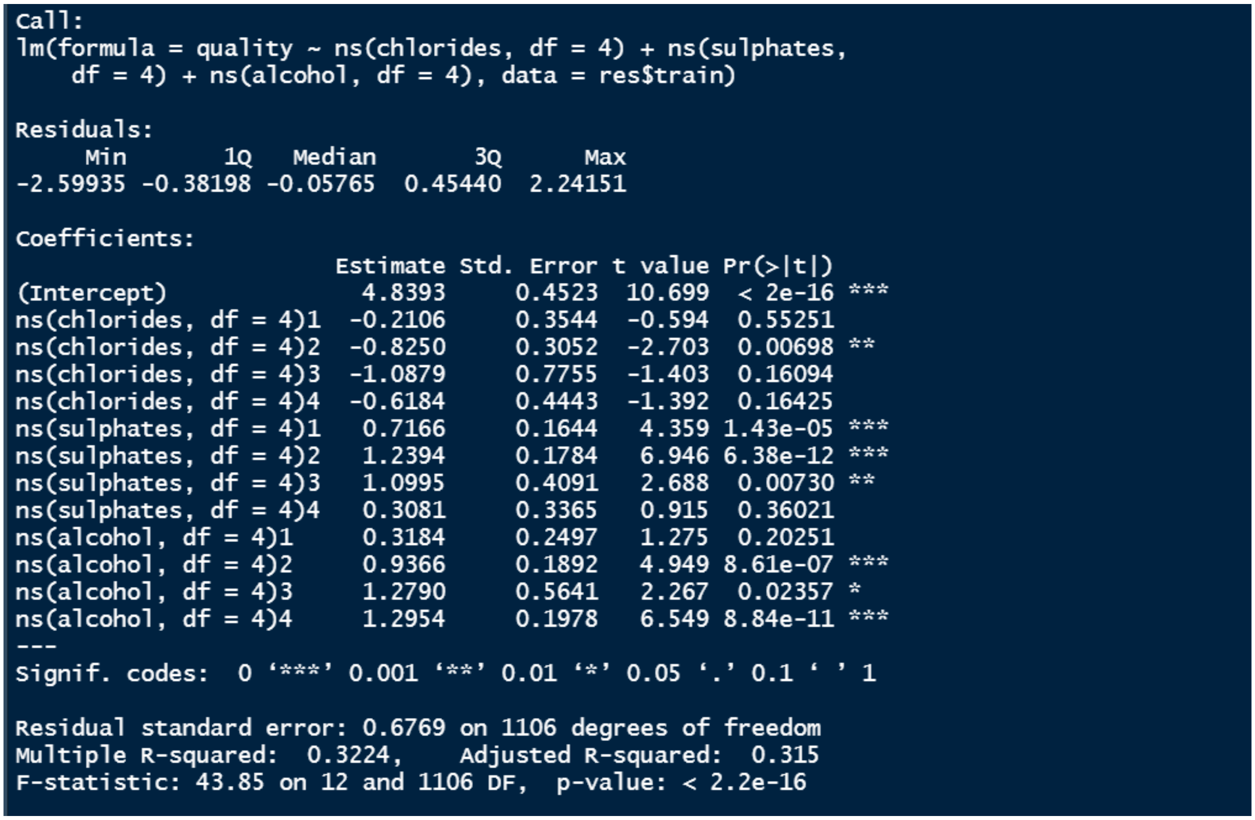
```r
#Predict on the validation set
val.prediction <- predict(na_cub_spline.lm_model, res$validate)

#Result
val.results <- cbind(val.prediction, res$test$quality)
colnames(val.results) <- c('pred', 'valid')
val.results <- as.data.frame(val.results)

#Error
val_mse <- mean((val.results$valid - val.results$pred)^2)
cat('\nValidation MSE: ', val_mse, '\n')

#R^2 ajdusted
cat('\nR^2 adjusted ', summary(na_cub_spline.lm_model)$adj.r.squared, '\n')

#Predict on the test set
cat('\n********* Prediction the test set ***********')
test_predictions <- predict(na_cub_spline.lm_model, res$test)

#Error
test_mse <- mean((test_predictions - res$test$quality)^2)
cat('\n Test MSE: ', test_mse, '\n')

```

Code output



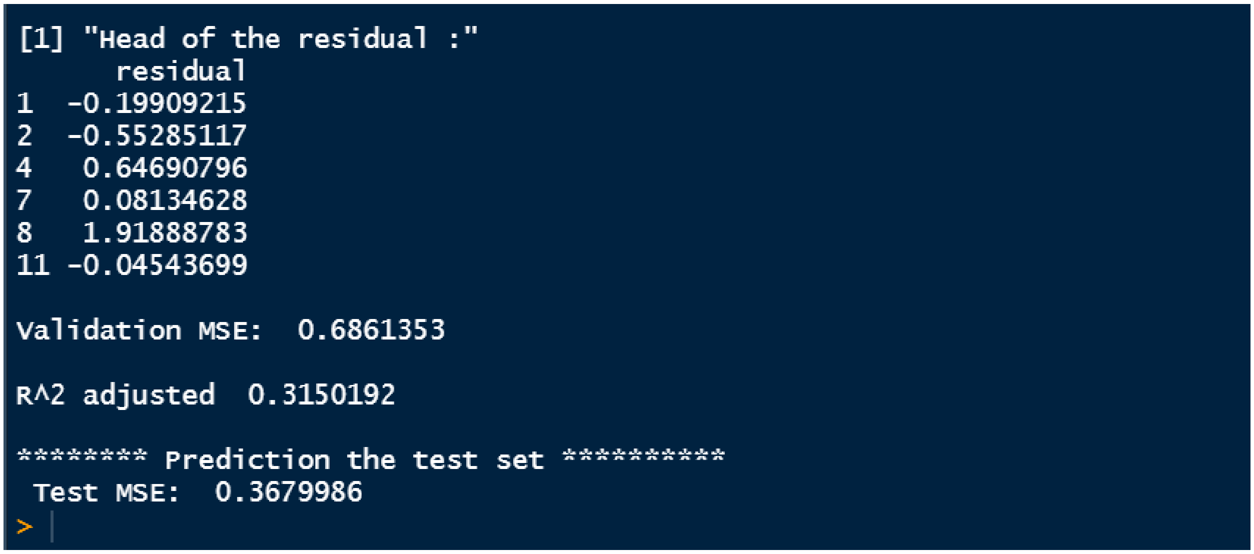
```
> source("~/.active-rstudio-document")
[1] FALSE
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
NULL

 Shape of train: 
 1119 12 
 Shape of validation: 
 240 12 
 Shape of test: 
 240 12
```

Model summary



```
Call:
lm(formula = quality ~ ns(chlorides, df = 4) + ns(sulphates,
    df = 4) + ns(alcohol, df = 4), data = res$train)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.59935 -0.38198 -0.05765  0.45440  2.24151 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)              4.8393     0.4523  10.699  < 2e-16 ***
ns(chlorides, df = 4)1  -0.2106     0.3544  -0.594  0.55251    
ns(chlorides, df = 4)2  -0.8250     0.3052  -2.703  0.00698 ** 
ns(chlorides, df = 4)3  -1.0879     0.7755  -1.403  0.16094    
ns(chlorides, df = 4)4  -0.6184     0.4443  -1.392  0.16425    
ns(sulphates, df = 4)1   0.7166     0.1644   4.359 1.43e-05 ***
ns(sulphates, df = 4)2   1.2394     0.1784   6.946 6.38e-12 ***
ns(sulphates, df = 4)3   1.0995     0.4091   2.688  0.00730 ** 
ns(sulphates, df = 4)4   0.3081     0.3365   0.915  0.36021    
ns(alcohol, df = 4)1     0.3184     0.2497   1.275  0.20251    
ns(alcohol, df = 4)2     0.9366     0.1892   4.949 8.61e-07 ***
ns(alcohol, df = 4)3     1.2790     0.5641   2.267  0.02357 *  
ns(alcohol, df = 4)4     1.2954     0.1978   6.549 8.84e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6769 on 1106 degrees of freedom
Multiple R-squared:  0.3224,	Adjusted R-squared:  0.315 
F-statistic: 43.85 on 12 and 1106 DF,  p-value: < 2.2e-16
```

Error



```
[1] "Head of the residual :"
      residual
1  -0.19909215
2  -0.55285117
4   0.64690796
7   0.08134628
8   1.91888783
11 -0.04543699

Validation MSE:  0.6861353

R^2 adjusted  0.3150192

******** Prediction the test set **********
 Test MSE:  0.3679986
> 
```

Natural Cubic Spline with 3 predictor variables (chlorides, sulphates, and alcohol) along with 4 knots (df = 4) with almost the same MSE and R_squared adjusted as the Multiple Linear

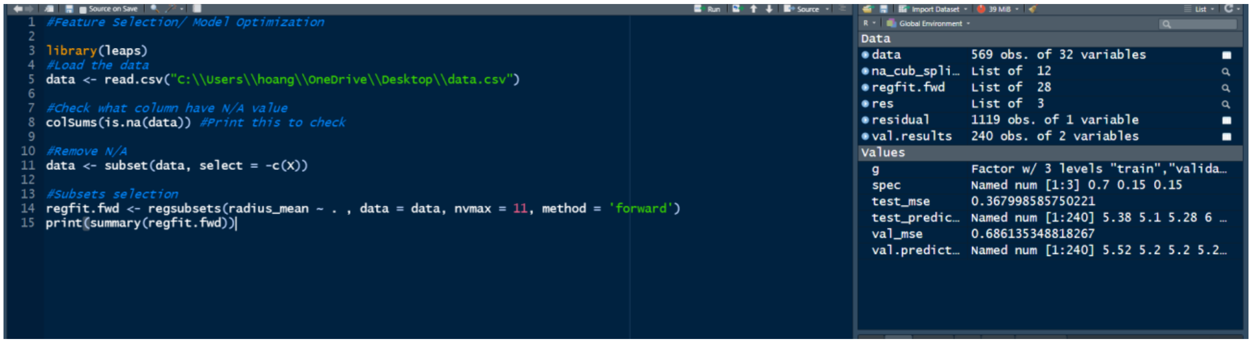
Regression. This indicates that the Natural Cubic Spline performs better than Multiple Linear Regression in terms of fewer independent variables. Hence lower computation expense compared to Multiple Linear Regression

2. Feature Selection/ Model Optimization

Breast Cancer dataset

I. Forward/Backward Stepwise Selection

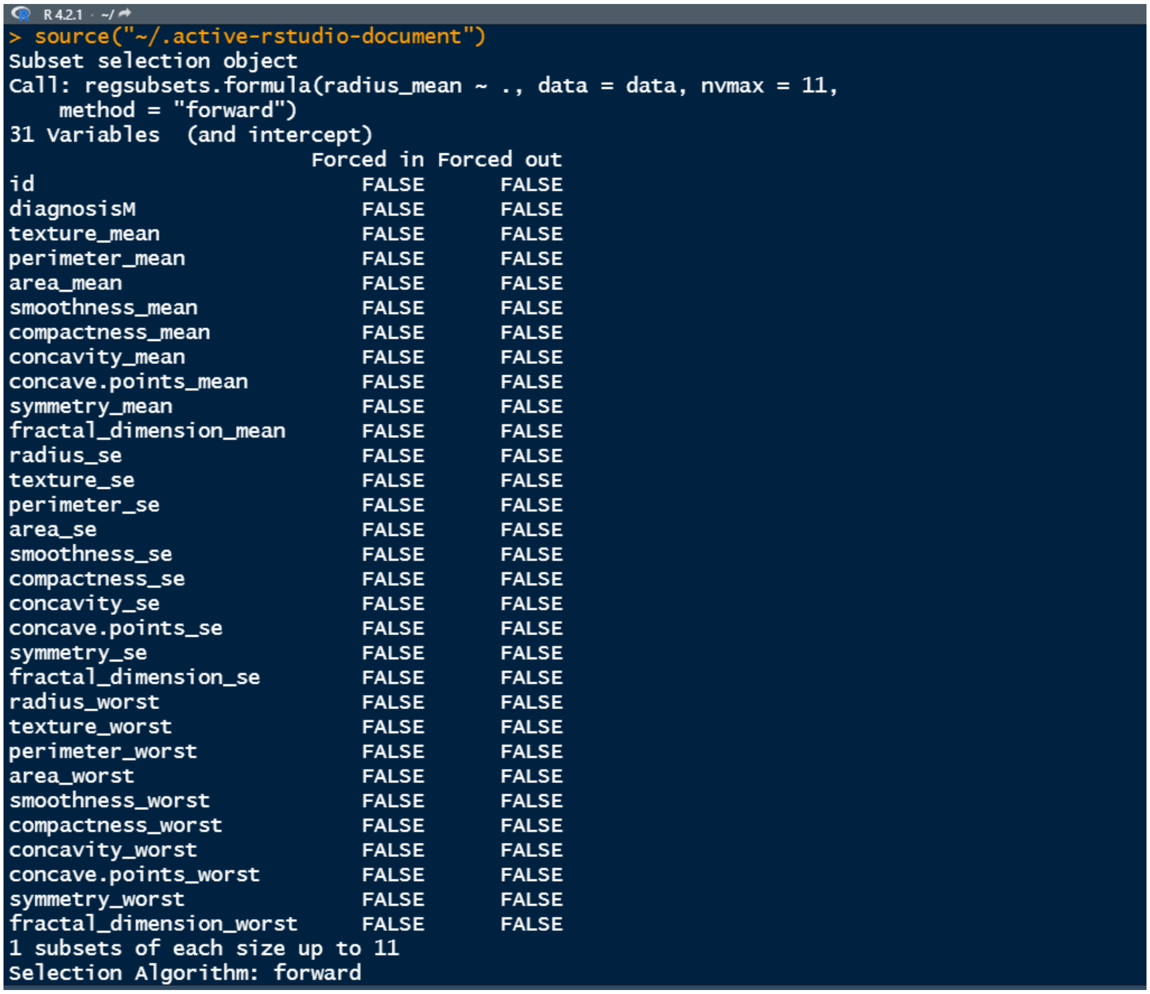
a. Forward Stepwise Selection



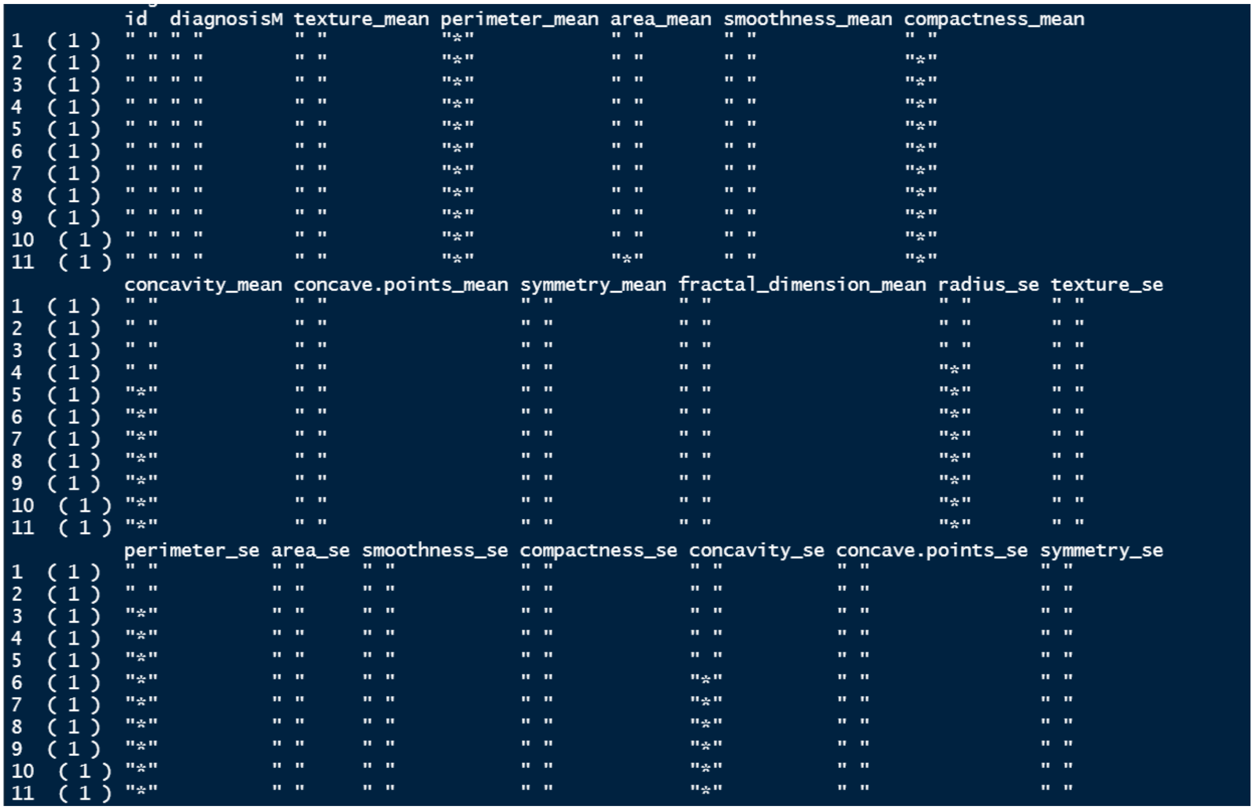
```r
#Feature Selection/ Model Optimization

library(leaps)
#Load the data
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\data.csv")

#Check what column have N/A value
colSums(is.na(data)) #Print this to check

#Remove N/A
data <- subset(data, select = -c(X))

#Subsets selection
regfit.fwd <- regsubsets(radius_mean ~ . , data = data, nvmax = 11, method = 'forward')
print(summary(regfit.fwd))
```

```
Data
data            569 obs. of 32 variables
na_cub_spli…    List of  12
regfit.fwd      List of  28
res             List of  3
residual        1119 obs. of 1 variable
val.results     240 obs. of 2 variables
Values
g               Factor w/ 3 levels "train","valida…
spec            Named num [1:3] 0.7 0.15 0.15
test_mse        0.367998585750221
test_predic…    Named num [1:240] 5.38 5.1 5.28 6 …
val_mse         0.686135348818267
val.predict…    Named num [1:240] 5.52 5.2 5.2 5.2…
```

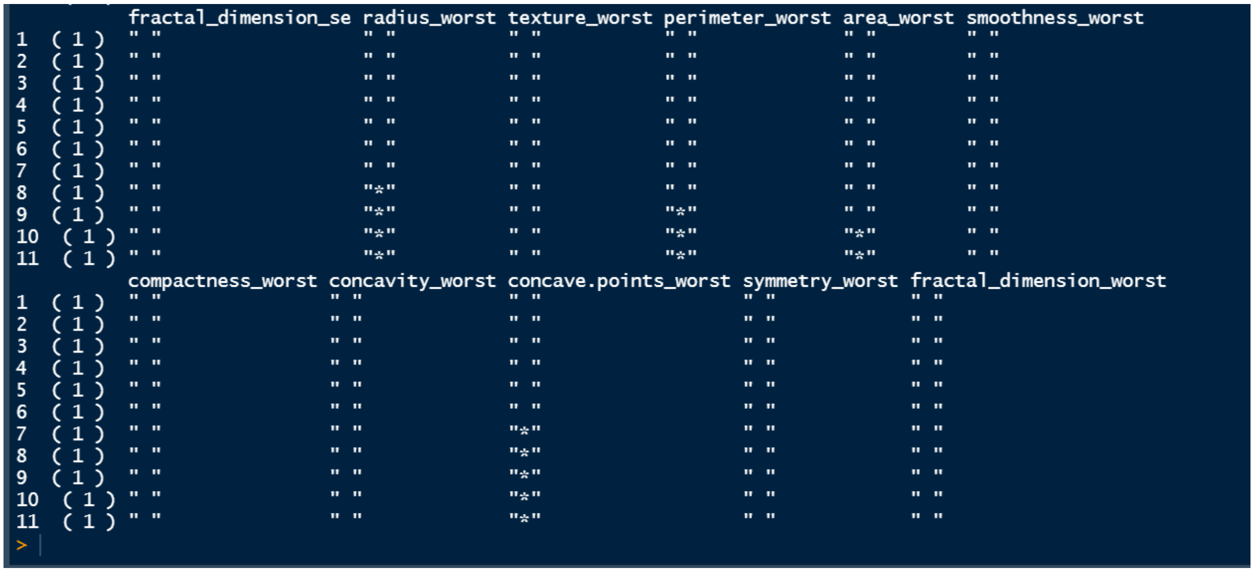
Code output



```
> source("~/.active-rstudio-document")
Subset selection object
Call: regsubsets.formula(radius_mean ~ ., data = data, nvmax = 11,
    method = "forward")
31 Variables  (and intercept)
                        Forced in Forced out
id                          FALSE      FALSE
diagnosisM                  FALSE      FALSE
texture_mean                FALSE      FALSE
perimeter_mean              FALSE      FALSE
area_mean                   FALSE      FALSE
smoothness_mean             FALSE      FALSE
compactness_mean            FALSE      FALSE
concavity_mean              FALSE      FALSE
concave.points_mean         FALSE      FALSE
symmetry_mean               FALSE      FALSE
fractal_dimension_mean      FALSE      FALSE
radius_se                   FALSE      FALSE
texture_se                  FALSE      FALSE
perimeter_se                FALSE      FALSE
area_se                     FALSE      FALSE
smoothness_se               FALSE      FALSE
compactness_se              FALSE      FALSE
concavity_se                FALSE      FALSE
concave.points_se           FALSE      FALSE
symmetry_se                 FALSE      FALSE
fractal_dimension_se        FALSE      FALSE
radius_worst                FALSE      FALSE
texture_worst               FALSE      FALSE
perimeter_worst             FALSE      FALSE
area_worst                  FALSE      FALSE
smoothness_worst            FALSE      FALSE
compactness_worst           FALSE      FALSE
concavity_worst             FALSE      FALSE
concave.points_worst        FALSE      FALSE
symmetry_worst              FALSE      FALSE
fractal_dimension_worst     FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: forward
```



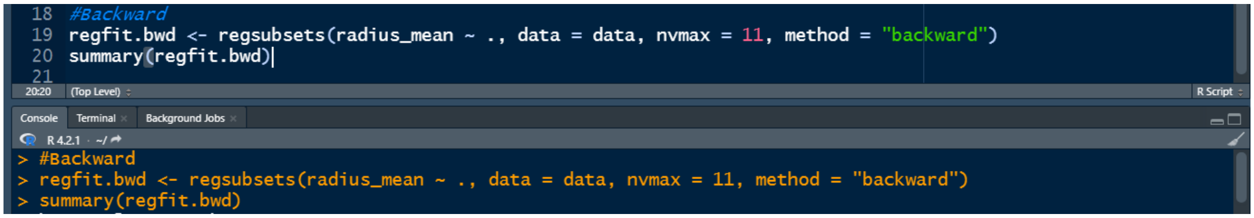
```
          id  diagnosisM texture_mean perimeter_mean area_mean smoothness_mean compactness_mean
1  ( 1 )  " " " "        " "          "*"            " "       " "             " "
2  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
3  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
4  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
5  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
6  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
7  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
8  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
9  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
10  ( 1 ) " " " "        " "          "*"            " "       " "             "*"
11  ( 1 ) " " " "        " "          "*"            "*"       " "             "*"
          concavity_mean concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se
1  ( 1 )  " "            " "                 " "           " "                    " "       " "
2  ( 1 )  " "            " "                 " "           " "                    " "       " "
3  ( 1 )  " "            " "                 " "           " "                    " "       " "
4  ( 1 )  " "            " "                 " "           " "                    "*"       " "
5  ( 1 )  "*"            " "                 " "           " "                    "*"       " "
6  ( 1 )  "*"            " "                 " "           " "                    "*"       " "
7  ( 1 )  "*"            " "                 " "           " "                    "*"       " "
8  ( 1 )  "*"            " "                 " "           " "                    "*"       " "
9  ( 1 )  "*"            " "                 " "           " "                    "*"       " "
10  ( 1 ) "*"            " "                 " "           " "                    "*"       " "
11  ( 1 ) "*"            " "                 " "           " "                    "*"       " "
          perimeter_se area_se smoothness_se compactness_se concavity_se concave.points_se symmetry_se
1  ( 1 )  " "          " "     " "           " "            " "          " "               " "
2  ( 1 )  " "          " "     " "           " "            " "          " "               " "
3  ( 1 )  "*"          " "     " "           " "            " "          " "               " "
4  ( 1 )  "*"          " "     " "           " "            " "          " "               " "
5  ( 1 )  "*"          " "     " "           " "            " "          " "               " "
6  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
7  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
8  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
9  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
10  ( 1 ) "*"          " "     " "           " "            "*"          " "               " "
11  ( 1 ) "*"          " "     " "           " "            "*"          " "               " "
```



```
          fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst smoothness_worst
1  ( 1 )  " "                  " "          " "           " "             " "        " "
2  ( 1 )  " "                  " "          " "           " "             " "        " "
3  ( 1 )  " "                  " "          " "           " "             " "        " "
4  ( 1 )  " "                  " "          " "           " "             " "        " "
5  ( 1 )  " "                  " "          " "           " "             " "        " "
6  ( 1 )  " "                  " "          " "           " "             " "        " "
7  ( 1 )  " "                  " "          " "           " "             " "        " "
8  ( 1 )  " "                  "*"          " "           " "             " "        " "
9  ( 1 )  " "                  "*"          " "           "*"             " "        " "
10  ( 1 ) " "                  "*"          " "           "*"             "*"        " "
11  ( 1 ) " "                  "*"          " "           "*"             "*"        " "
          compactness_worst concavity_worst concave.points_worst symmetry_worst fractal_dimension_worst
1  ( 1 )  " "               " "             " "                  " "            " "
2  ( 1 )  " "               " "             " "                  " "            " "
3  ( 1 )  " "               " "             " "                  " "            " "
4  ( 1 )  " "               " "             " "                  " "            " "
5  ( 1 )  " "               " "             " "                  " "            " "
6  ( 1 )  " "               " "             " "                  " "            " "
7  ( 1 )  " "               " "             "*"                  " "            " "
8  ( 1 )  " "               " "             "*"                  " "            " "
9  ( 1 )  " "               " "             "*"                  " "            " "
10  ( 1 ) " "               " "             "*"                  " "            " "
11  ( 1 ) " "               " "             "*"                  " "            " "
> 
```

b. Backward Stepwise Selection

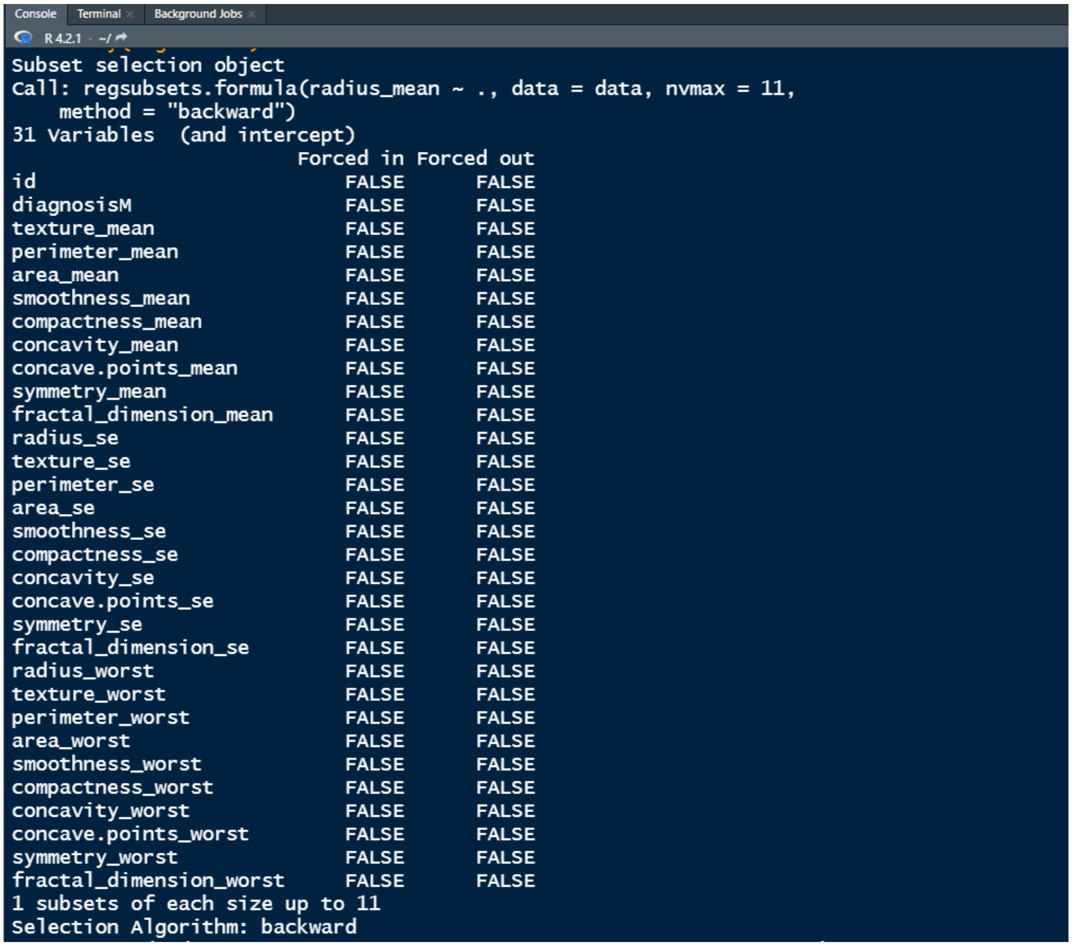
Code



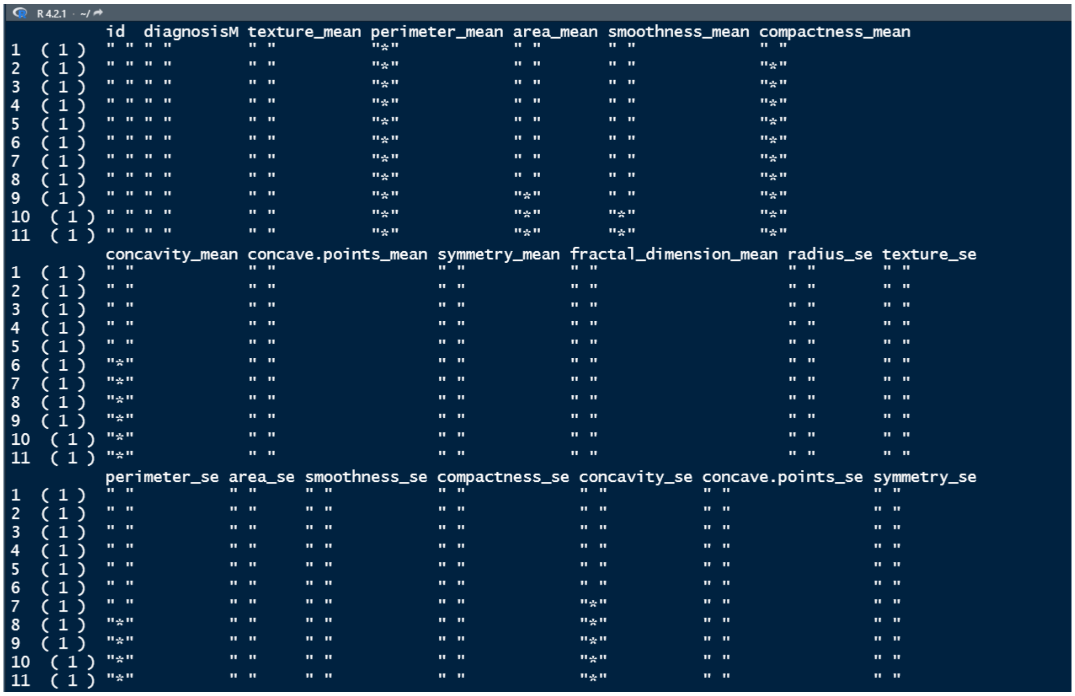
```
#Backward
regfit.bwd <- regsubsets(radius_mean ~ ., data = data, nvmax = 11, method = "backward")
summary(regfit.bwd)
```

```
> #Backward
> regfit.bwd <- regsubsets(radius_mean ~ ., data = data, nvmax = 11, method = "backward")
> summary(regfit.bwd)
```

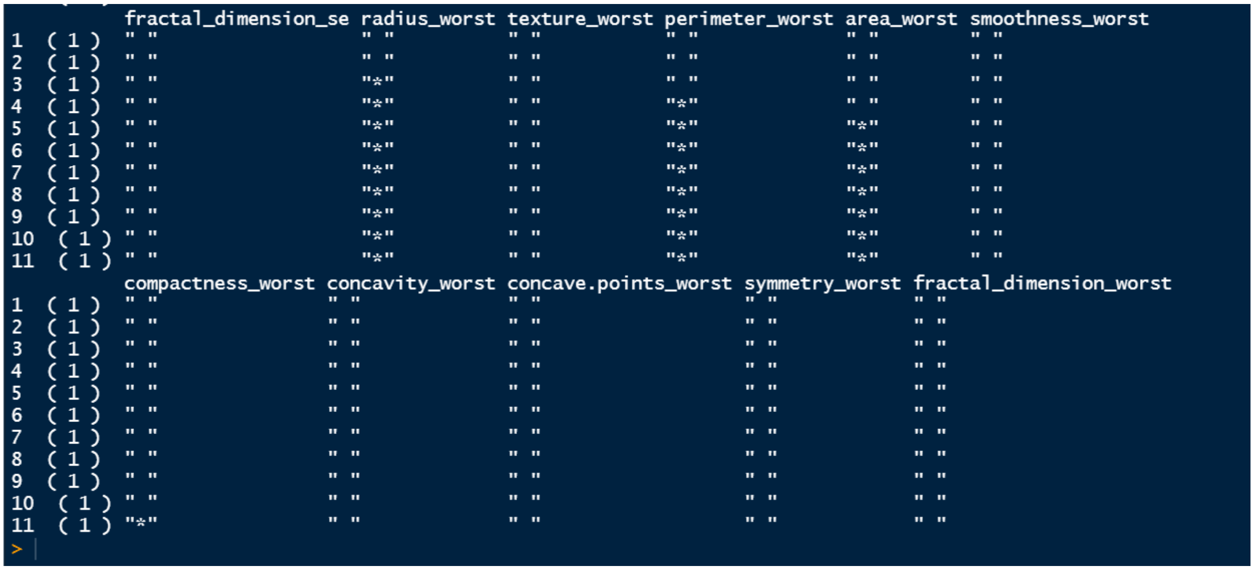
Code output



```
Subset selection object
Call: regsubsets.formula(radius_mean ~ ., data = data, nvmax = 11,
    method = "backward")
31 Variables  (and intercept)
                        Forced in Forced out
id                          FALSE      FALSE
diagnosisM                  FALSE      FALSE
texture_mean                FALSE      FALSE
perimeter_mean              FALSE      FALSE
area_mean                   FALSE      FALSE
smoothness_mean             FALSE      FALSE
compactness_mean            FALSE      FALSE
concavity_mean              FALSE      FALSE
concave.points_mean         FALSE      FALSE
symmetry_mean               FALSE      FALSE
fractal_dimension_mean      FALSE      FALSE
radius_se                   FALSE      FALSE
texture_se                  FALSE      FALSE
perimeter_se                FALSE      FALSE
area_se                     FALSE      FALSE
smoothness_se               FALSE      FALSE
compactness_se              FALSE      FALSE
concavity_se                FALSE      FALSE
concave.points_se           FALSE      FALSE
symmetry_se                 FALSE      FALSE
fractal_dimension_se        FALSE      FALSE
radius_worst                FALSE      FALSE
texture_worst               FALSE      FALSE
perimeter_worst             FALSE      FALSE
area_worst                  FALSE      FALSE
smoothness_worst            FALSE      FALSE
compactness_worst           FALSE      FALSE
concavity_worst             FALSE      FALSE
concave.points_worst        FALSE      FALSE
symmetry_worst              FALSE      FALSE
fractal_dimension_worst     FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: backward
```



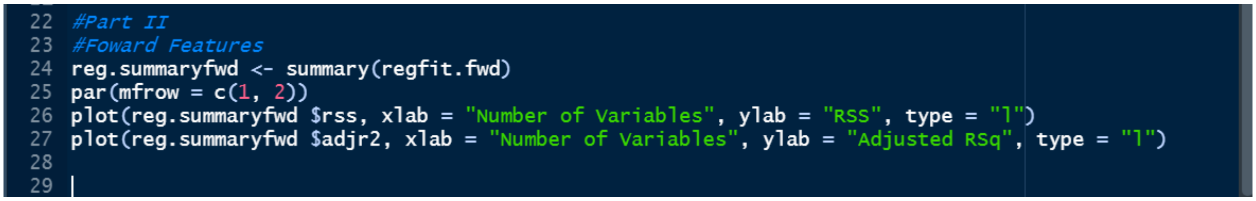
```
          id  diagnosisM texture_mean perimeter_mean area_mean smoothness_mean compactness_mean
1  ( 1 )  " " " "        " "          "*"            " "       " "             " "
2  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
3  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
4  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
5  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
6  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
7  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
8  ( 1 )  " " " "        " "          "*"            " "       " "             "*"
9  ( 1 )  " " " "        " "          "*"            "*"       " "             "*"
10  ( 1 ) " " " "        " "          "*"            "*"       "*"             "*"
11  ( 1 ) " " " "        " "          "*"            "*"       "*"             "*"
          concavity_mean concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se
1  ( 1 )  " "            " "                 " "           " "                    " "       " "
2  ( 1 )  " "            " "                 " "           " "                    " "       " "
3  ( 1 )  " "            " "                 " "           " "                    " "       " "
4  ( 1 )  " "            " "                 " "           " "                    " "       " "
5  ( 1 )  " "            " "                 " "           " "                    " "       " "
6  ( 1 )  "*"            " "                 " "           " "                    " "       " "
7  ( 1 )  "*"            " "                 " "           " "                    " "       " "
8  ( 1 )  "*"            " "                 " "           " "                    " "       " "
9  ( 1 )  "*"            " "                 " "           " "                    " "       " "
10  ( 1 ) "*"            " "                 " "           " "                    " "       " "
11  ( 1 ) "*"            " "                 " "           " "                    " "       " "
          perimeter_se area_se smoothness_se compactness_se concavity_se concave.points_se symmetry_se
1  ( 1 )  " "          " "     " "           " "            " "          " "               " "
2  ( 1 )  " "          " "     " "           " "            " "          " "               " "
3  ( 1 )  " "          " "     " "           " "            " "          " "               " "
4  ( 1 )  " "          " "     " "           " "            " "          " "               " "
5  ( 1 )  " "          " "     " "           " "            " "          " "               " "
6  ( 1 )  " "          " "     " "           " "            " "          " "               " "
7  ( 1 )  " "          " "     " "           " "            "*"          " "               " "
8  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
9  ( 1 )  "*"          " "     " "           " "            "*"          " "               " "
10  ( 1 ) "*"          " "     " "           " "            "*"          " "               " "
11  ( 1 ) "*"          " "     " "           " "            "*"          " "               " "
```



```
          fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst smoothness_worst
1  ( 1 )  " "                  " "          " "           " "             " "        " "
2  ( 1 )  " "                  " "          " "           " "             " "        " "
3  ( 1 )  " "                  "*"          " "           " "             " "        " "
4  ( 1 )  " "                  "*"          " "           "*"             " "        " "
5  ( 1 )  " "                  "*"          " "           "*"             "*"        " "
6  ( 1 )  " "                  "*"          " "           "*"             "*"        " "
7  ( 1 )  " "                  "*"          " "           "*"             "*"        " "
8  ( 1 )  " "                  "*"          " "           "*"             "*"        " "
9  ( 1 )  " "                  "*"          " "           "*"             "*"        " "
10  ( 1 ) " "                  "*"          " "           "*"             "*"        " "
11  ( 1 ) " "                  "*"          " "           "*"             "*"        " "
          compactness_worst concavity_worst concave.points_worst symmetry_worst fractal_dimension_worst
1  ( 1 )  " "               " "             " "                  " "            " "
2  ( 1 )  " "               " "             " "                  " "            " "
3  ( 1 )  " "               " "             " "                  " "            " "
4  ( 1 )  " "               " "             " "                  " "            " "
5  ( 1 )  " "               " "             " "                  " "            " "
6  ( 1 )  " "               " "             " "                  " "            " "
7  ( 1 )  " "               " "             " "                  " "            " "
8  ( 1 )  " "               " "             " "                  " "            " "
9  ( 1 )  " "               " "             " "                  " "            " "
10  ( 1 ) " "               " "             " "                  " "            " "
11  ( 1 ) "*"               " "             " "                  " "            " "
>
```

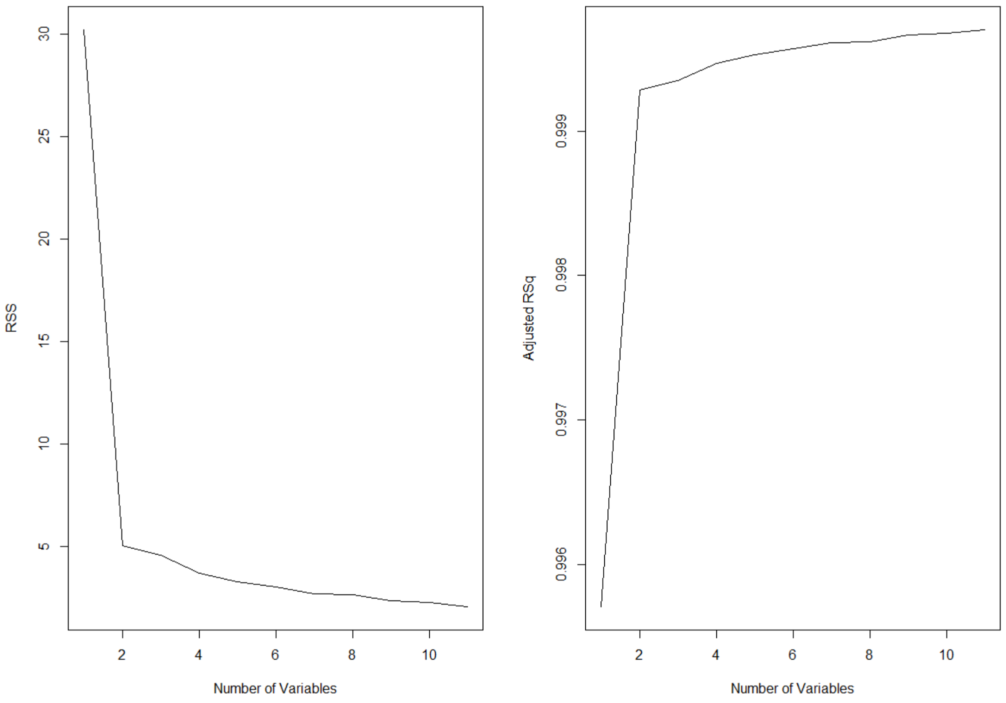
II. Forward/Backward Features

a. Forward Features

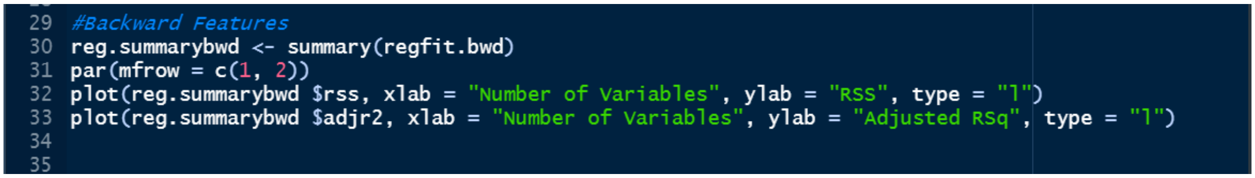


```
#Part II
#Foward Features
reg.summaryfwd <- summary(regfit.fwd)
par(mfrow = c(1, 2))
plot(reg.summaryfwd $rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg.summaryfwd $adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")
```

Code output

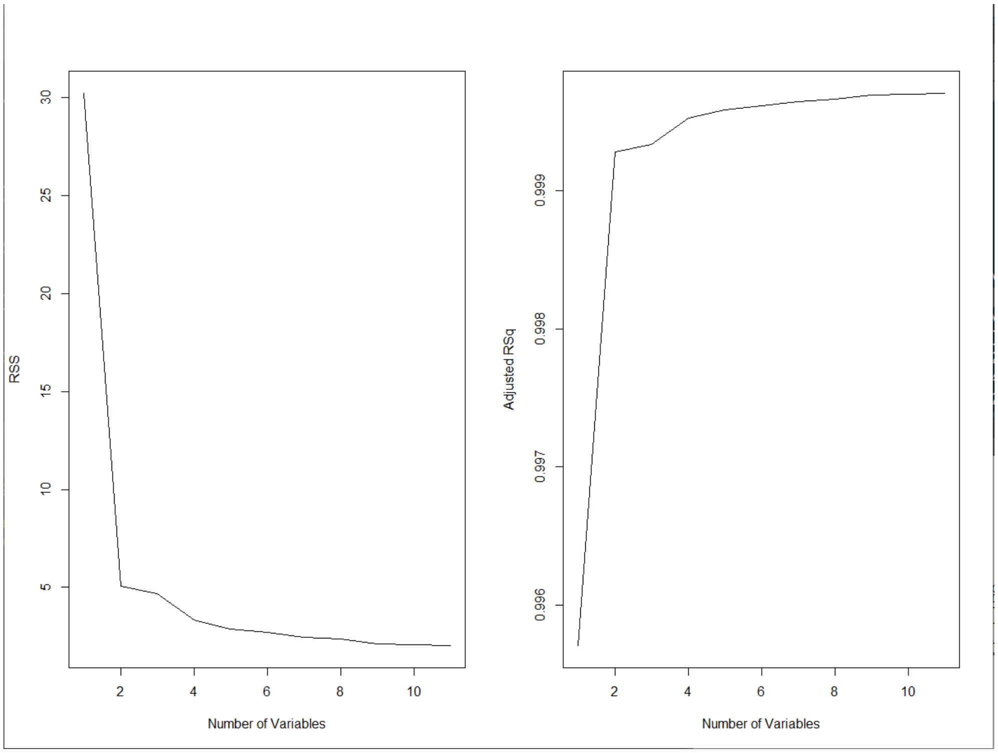


b. Backward Features



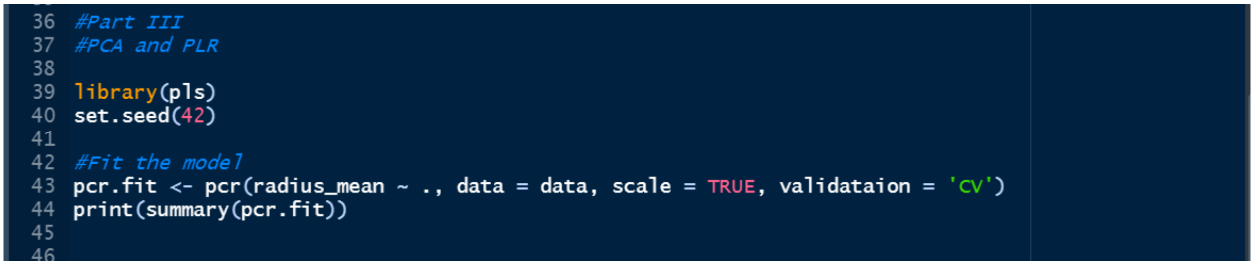
```
#Backward Features
reg.summarybwd <- summary(regfit.bwd)
par(mfrow = c(1, 2))
plot(reg.summarybwd $rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg.summarybwd $adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")
```

Code output



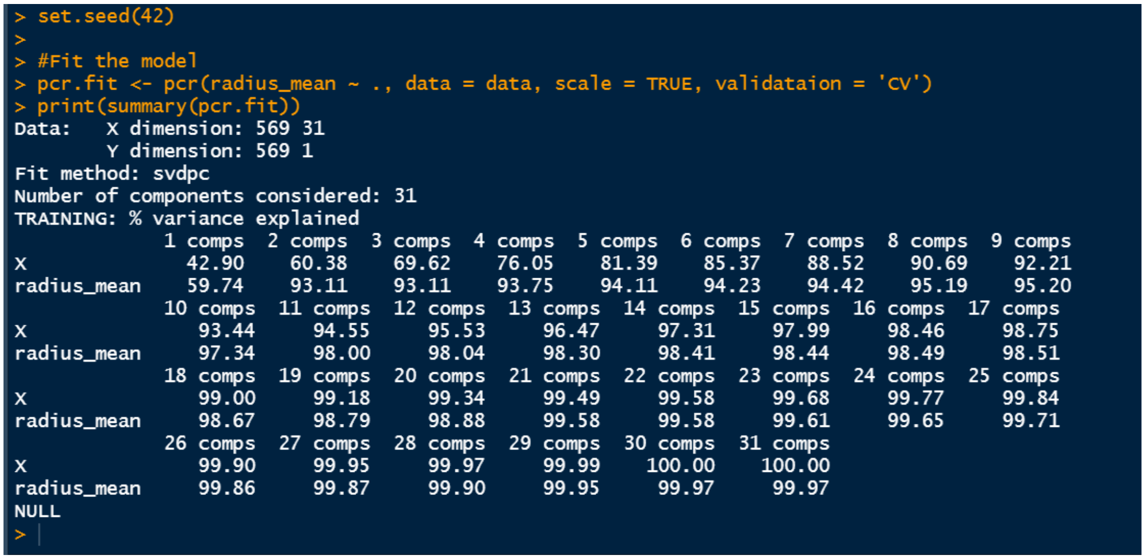
III. PCA

Code



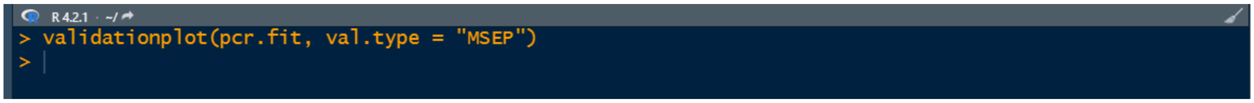
```
#Part III
#PCA and PLR

library(pls)
set.seed(42)

#Fit the model
pcr.fit <- pcr(radius_mean ~ ., data = data, scale = TRUE, validataion = 'CV')
print(summary(pcr.fit))
```

Code output



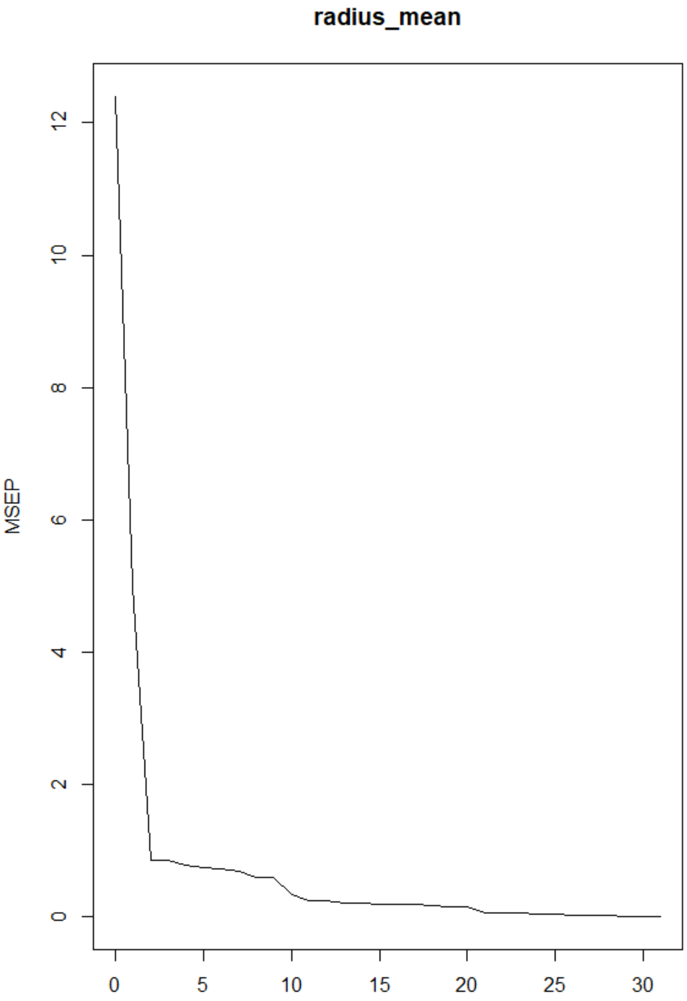
```
> set.seed(42)
>
> #Fit the model
> pcr.fit <- pcr(radius_mean ~ ., data = data, scale = TRUE, validataion = 'CV')
> print(summary(pcr.fit))
Data:   X dimension: 569 31
        Y dimension: 569 1
Fit method: svdpc
Number of components considered: 31
TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
X              42.90    60.38    69.62    76.05    81.39    85.37    88.52    90.69    92.21
radius_mean    59.74    93.11    93.11    93.75    94.11    94.23    94.42    95.19    95.20
             10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
X               93.44     94.55     95.53     96.47     97.31     97.99     98.46     98.75
radius_mean     97.34     98.00     98.04     98.30     98.41     98.44     98.49     98.51
             18 comps  19 comps  20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
X               99.00     99.18     99.34     99.49     99.58     99.68     99.77     99.84
radius_mean     98.67     98.79     98.88     99.58     99.58     99.61     99.65     99.71
             26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
X               99.90     99.95     99.97     99.99    100.00    100.00
radius_mean     99.86     99.87     99.90     99.95     99.97     99.97
NULL
> |
```

Code

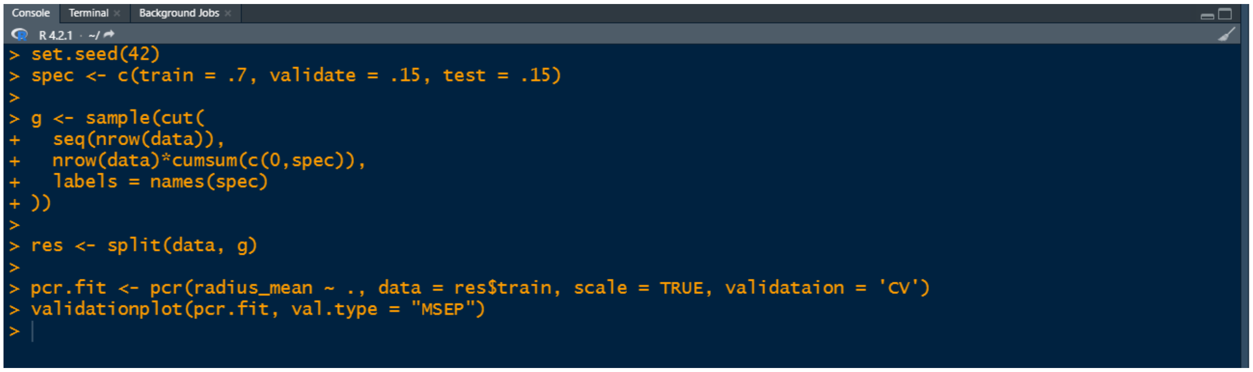


```
> validationplot(pcr.fit, val.type = "MSEP")
> |
```

Code output

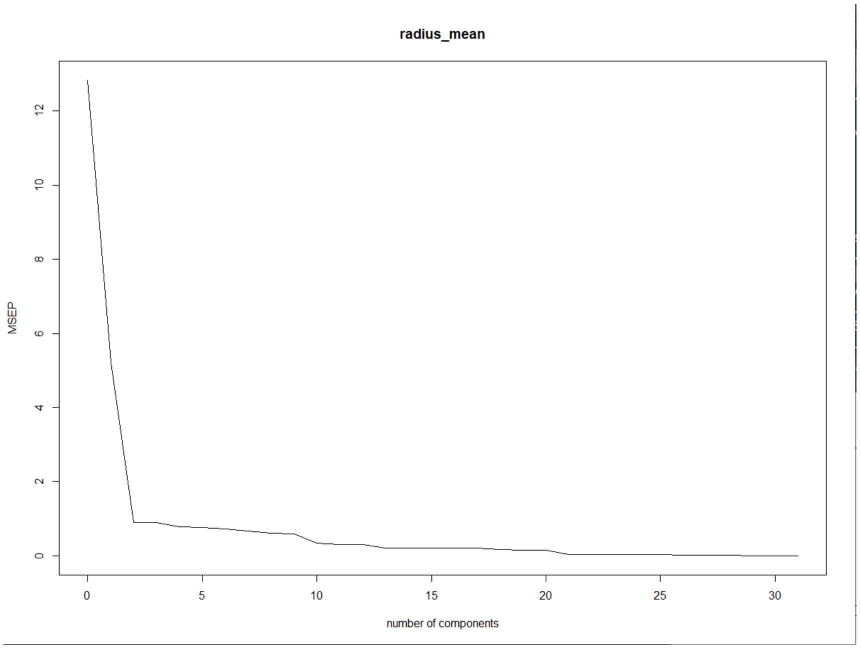


Code

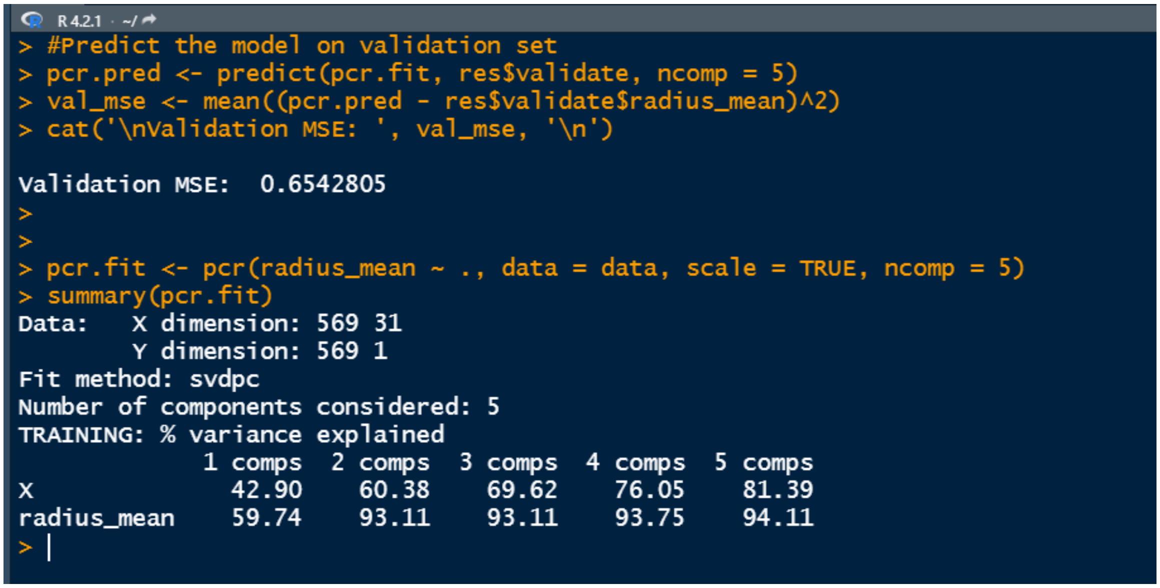


```
> set.seed(42)
> spec <- c(train = .7, validate = .15, test = .15)
>
> g <- sample(cut(
+   seq(nrow(data)),
+   nrow(data)*cumsum(c(0,spec)),
+   labels = names(spec)
+ ))
>
> res <- split(data, g)
>
> pcr.fit <- pcr(radius_mean ~ ., data = res$train, scale = TRUE, validataion = 'CV')
> validationplot(pcr.fit, val.type = "MSEP")
>
```

Code output



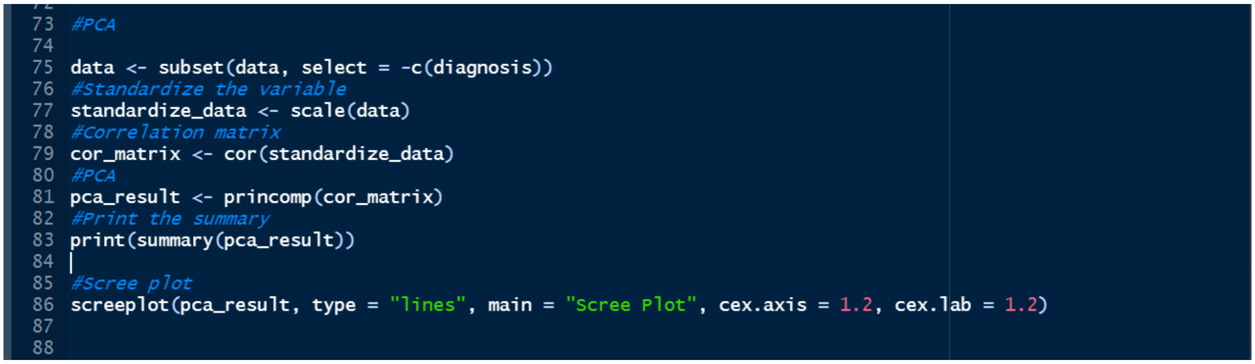
Code output



```
> #Predict the model on validation set
> pcr.pred <- predict(pcr.fit, res$validate, ncomp = 5)
> val_mse <- mean((pcr.pred - res$validate$radius_mean)^2)
> cat('\nValidation MSE: ', val_mse, '\n')

Validation MSE:  0.6542805
>
>
> pcr.fit <- pcr(radius_mean ~ ., data = data, scale = TRUE, ncomp = 5)
> summary(pcr.fit)
Data:   X dimension: 569 31
        Y dimension: 569 1
Fit method: svdpc
Number of components considered: 5
TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps
X              42.90    60.38    69.62    76.05    81.39
radius_mean    59.74    93.11    93.11    93.75    94.11
>
```

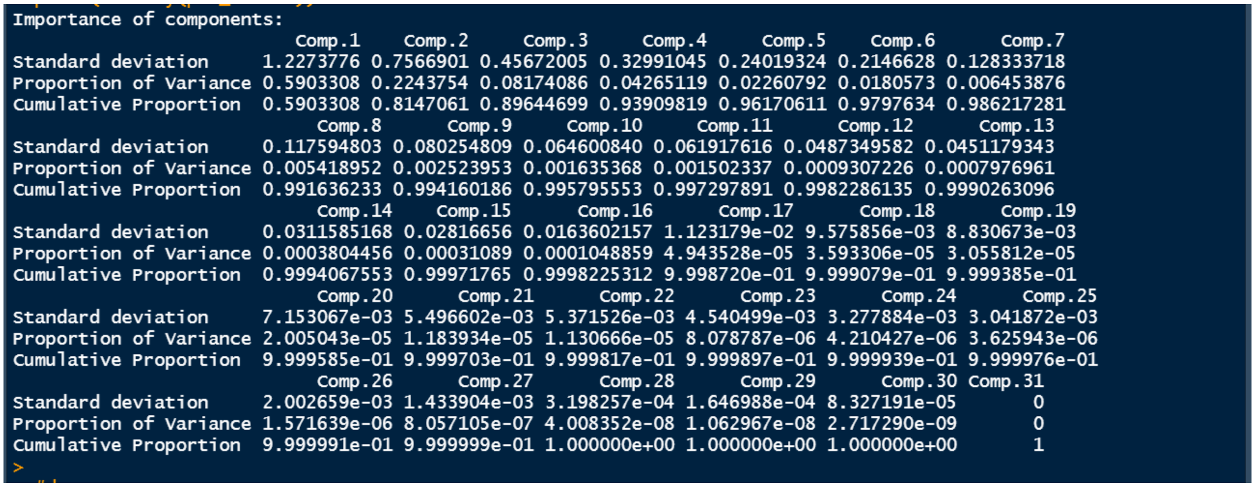
Principle Component Analysis

Code



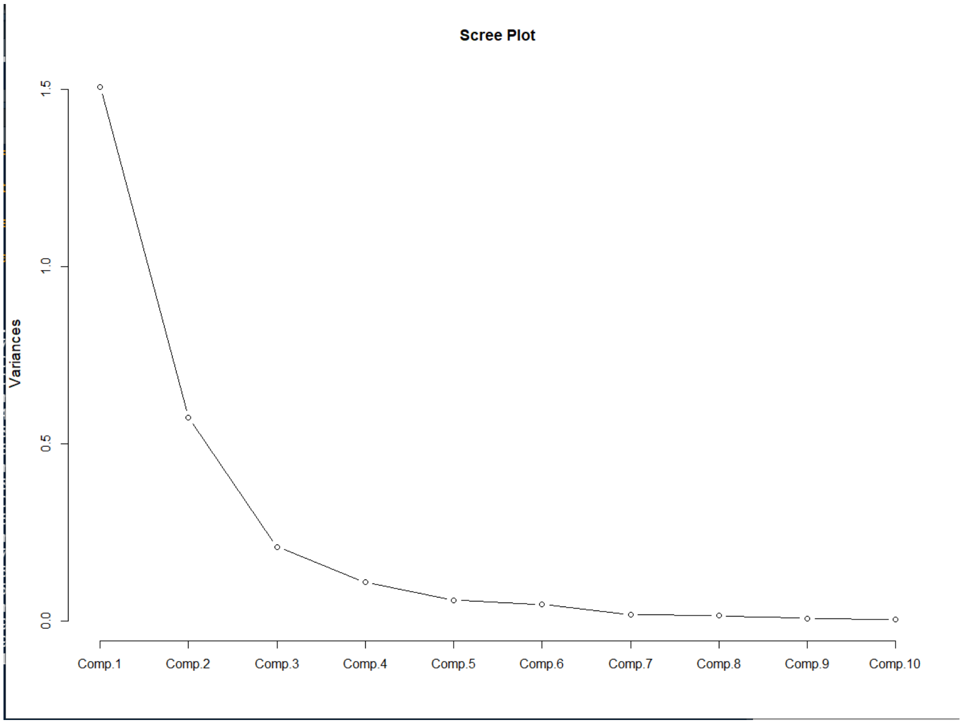
```r
#PCA

data <- subset(data, select = -c(diagnosis))
#Standardize the variable
standardize_data <- scale(data)
#Correlation matrix
cor_matrix <- cor(standardize_data)
#PCA
pca_result <- princomp(cor_matrix)
#Print the summary
print(summary(pca_result))

#Scree plot
screeplot(pca_result, type = "lines", main = "Scree Plot", cex.axis = 1.2, cex.lab = 1.2)
```

Code output

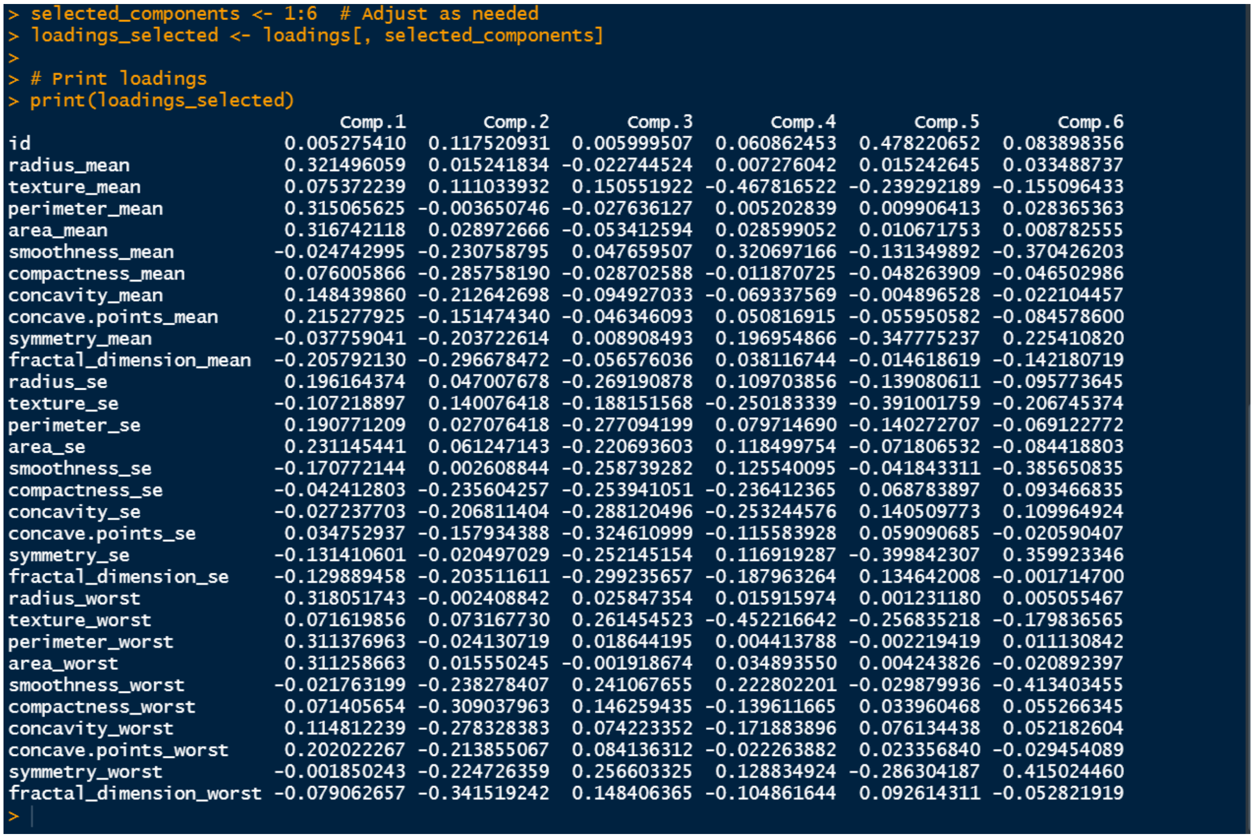


```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4     Comp.5    Comp.6      Comp.7
Standard deviation     1.2273776 0.7566901 0.45672005 0.32991045 0.24019324 0.2146628 0.128333718
Proportion of Variance 0.5903308 0.2243754 0.08174086 0.04265119 0.02260792 0.0180573 0.006453876
Cumulative Proportion  0.5903308 0.8147061 0.89644699 0.93909819 0.96170611 0.9797634 0.986217281
                            Comp.8      Comp.9     Comp.10     Comp.11      Comp.12      Comp.13
Standard deviation     0.117594803 0.080254809 0.064600840 0.061917616 0.0487349582 0.0451179343
Proportion of Variance 0.005418952 0.002523953 0.001635368 0.001502337 0.0009307226 0.0007976961
Cumulative Proportion  0.991636233 0.994160186 0.995795553 0.997297891 0.9982286135 0.9990263096
                            Comp.14    Comp.15      Comp.16      Comp.17      Comp.18      Comp.19
Standard deviation     0.0311585168 0.02816656 0.0163602157 1.123179e-02 9.575856e-03 8.830673e-03
Proportion of Variance 0.0003804456 0.00031089 0.0001048859 4.943528e-05 3.593306e-05 3.055812e-05
Cumulative Proportion  0.9994067553 0.99971765 0.9998225312 9.998720e-01 9.999079e-01 9.999385e-01
                            Comp.20      Comp.21      Comp.22      Comp.23      Comp.24      Comp.25
Standard deviation     7.153067e-03 5.496602e-03 5.371526e-03 4.540499e-03 3.277884e-03 3.041872e-03
Proportion of Variance 2.005043e-05 1.183934e-05 1.130666e-05 8.078787e-06 4.210427e-06 3.625943e-06
Cumulative Proportion  9.999585e-01 9.999703e-01 9.999817e-01 9.999897e-01 9.999939e-01 9.999976e-01
                            Comp.26      Comp.27      Comp.28      Comp.29      Comp.30 Comp.31
Standard deviation     2.002659e-03 1.433904e-03 3.198257e-04 1.646988e-04 8.327191e-05       0
Proportion of Variance 1.571639e-06 8.057105e-07 4.008352e-08 1.062967e-08 2.717290e-09       0
Cumulative Proportion  9.999991e-01 9.999999e-01 1.000000e+00 1.000000e+00 1.000000e+00       1
>
```

Scree plot



Since 6 components capture 97% of the total variance. Hence select 6 PCs



```
> selected_components <- 1:6  # Adjust as needed
> loadings_selected <- loadings[, selected_components]
>
> # Print loadings
> print(loadings_selected)
                              Comp.1       Comp.2       Comp.3       Comp.4       Comp.5       Comp.6
id                       0.005275410  0.117520931  0.005999507  0.060862453  0.478220652  0.083898356
radius_mean              0.321496059  0.015241834 -0.022744524  0.007276042  0.015242645  0.033488737
texture_mean             0.075372239  0.111033932  0.150551922 -0.467816522 -0.239292189 -0.155096433
perimeter_mean           0.315065625 -0.003650746 -0.027636127  0.005202839  0.009906413  0.028365363
area_mean                0.316742118  0.028972666 -0.053412594  0.028599052  0.010671753  0.008782555
smoothness_mean         -0.024742995 -0.230758795  0.047659507  0.320697166 -0.131349892 -0.370426203
compactness_mean         0.076005866 -0.285758190 -0.028702588 -0.011870725 -0.048263909 -0.046502986
concavity_mean           0.148439860 -0.212642698 -0.094927033 -0.069337569 -0.004896528 -0.022104457
concave.points_mean      0.215277925 -0.151474340 -0.046346093  0.050816915 -0.055950582 -0.084578600
symmetry_mean           -0.037759041 -0.203722614  0.008908493  0.196954866 -0.347775237  0.225410820
fractal_dimension_mean  -0.205792130 -0.296678472 -0.056576036  0.038116744 -0.014618619 -0.142180719
radius_se                0.196164374  0.047007678 -0.269190878  0.109703856 -0.139080611 -0.095773645
texture_se              -0.107218897  0.140076418 -0.188151568 -0.250183339 -0.391001759 -0.206745374
perimeter_se             0.190771209  0.027076418 -0.277094199  0.079714690 -0.140272707 -0.069122772
area_se                  0.231145441  0.061247143 -0.220693603  0.118499754 -0.071806532 -0.084418803
smoothness_se           -0.170772144  0.002608844 -0.258739282  0.125540095 -0.041843311 -0.385650835
compactness_se          -0.042412803 -0.235604257 -0.253941051 -0.236412365  0.068783897  0.093466835
concavity_se            -0.027237703 -0.206811404 -0.288120496 -0.253244576  0.140509773  0.109964924
concave.points_se        0.034752937 -0.157934388 -0.324610999 -0.115583928  0.059090685 -0.020590407
symmetry_se             -0.131410601 -0.020497029 -0.252145154  0.116919287 -0.399842307  0.359923346
fractal_dimension_se    -0.129889458 -0.203511611 -0.299235657 -0.187963264  0.134642008 -0.001714700
radius_worst             0.318051743 -0.002408842  0.025847354  0.015915974  0.001231180  0.005055467
texture_worst            0.071619856  0.073167730  0.261454523 -0.452216642 -0.256835218 -0.179836565
perimeter_worst          0.311376963 -0.024130719  0.018644195  0.004413788 -0.002219419  0.011130842
area_worst               0.311258663  0.015550245 -0.001918674  0.034893550  0.004243826 -0.020892397
smoothness_worst        -0.021763199 -0.238278407  0.241067655  0.222802201 -0.029879936 -0.413403455
compactness_worst        0.071405654 -0.309037963  0.146259435 -0.139611665  0.033960468  0.055266345
concavity_worst          0.114812239 -0.278328383  0.074223352 -0.171883896  0.076134438  0.052182604
concave.points_worst     0.202022267 -0.213855067  0.084136312 -0.022263882  0.023356840 -0.029454089
symmetry_worst          -0.001850243 -0.224726359  0.256603325  0.128834924 -0.286304187  0.415024460
fractal_dimension_worst -0.079062657 -0.341519242  0.148406365 -0.104861644  0.092614311 -0.052821919
> 
```

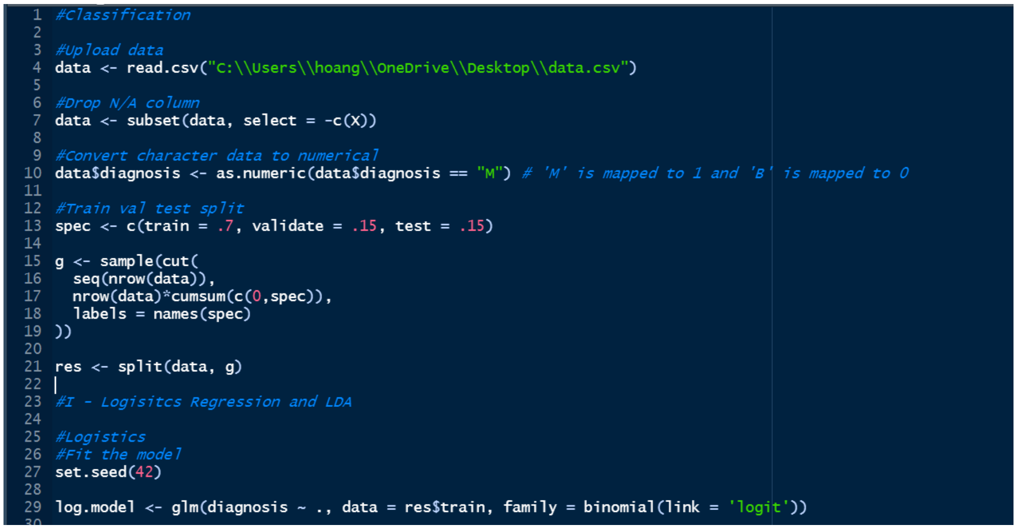
Each number in the PCs indicates the correlation between variables. For example, PC1 indicates slight correlation between perimeter_mean (0.315), area_mean (0.316), etc.

3. Classification

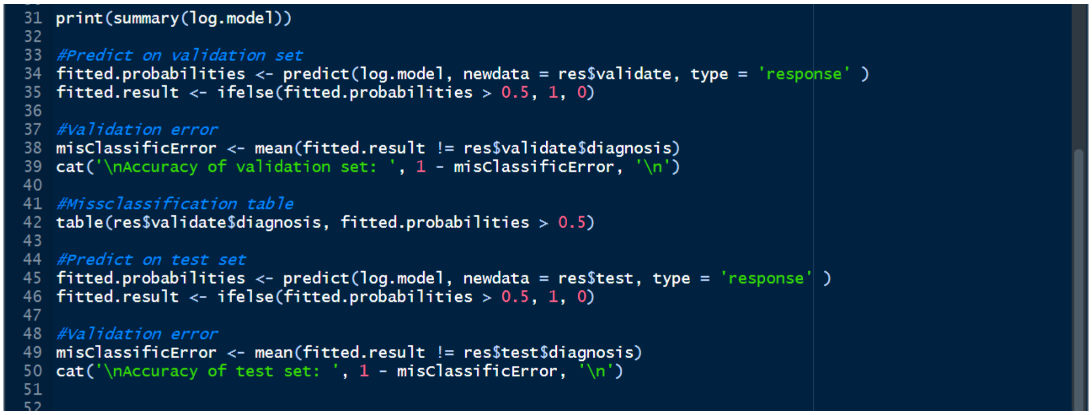
I.

Logistics Regression

Code

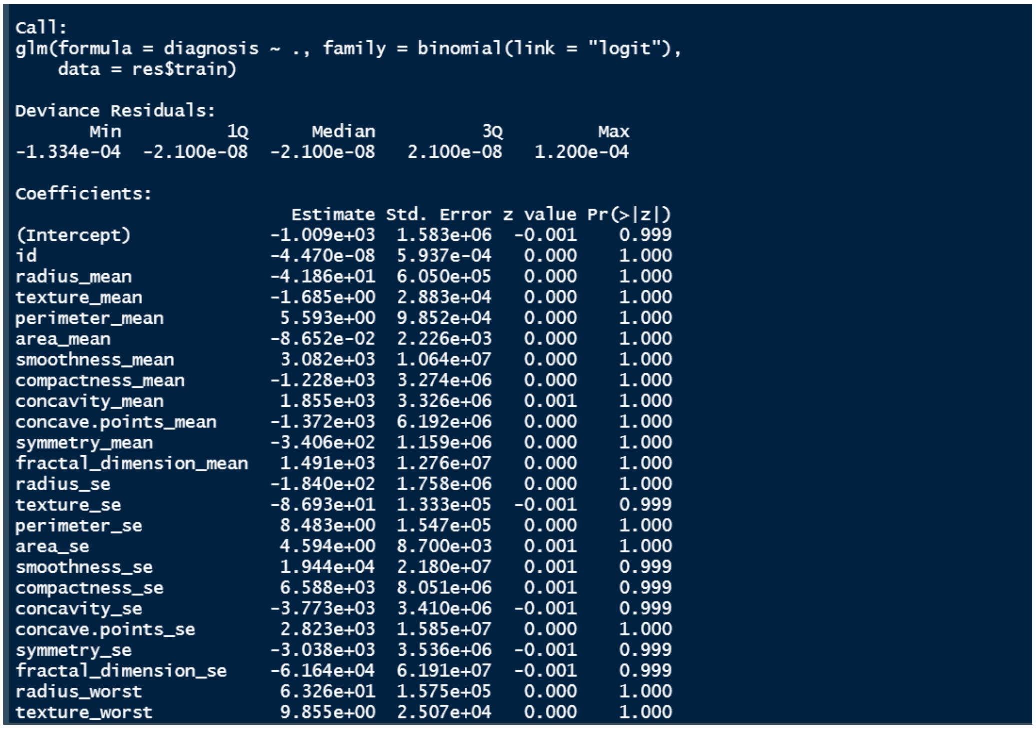


```
#Classification

#Upload data
data <- read.csv("C:\\Users\\hoang\\OneDrive\\Desktop\\data.csv")

#Drop N/A column
data <- subset(data, select = -c(X))

#Convert character data to numerical
data$diagnosis <- as.numeric(data$diagnosis == "M") # 'M' is mapped to 1 and 'B' is mapped to 0

#Train val test split
spec <- c(train = .7, validate = .15, test = .15)

g <- sample(cut(
  seq(nrow(data)),
  nrow(data)*cumsum(c(0,spec)),
  labels = names(spec)
))

res <- split(data, g)

#I - Logisitcs Regression and LDA

#Logistics
#Fit the model
set.seed(42)

log.model <- glm(diagnosis ~ ., data = res$train, family = binomial(link = 'logit'))
```

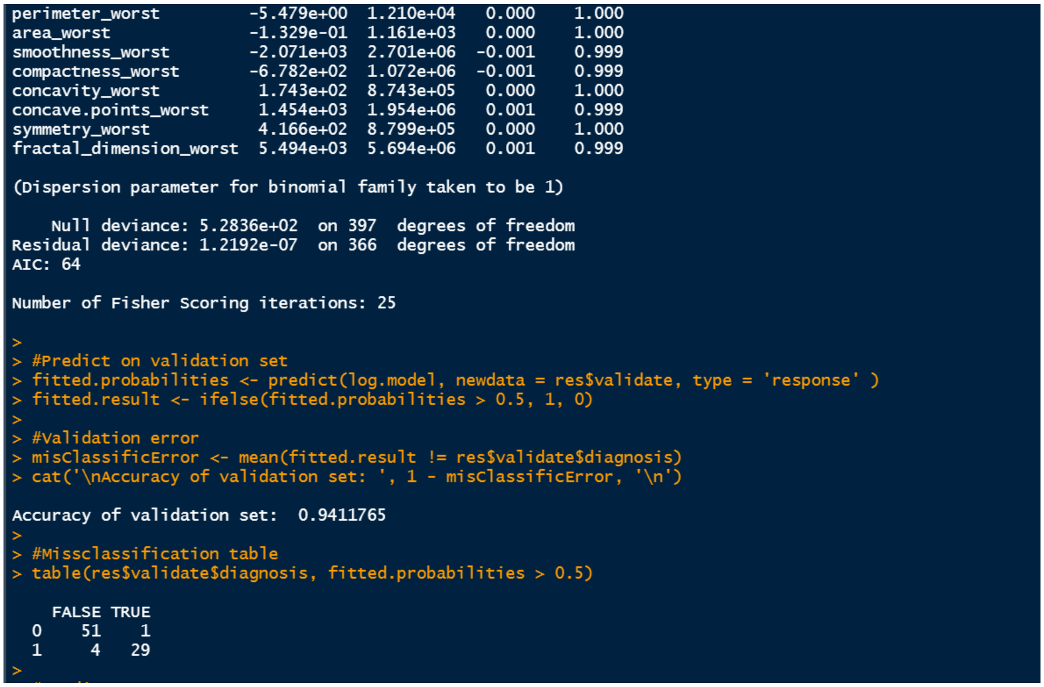


```r
print(summary(log.model))

#Predict on validation set
fitted.probabilities <- predict(log.model, newdata = res$validate, type = 'response' )
fitted.result <- ifelse(fitted.probabilities > 0.5, 1, 0)

#Validation error
misClassificError <- mean(fitted.result != res$validate$diagnosis)
cat('\nAccuracy of validation set: ', 1 - misClassificError, '\n')

#Missclassification table
table(res$validate$diagnosis, fitted.probabilities > 0.5)

#Predict on test set
fitted.probabilities <- predict(log.model, newdata = res$test, type = 'response' )
fitted.result <- ifelse(fitted.probabilities > 0.5, 1, 0)

#Validation error
misClassificError <- mean(fitted.result != res$test$diagnosis)
cat('\nAccuracy of test set: ', 1 - misClassificError, '\n')
```

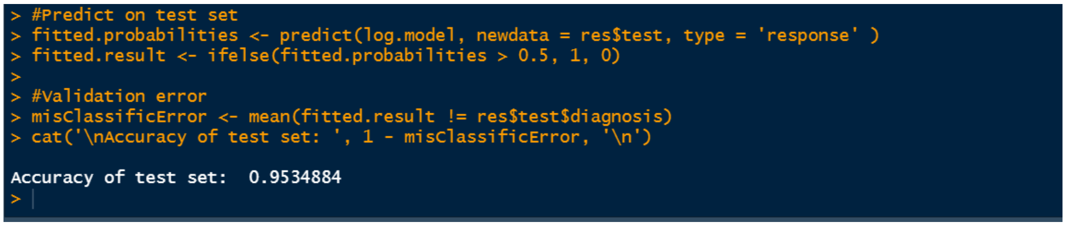
Code output:



```
Call:
glm(formula = diagnosis ~ ., family = binomial(link = "logit"),
    data = res$train)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.334e-04  -2.100e-08  -2.100e-08   2.100e-08   1.200e-04

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -1.009e+03  1.583e+06  -0.001    0.999
id                      -4.470e-08  5.937e-04   0.000    1.000
radius_mean             -4.186e+01  6.050e+05   0.000    1.000
texture_mean            -1.685e+00  2.883e+04   0.000    1.000
perimeter_mean           5.593e+00  9.852e+04   0.000    1.000
area_mean               -8.652e-02  2.226e+03   0.000    1.000
smoothness_mean          3.082e+03  1.064e+07   0.000    1.000
compactness_mean        -1.228e+03  3.274e+06   0.000    1.000
concavity_mean           1.855e+03  3.326e+06   0.001    1.000
concave.points_mean     -1.372e+03  6.192e+06   0.000    1.000
symmetry_mean           -3.406e+02  1.159e+06   0.000    1.000
fractal_dimension_mean   1.491e+03  1.276e+07   0.000    1.000
radius_se               -1.840e+02  1.758e+06   0.000    1.000
texture_se              -8.693e+01  1.333e+05  -0.001    0.999
perimeter_se             8.483e+00  1.547e+05   0.000    1.000
area_se                  4.594e+00  8.700e+03   0.001    1.000
smoothness_se            1.944e+04  2.180e+07   0.001    0.999
compactness_se           6.588e+03  8.051e+06   0.001    0.999
concavity_se            -3.773e+03  3.410e+06  -0.001    0.999
concave.points_se        2.823e+03  1.585e+07   0.000    1.000
symmetry_se             -3.038e+03  3.536e+06  -0.001    0.999
fractal_dimension_se    -6.164e+04  6.191e+07  -0.001    0.999
radius_worst             6.326e+01  1.575e+05   0.000    1.000
texture_worst            9.855e+00  2.507e+04   0.000    1.000
```



```
perimeter_worst         -5.479e+00  1.210e+04   0.000    1.000
area_worst              -1.329e-01  1.161e+03   0.000    1.000
smoothness_worst        -2.071e+03  2.701e+06  -0.001    0.999
compactness_worst       -6.782e+02  1.072e+06  -0.001    0.999
concavity_worst          1.743e+02  8.743e+05   0.000    1.000
concave.points_worst     1.454e+03  1.954e+06   0.001    0.999
symmetry_worst           4.166e+02  8.799e+05   0.000    1.000
fractal_dimension_worst  5.494e+03  5.694e+06   0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5.2836e+02  on 397  degrees of freedom
Residual deviance: 1.2192e-07  on 366  degrees of freedom
AIC: 64

Number of Fisher Scoring iterations: 25

> 
> #Predict on validation set
> fitted.probabilities <- predict(log.model, newdata = res$validate, type = 'response' )
> fitted.result <- ifelse(fitted.probabilities > 0.5, 1, 0)
> 
> #Validation error
> misClassificError <- mean(fitted.result != res$validate$diagnosis)
> cat('\nAccuracy of validation set: ', 1 - misClassificError, '\n')

Accuracy of validation set:  0.9411765 
> 
> #Missclassification table
> table(res$validate$diagnosis, fitted.probabilities > 0.5)
   
    FALSE TRUE
  0    51    1
  1     4   29
> 
```

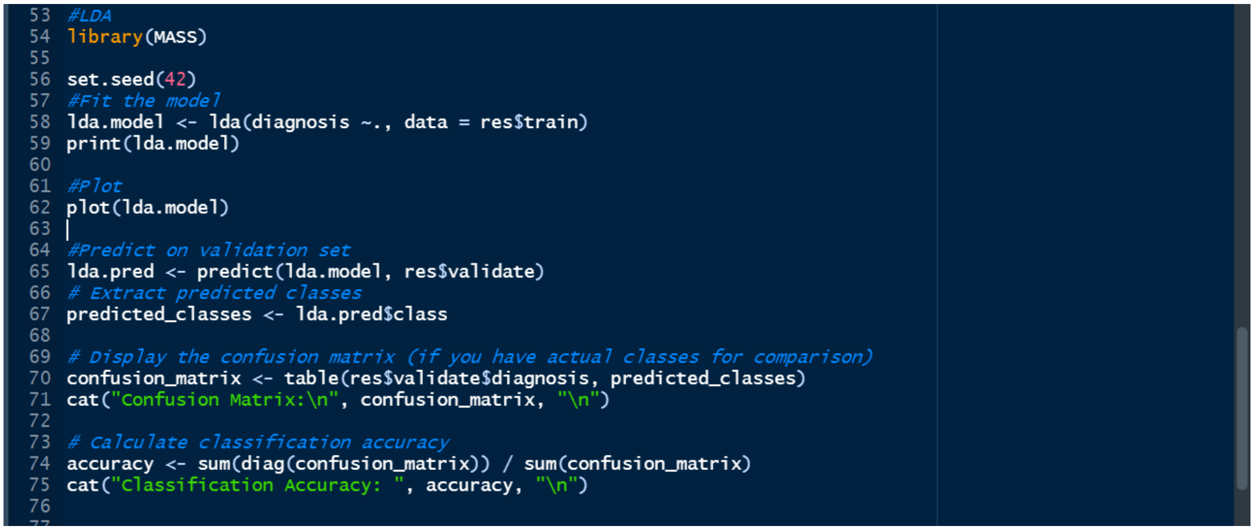


```
> #Predict on test set
> fitted.probabilities <- predict(log.model, newdata = res$test, type = 'response' )
> fitted.result <- ifelse(fitted.probabilities > 0.5, 1, 0)
>
> #Validation error
> misClassificError <- mean(fitted.result != res$test$diagnosis)
> cat('\nAccuracy of test set: ', 1 - misClassificError, '\n')

Accuracy of test set:  0.9534884
>
```

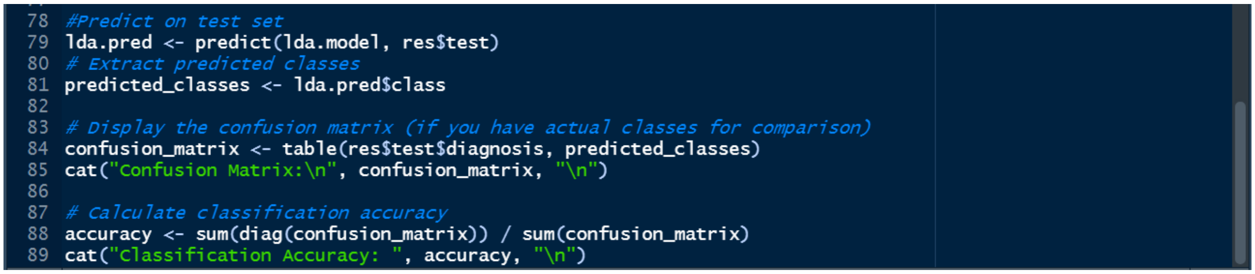
The accuracy of the test set is higher than validation set indicate the model is not overfitting.

Linear Discriminant Analysis

Code

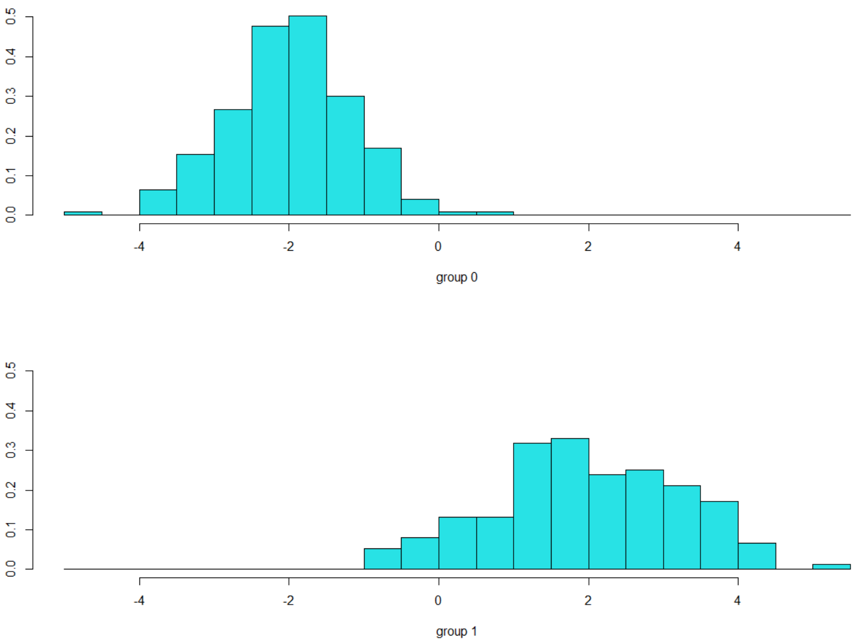


```
#LDA
library(MASS)

set.seed(42)
#Fit the model
lda.model <- lda(diagnosis ~., data = res$train)
print(lda.model)

#Plot
plot(lda.model)

#Predict on validation set
lda.pred <- predict(lda.model, res$validate)
# Extract predicted classes
predicted_classes <- lda.pred$class

# Display the confusion matrix (if you have actual classes for comparison)
confusion_matrix <- table(res$validate$diagnosis, predicted_classes)
cat("Confusion Matrix:\n", confusion_matrix, "\n")

# Calculate classification accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Classification Accuracy: ", accuracy, "\n")

```

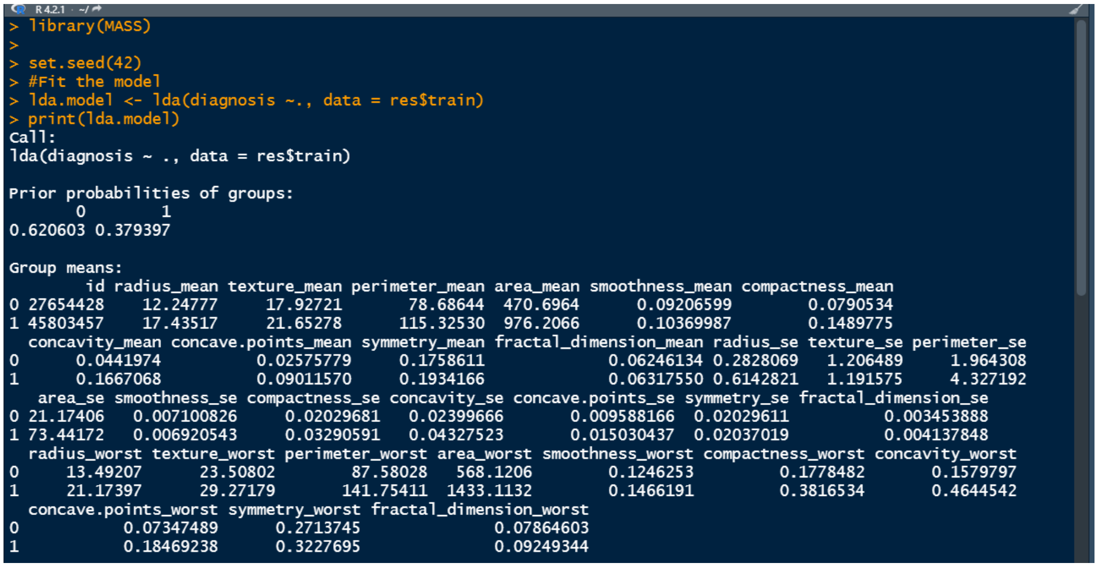


```
#Predict on test set
lda.pred <- predict(lda.model, res$test)
# Extract predicted classes
predicted_classes <- lda.pred$class

# Display the confusion matrix (if you have actual classes for comparison)
confusion_matrix <- table(res$test$diagnosis, predicted_classes)
cat("Confusion Matrix:\n", confusion_matrix, "\n")

# Calculate classification accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Classification Accuracy: ", accuracy, "\n")
```

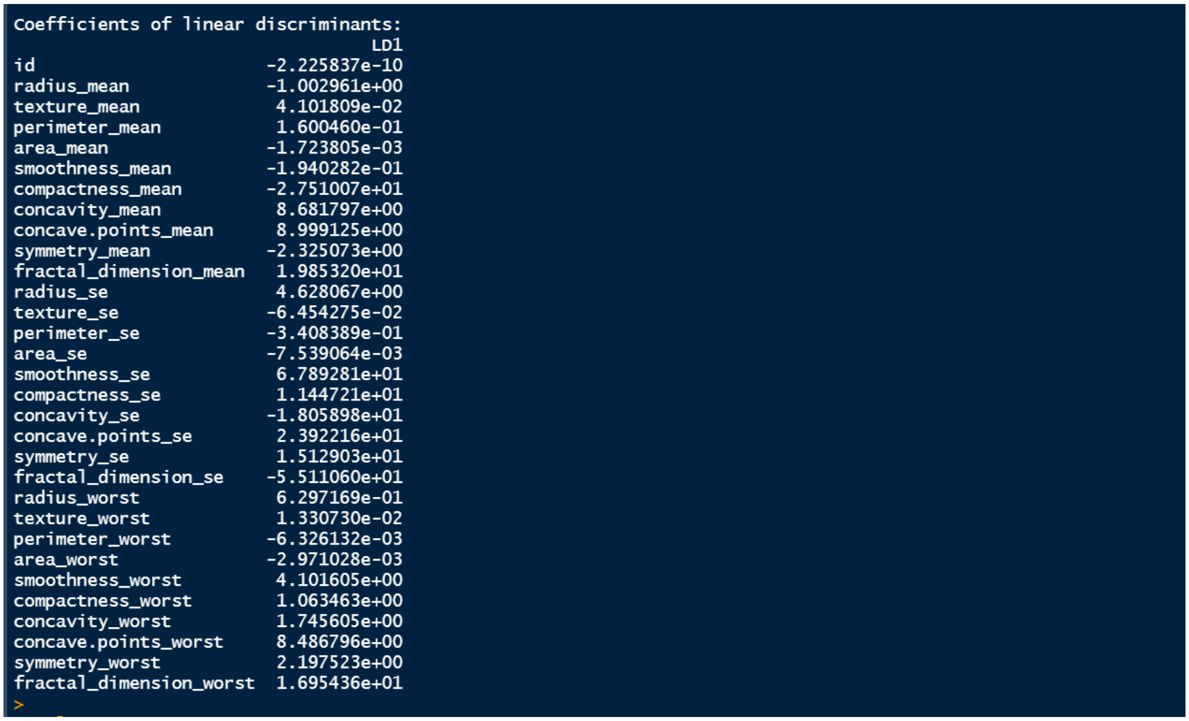
Plot



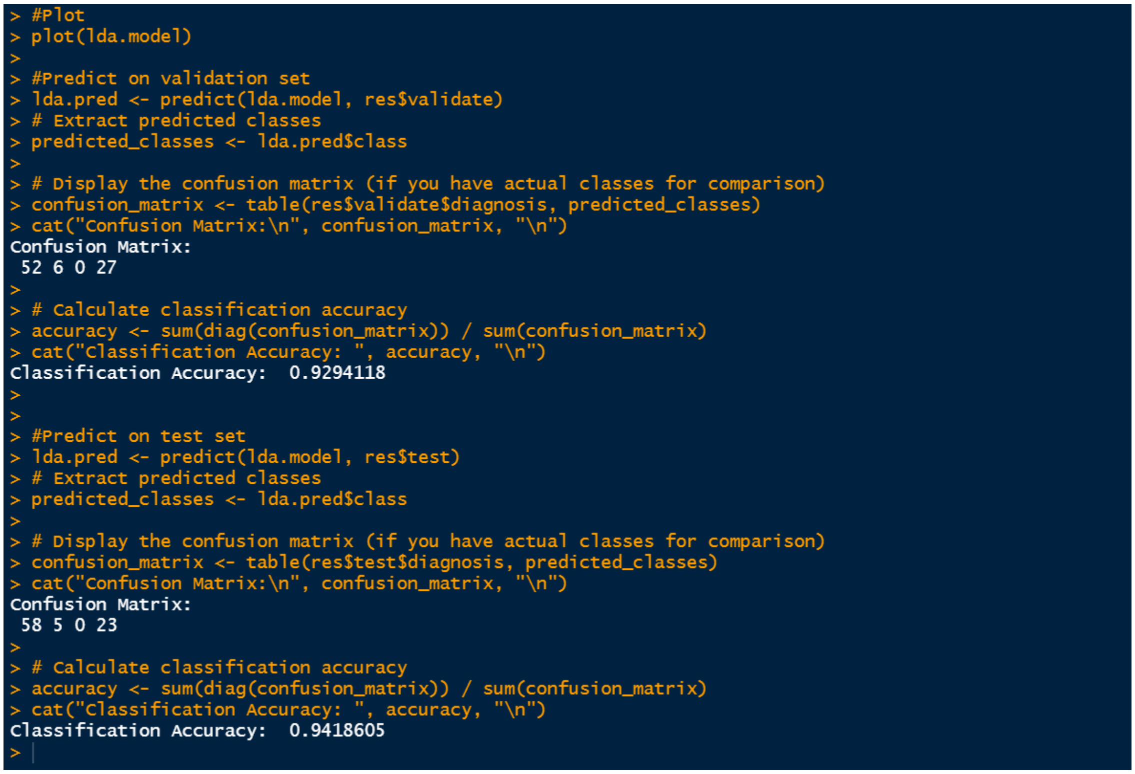
Code output



```
> library(MASS)
>
> set.seed(42)
> #Fit the model
> lda.model <- lda(diagnosis ~., data = res$train)
> print(lda.model)
Call:
lda(diagnosis ~ ., data = res$train)

Prior probabilities of groups:
       0        1 
0.620603 0.379397 

Group means:
        id radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean
0 27654428    12.24777     17.92721       78.68644  470.6964      0.09206599        0.0790534
1 45803457    17.43517     21.65278      115.32530  976.2066      0.10369987        0.1489775
  concavity_mean concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
0      0.0441974          0.02575779     0.1758611             0.06246134 0.2828069   1.206489     1.964308
1      0.1667068          0.09011570     0.1934166             0.06317550 0.6142821   1.191575     4.327192
   area_se smoothness_se compactness_se concavity_se concave.points_se symmetry_se fractal_dimension_se
0 21.17406   0.007100826     0.02029681   0.02399666       0.009588166  0.02029611          0.003453888
1 73.44172   0.006920543     0.03290591   0.04327523       0.015030437  0.02037019          0.004137848
  radius_worst texture_worst perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
0     13.49207      23.50802        87.58028   568.1206        0.1246253         0.1778482       0.1579797
1     21.17397      29.27179       141.75411  1433.1132        0.1466191         0.3816534       0.4644542
  concave.points_worst symmetry_worst fractal_dimension_worst
0           0.07347489      0.2713745              0.07864603
1           0.18469238      0.3227695              0.09249344
```



```
Coefficients of linear discriminants:
                                  LD1
id                      -2.225837e-10
radius_mean             -1.002961e+00
texture_mean             4.101809e-02
perimeter_mean           1.600460e-01
area_mean               -1.723805e-03
smoothness_mean         -1.940282e-01
compactness_mean        -2.751007e+01
concavity_mean           8.681797e+00
concave.points_mean      8.999125e+00
symmetry_mean           -2.325073e+00
fractal_dimension_mean   1.985320e+01
radius_se                4.628067e+00
texture_se              -6.454275e-02
perimeter_se            -3.408389e-01
area_se                 -7.539064e-03
smoothness_se            6.789281e+01
compactness_se           1.144721e+01
concavity_se            -1.805898e+01
concave.points_se        2.392216e+01
symmetry_se              1.512903e+01
fractal_dimension_se    -5.511060e+01
radius_worst             6.297169e-01
texture_worst            1.330730e-02
perimeter_worst         -6.326132e-03
area_worst              -2.971028e-03
smoothness_worst         4.101605e+00
compactness_worst        1.063463e+00
concavity_worst          1.745605e+00
concave.points_worst     8.486796e+00
symmetry_worst           2.197523e+00
fractal_dimension_worst  1.695436e+01
>
```



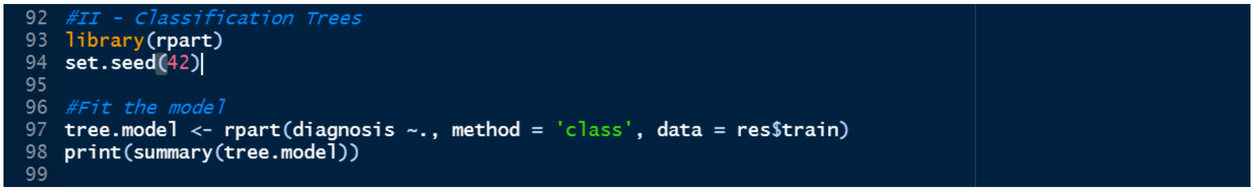
```
> #Plot
> plot(lda.model)
>
> #Predict on validation set
> lda.pred <- predict(lda.model, res$validate)
> # Extract predicted classes
> predicted_classes <- lda.pred$class
>
> # Display the confusion matrix (if you have actual classes for comparison)
> confusion_matrix <- table(res$validate$diagnosis, predicted_classes)
> cat("Confusion Matrix:\n", confusion_matrix, "\n")
Confusion Matrix:
 52 6 0 27
>
> # Calculate classification accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> cat("Classification Accuracy: ", accuracy, "\n")
Classification Accuracy:  0.9294118
>
>
> #Predict on test set
> lda.pred <- predict(lda.model, res$test)
> # Extract predicted classes
> predicted_classes <- lda.pred$class
>
> # Display the confusion matrix (if you have actual classes for comparison)
> confusion_matrix <- table(res$test$diagnosis, predicted_classes)
> cat("Confusion Matrix:\n", confusion_matrix, "\n")
Confusion Matrix:
 58 5 0 23
>
> # Calculate classification accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> cat("Classification Accuracy: ", accuracy, "\n")
Classification Accuracy:  0.9418605
>
```

Again, the accuracy of the test set is higher than the validation set indicating the LDA is not overfitting.

II. Tree classifier

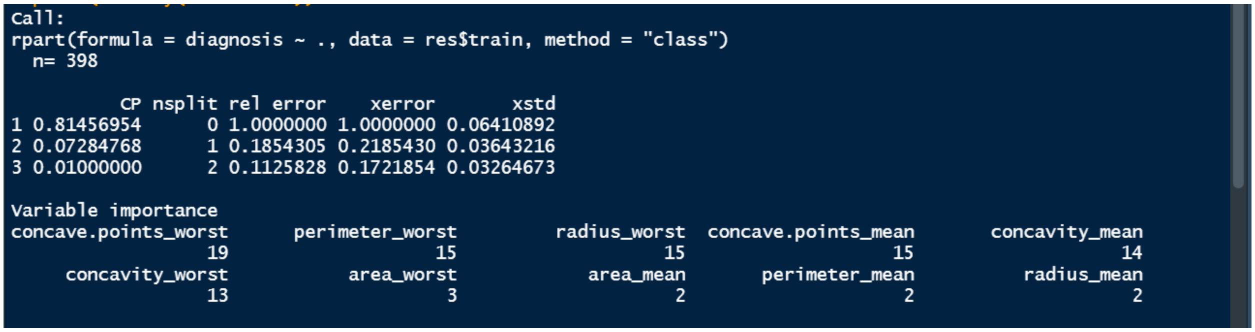
a.

Code

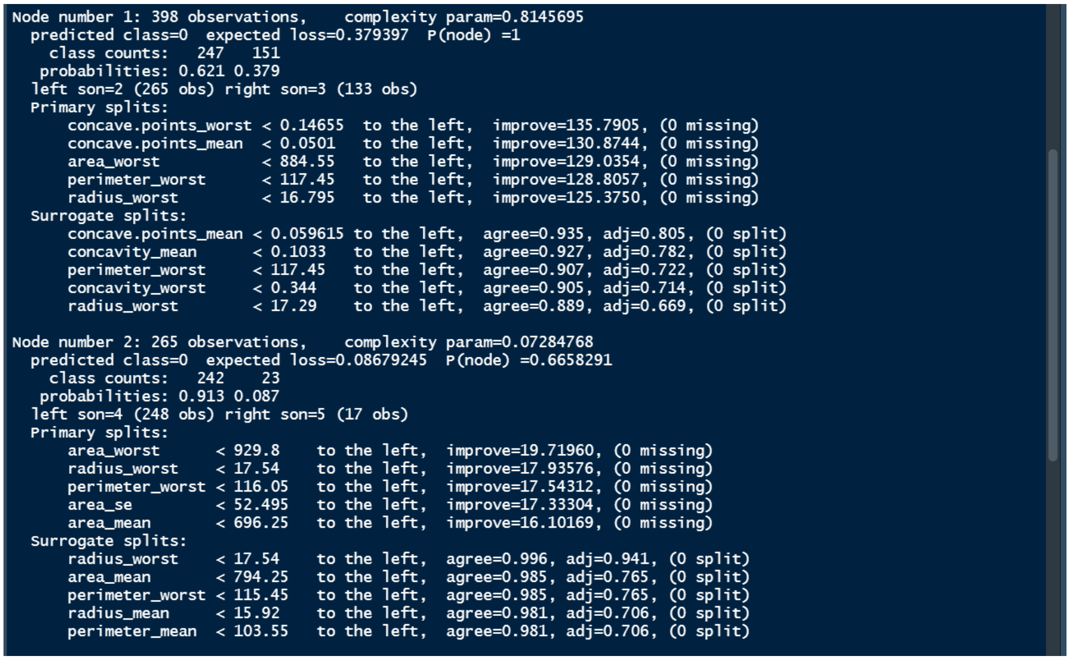


```
#II - Classification Trees
library(rpart)
set.seed(42)

#Fit the model
tree.model <- rpart(diagnosis ~., method = 'class', data = res$train)
print(summary(tree.model))

```

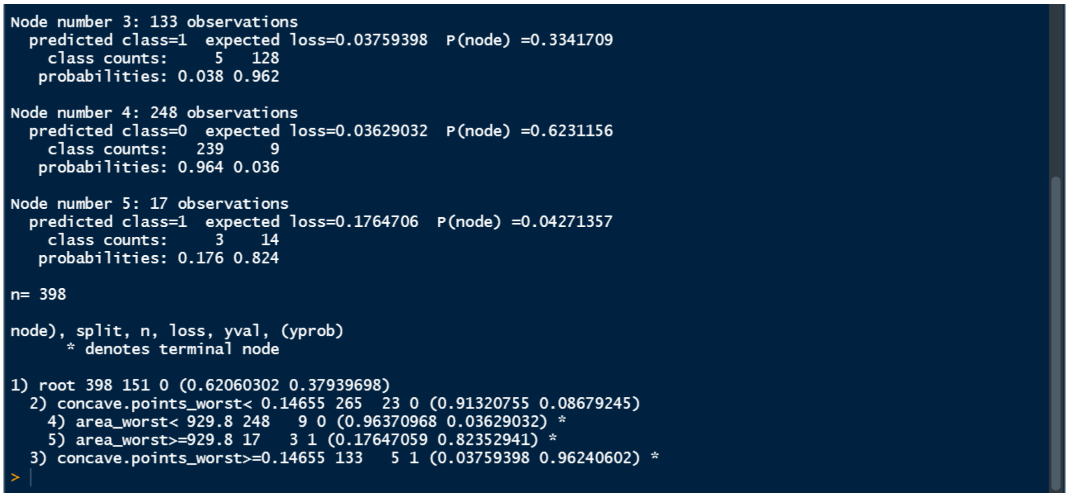
Code output



```
Call:
rpart(formula = diagnosis ~ ., data = res$train, method = "class")
  n= 398

          CP nsplit rel error    xerror       xstd
1 0.81456954      0 1.0000000 1.0000000 0.06410892
2 0.07284768      1 0.1854305 0.2185430 0.03643216
3 0.01000000      2 0.1125828 0.1721854 0.03264673

Variable importance
concave.points_worst      perimeter_worst         radius_worst concave.points_mean       concavity_mean
                  19                   15                   15                  15                   14
     concavity_worst           area_worst            area_mean      perimeter_mean          radius_mean
                  13                    3                    2                   2                    2
```



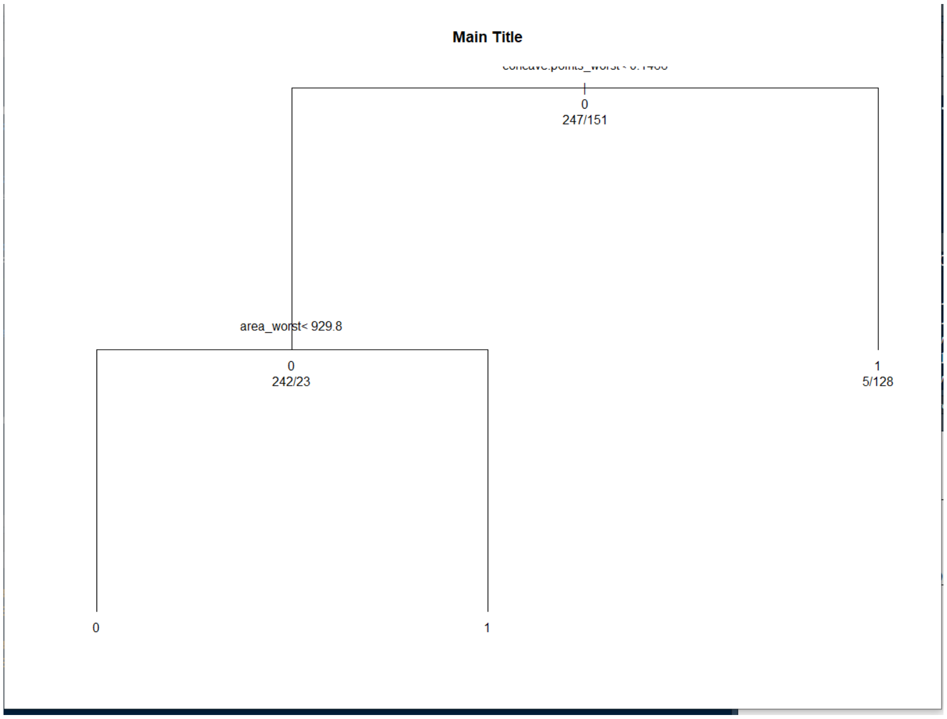
```
Node number 1: 398 observations,    complexity param=0.8145695
  predicted class=0  expected loss=0.379397  P(node) =1
    class counts:   247   151
   probabilities: 0.621 0.379
  left son=2 (265 obs) right son=3 (133 obs)
  Primary splits:
      concave.points_worst < 0.14655  to the left,  improve=135.7905, (0 missing)
      concave.points_mean  < 0.0501   to the left,  improve=130.8744, (0 missing)
      area_worst           < 884.55   to the left,  improve=129.0354, (0 missing)
      perimeter_worst      < 117.45   to the left,  improve=128.8057, (0 missing)
      radius_worst         < 16.795   to the left,  improve=125.3750, (0 missing)
  Surrogate splits:
      concave.points_mean < 0.059615 to the left,  agree=0.935, adj=0.805, (0 split)
      concavity_mean      < 0.1033   to the left,  agree=0.927, adj=0.782, (0 split)
      perimeter_worst     < 117.45   to the left,  agree=0.907, adj=0.722, (0 split)
      concavity_worst     < 0.344    to the left,  agree=0.905, adj=0.714, (0 split)
      radius_worst        < 17.29    to the left,  agree=0.889, adj=0.669, (0 split)

Node number 2: 265 observations,    complexity param=0.07284768
  predicted class=0  expected loss=0.08679245  P(node) =0.6658291
    class counts:   242    23
   probabilities: 0.913 0.087
  left son=4 (248 obs) right son=5 (17 obs)
  Primary splits:
      area_worst      < 929.8    to the left,  improve=19.71960, (0 missing)
      radius_worst    < 17.54    to the left,  improve=17.93576, (0 missing)
      perimeter_worst < 116.05   to the left,  improve=17.54312, (0 missing)
      area_se         < 52.495   to the left,  improve=17.33304, (0 missing)
      area_mean       < 696.25   to the left,  improve=16.10169, (0 missing)
  Surrogate splits:
      radius_worst    < 17.54    to the left,  agree=0.996, adj=0.941, (0 split)
      area_mean       < 794.25   to the left,  agree=0.985, adj=0.765, (0 split)
      perimeter_worst < 115.45   to the left,  agree=0.985, adj=0.765, (0 split)
      radius_mean     < 15.92    to the left,  agree=0.981, adj=0.706, (0 split)
      perimeter_mean  < 103.55   to the left,  agree=0.981, adj=0.706, (0 split)
```



```
Node number 3: 133 observations
  predicted class=1  expected loss=0.03759398  P(node) =0.3341709
    class counts:     5   128
   probabilities: 0.038 0.962

Node number 4: 248 observations
  predicted class=0  expected loss=0.03629032  P(node) =0.6231156
    class counts:   239     9
   probabilities: 0.964 0.036

Node number 5: 17 observations
  predicted class=1  expected loss=0.1764706  P(node) =0.04271357
    class counts:     3    14
   probabilities: 0.176 0.824

n= 398

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 398 151 0 (0.62060302 0.37939698)
  2) concave.points_worst< 0.14655 265  23 0 (0.91320755 0.08679245)
    4) area_worst< 929.8 248   9 0 (0.96370968 0.03629032) *
    5) area_worst>=929.8 17   3 1 (0.17647059 0.82352941) *
  3) concave.points_worst>=0.14655 133   5 1 (0.03759398 0.96240602) *
> 
```

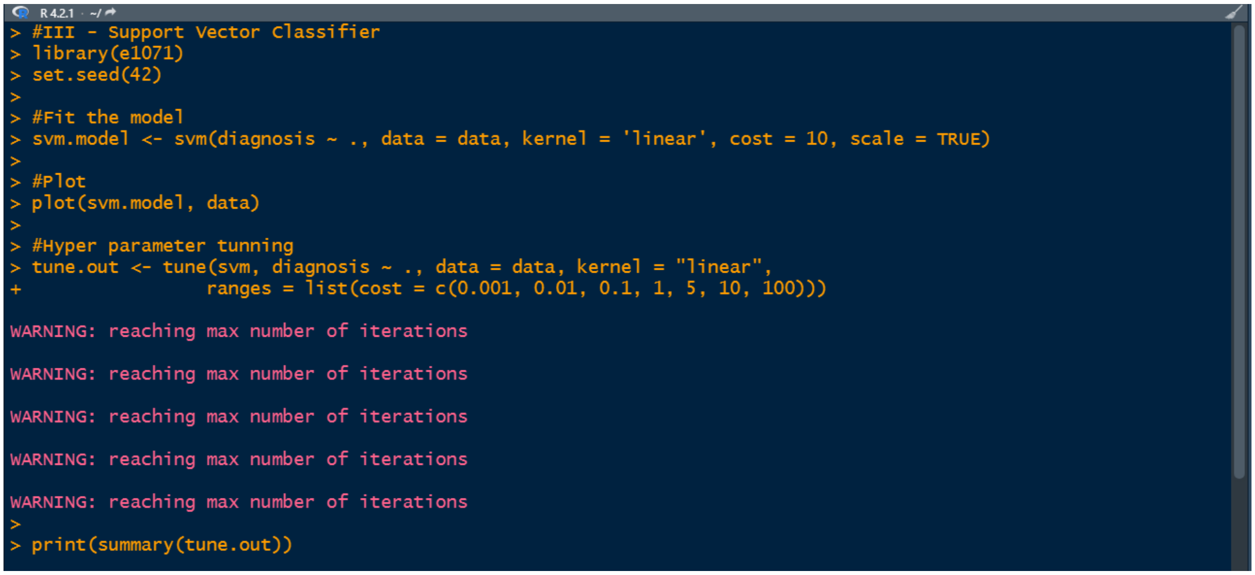
b.

Plot

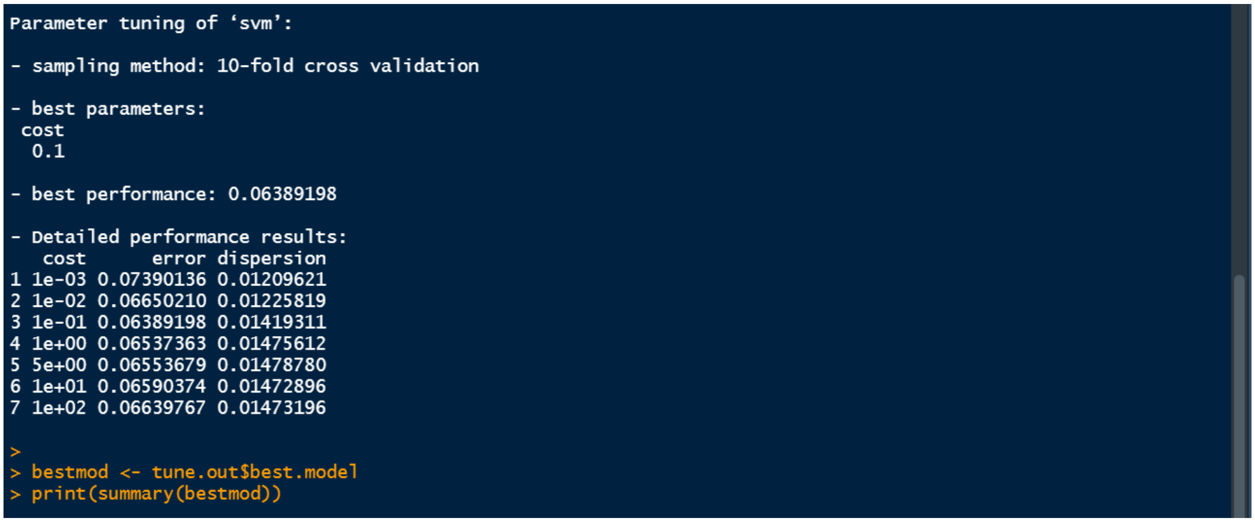


III. Support Vector Classifier

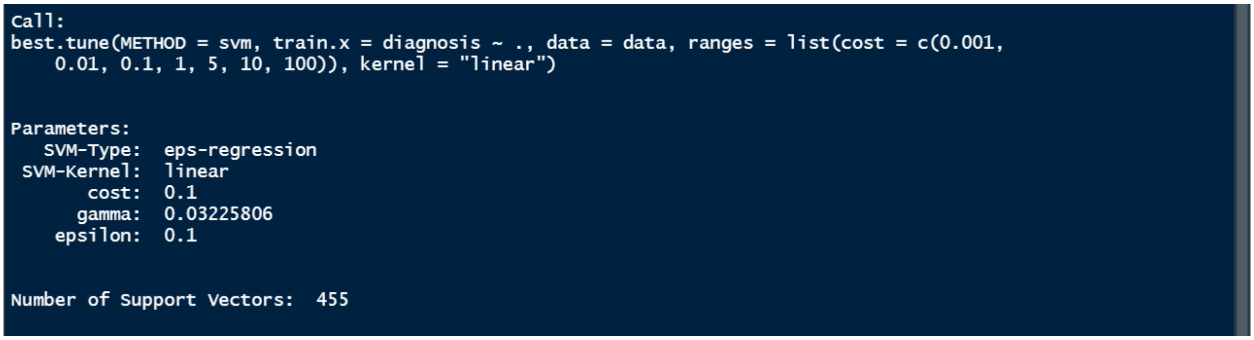
Code and output



```
> #III - Support Vector Classifier
> library(e1071)
> set.seed(42)
>
> #Fit the model
> svm.model <- svm(diagnosis ~ ., data = data, kernel = 'linear', cost = 10, scale = TRUE)
>
> #Plot
> plot(svm.model, data)
>
> #Hyper parameter tunning
> tune.out <- tune(svm, diagnosis ~ ., data = data, kernel = "linear",
+                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))

WARNING: reaching max number of iterations

WARNING: reaching max number of iterations

WARNING: reaching max number of iterations

WARNING: reaching max number of iterations

WARNING: reaching max number of iterations
>
> print(summary(tune.out))
```



```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.06389198

- Detailed performance results:
   cost      error dispersion
1 1e-03 0.07390136 0.01209621
2 1e-02 0.06650210 0.01225819
3 1e-01 0.06389198 0.01419311
4 1e+00 0.06537363 0.01475612
5 5e+00 0.06553679 0.01478780
6 1e+01 0.06590374 0.01472896
7 1e+02 0.06639767 0.01473196

>
> bestmod <- tune.out$best.model
> print(summary(bestmod))
```



```
Call:
best.tune(METHOD = svm, train.x = diagnosis ~ ., data = data, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.03225806
    epsilon:  0.1


Number of Support Vectors:  455
```

4. Models

a. Multiple Linear Regression. Because multiple linear regression allows modeling the relationship between a continuous response variable (like profits) and multiple explanatory variables (like inventory, employees, budget etc). It can determine how strongly each factor correlates with profits (the measure of success). Also, multiple regression allows quantifying the correlation and predictive relationships between inventory, employees, budget, and other numerical factors to the level of business success measured by profits. This can give actionable and data-driven insights to my friend on what drives success in their industry. The interpretation is also straightforward.

b. Logistics Regression. Because logistic regression is well-suited for binary outcomes, such as whether or not a shopper is likely to visit the store (1 for likely, 0 for not likely). In this case, the outcome variable can be binary based on the likelihood response in the survey data. Also, logistic regression provides interpretable coefficients, making it easier to understand the impact of predictor variables on the likelihood of shoppers visiting the store. This can be valuable when explaining the model to stakeholders.

c. Principal Component Analysis or Factor Analysis. Because this scenario involves dimensionality reduction techniques, and PCA is specifically designed to reduce the dimensionality of the data while preserving as much of the original variance as possible. With thousands of features, PCA can transform the data into a lower-dimensional space, capturing the most important patterns and reducing computational complexity. PCA creates new features, called principal components, that are linear combinations of the original features. These components are orthogonal and ordered by their importance. The resulting principal components can provide insight into the most significant patterns in the data, aiding interpretability.

d. Random Forest. First of all, because I’m a big fan of random forest when it comes to machine learning problem, random forest is the first thing that comes to my mind. Random Forest can handle complex relationships in the data and is robust to noisy and imprecise features. It builds multiple decision trees and combines their prediction. I’d choose Random Forest because of its simplicity and efficiency.