SDM_Assignment4_3

Sri Balaji Muruganandam

15/11/2021

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

Setting Working Directory

```
rm(list = ls())
setwd("G:\\SDM_Sem01\\Assignment4")
```

Importing necessary libraries

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.1
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.1.1
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.1
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.1
```

```
## Loading required package: lattice
```

```
library(MASS)
library(corrplot)
```

```
## corrplot 0.91 loaded
```

Loading Auto data set

```
data(Auto)
dim(Auto)
```

```
## [1] 392   9
```

```
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

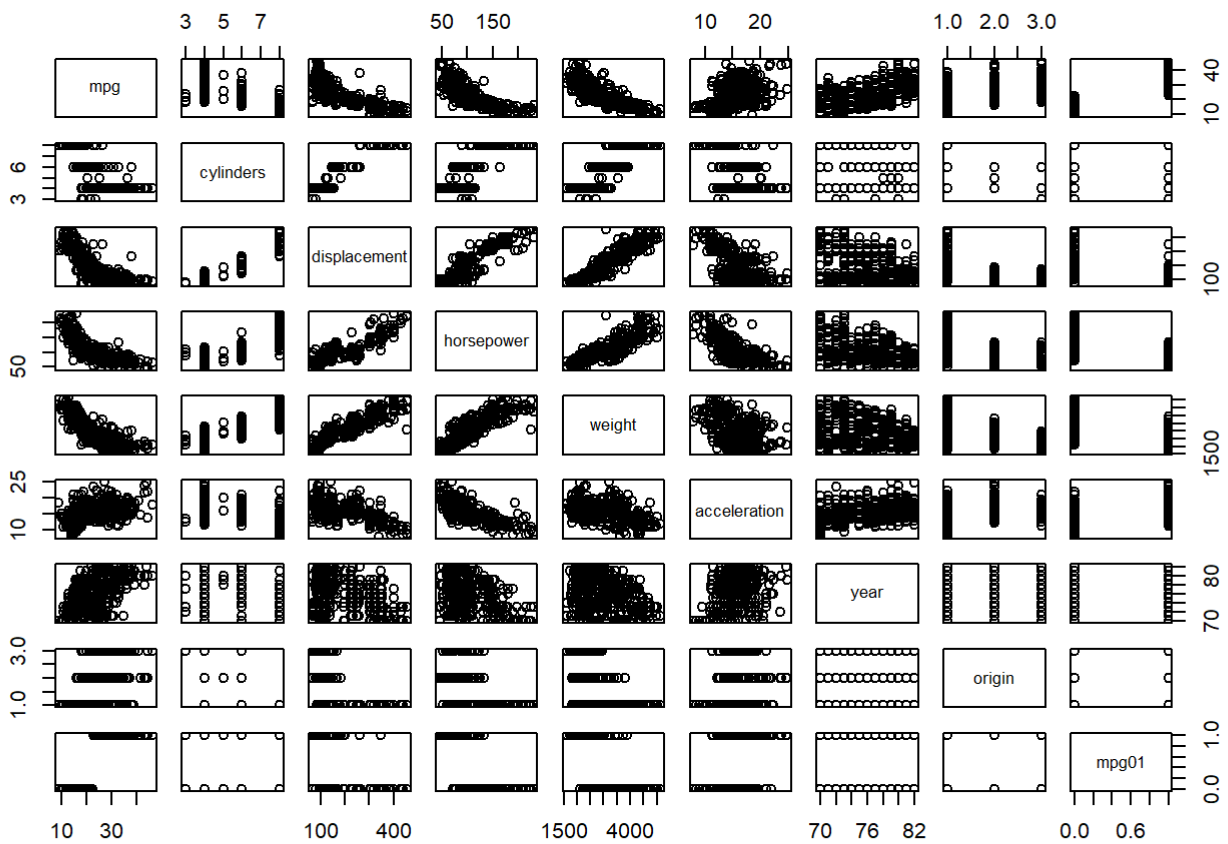
```
mpg_medi = median(Auto$mpg)
data = data.frame(Auto)
data = data[,-9]

data$mpg01[data$mpg>mpg_medi] <- 1
data$mpg01[data$mpg<=mpg_medi] <- 0
tail(data,13)
```

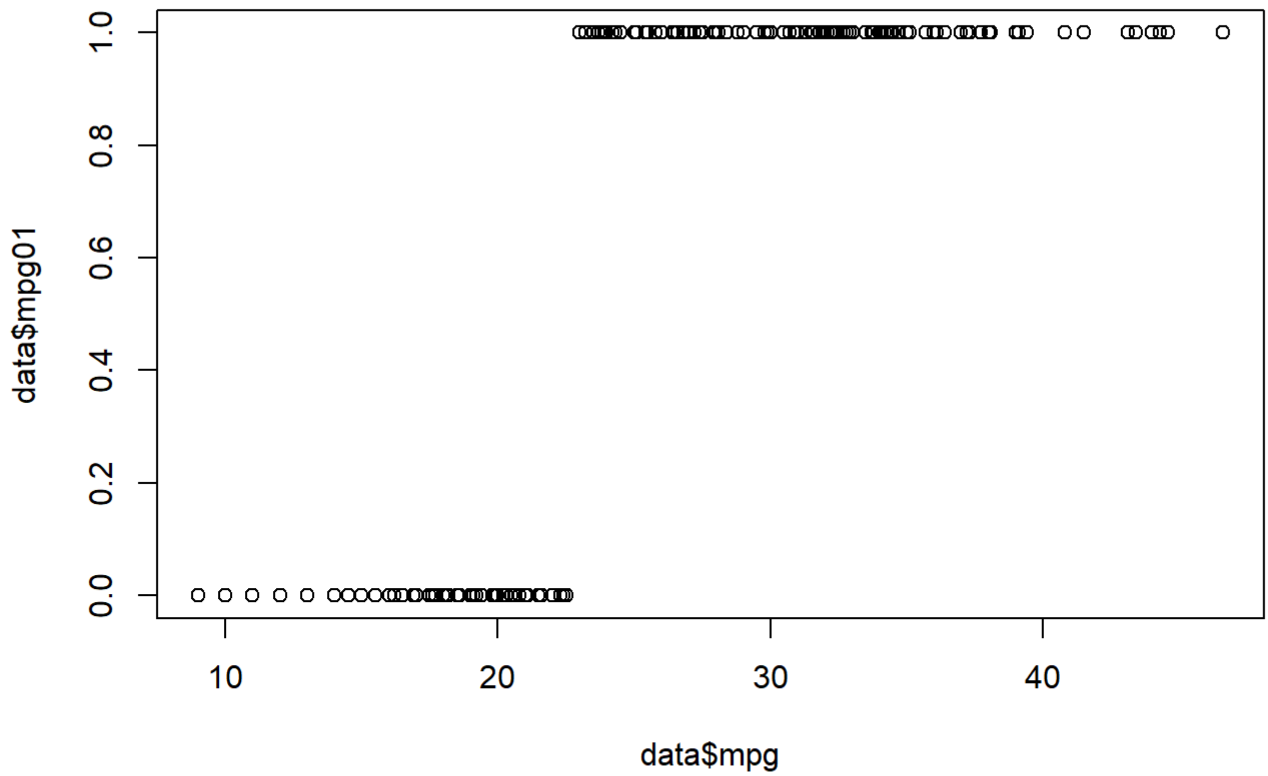
```
##     mpg cylinders displacement horsepower weight acceleration year origin mpg01
## 385  38         4           91         67   1995         16.2   82      3     1
## 386  25         6          181        110   2945         16.4   82      1     1
## 387  38         6          262         85   3015         17.0   82      1     1
## 388  26         4          156         92   2585         14.5   82      1     1
## 389  22         6          232        112   2835         14.7   82      1     0
## 390  32         4          144         96   2665         13.9   82      3     1
## 391  36         4          135         84   2370         13.0   82      1     1
## 392  27         4          151         90   2950         17.3   82      1     1
## 393  27         4          140         86   2790         15.6   82      1     1
## 394  44         4           97         52   2130         24.6   82      2     1
## 395  32         4          135         84   2295         11.6   82      1     1
## 396  28         4          120         79   2625         18.6   82      1     1
## 397  31         4          119         82   2720         19.4   82      1     1
```

(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

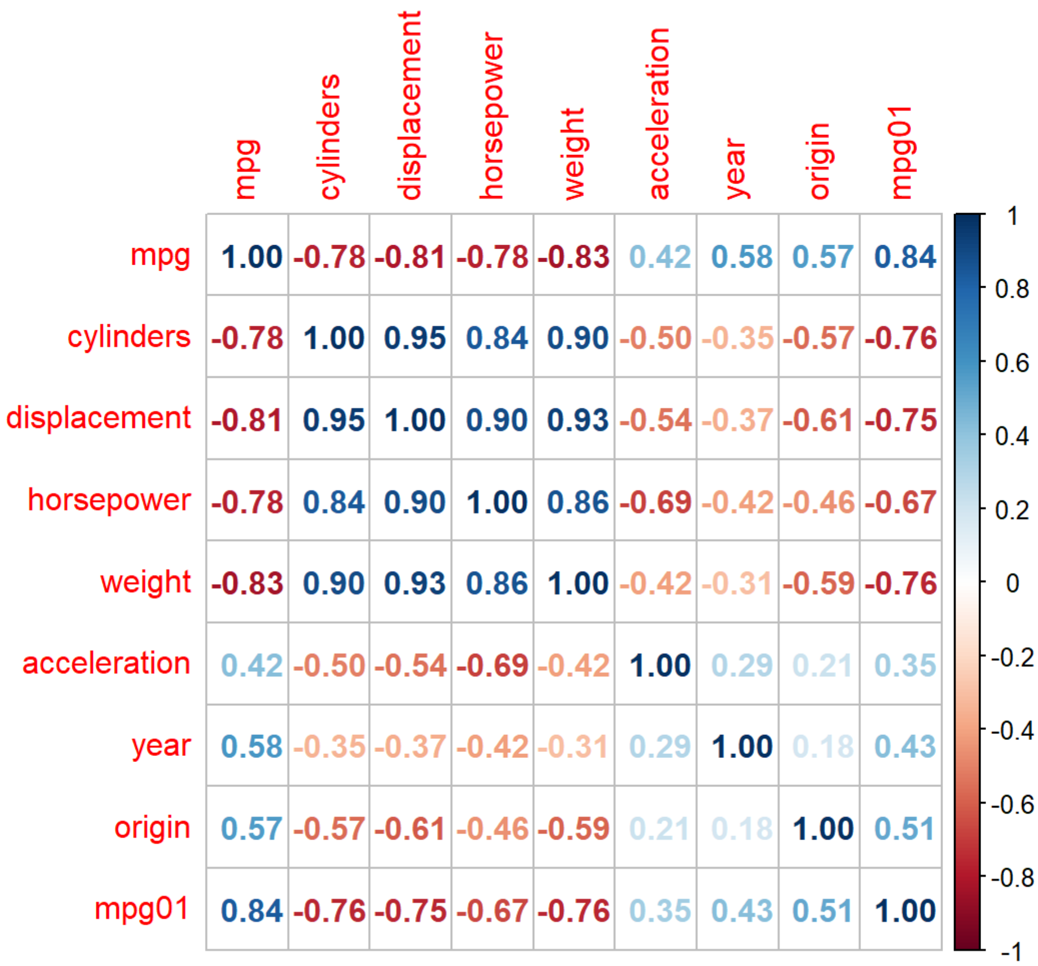
```
plot(data)
```




```
plot(data$mpg, data$mpg01)
```




```
corrplot(cor(data), method = 'number')
```



From the correlation it is observed that the column mpg, accleration, year and origin are useful in predicting mpg01

Removing unwanted columns from the dataset

```
head(data,4)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin mpg01
## 1  18         8          307        130   3504         12.0   70      1     0
## 2  15         8          350        165   3693         11.5   70      1     0
## 3  18         8          318        150   3436         11.0   70      1     0
## 4  16         8          304        150   3433         12.0   70      1     0
```

```
data = subset(data, select = c(1,2,6,7,8,9) )
head(data,5)
```

```
##   mpg cylinders acceleration year origin mpg01
## 1  18         8         12.0   70      1     0
## 2  15         8         11.5   70      1     0
## 3  18         8         11.0   70      1     0
## 4  16         8         12.0   70      1     0
## 5  17         8         10.5   70      1     0
```

(c) Split the data into a training set and a test set.

```
set.seed(23)
random_index = sample(c(1:nrow(data)), size = round(8/10 * nrow(data)), replace = FALSE)
train_data <- data[random_index,]
test_data <- data[-random_index,]

y_train_data = train_data$mpg01
y_test_data = test_data$mpg01

dim(train_data)
```

```
## [1] 314   6
```

```
dim(test_data)
```

```
## [1] 78  6
```

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

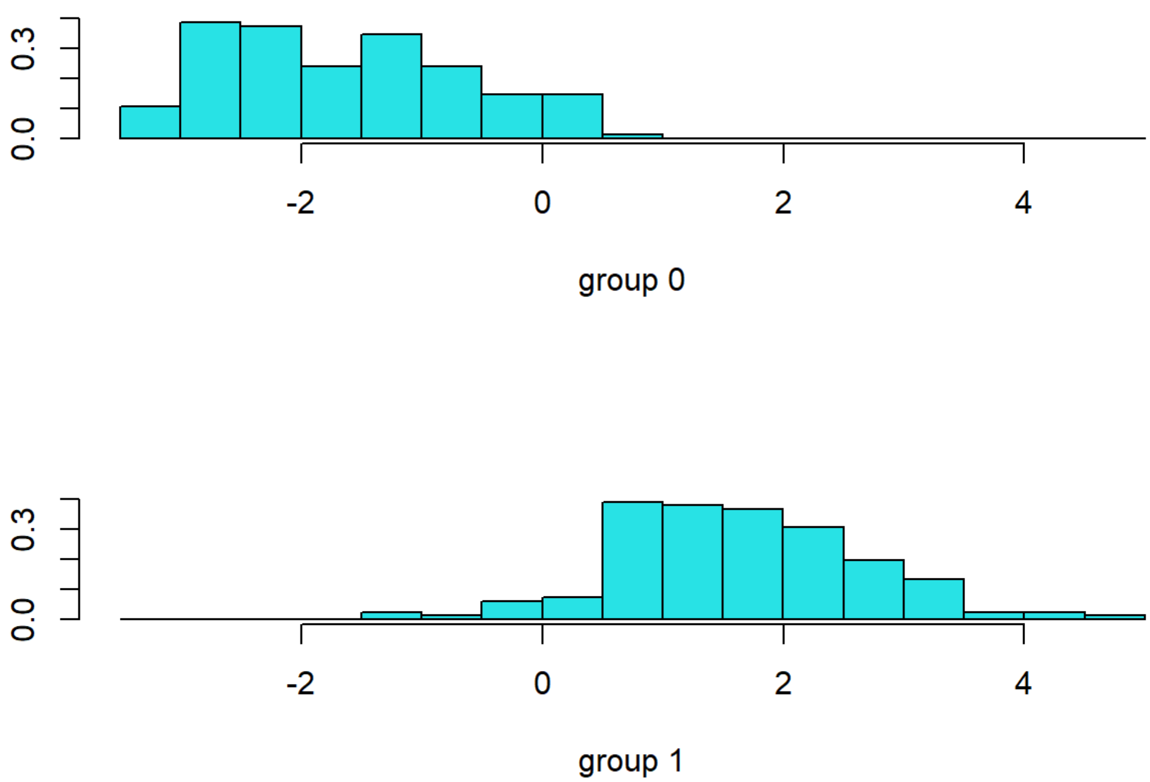
Modelling LDA

```
lda_model = lda(mpg01~., data = train_data)
summary(lda_model)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   10     -none- numeric
## scaling  5     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

Plotting the values of LDA

```
plot(lda_model)
```



Predicting for test data

```
lda_predict_train = predict(lda_model, newdata = train_data)
lda_predict_test = predict(lda_model, newdata = test_data)
res_train = lda_predict_train$class
res_test = lda_predict_test$class
```

Calculating the train and test errors

```
lda_result_train = which(res_train!=y_train_data)
lda_train_error=length(lda_result_train) / length(y_train_data)
print(lda_train_error)
```

```
## [1] 0.06050955
```

```
lda_result_test = which(res_test!=y_test_data)
lda_test_error=length(lda_result_test) / length(y_test_data)
print(lda_test_error)
```

```
## [1] 0.02564103
```

The train error is 0.06369427 and the test error is 0.02564103 when predicted using LDA.

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

Modelling QDA

```
qda_model = qda(mpg01~., data = train_data)
summary(qda_model)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   10     -none- numeric
## scaling 50     -none- numeric
## ldet     2     -none- numeric
## lev      2     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

Predicting for test data

```
qda_predict_train = predict(qda_model, newdata = train_data)
qda_predict_test = predict(qda_model, newdata = test_data)
res_trainq = qda_predict_train$class
res_testq = qda_predict_test$class
```

Calculating the train and test errors

```
qda_result_train = which(res_trainq!=y_train_data)
qda_train_error=length(qda_result_train) / length(y_train_data)
print(qda_train_error)
```

```
## [1] 0.05414013
```

```
qda_result_test = which(res_testq!=y_test_data)
qda_test_error=length(qda_result_test) / length(y_test_data)
print(qda_test_error)
```

```
## [1] 0.03846154
```

The train error is 0.05414013 and the test error is 0.03846154 when predicted using QDA.

(f) Perform logistic regression on the training data to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

Modelling Logistic Regression

```
model = glm(mpg01~., data = train_data, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model)
```

```
##
## Call:
## glm(formula = mpg01 ~ ., family = "binomial", data = train_data)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.933e-04  -2.000e-08   2.000e-08   2.000e-08   2.560e-04
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.539e+03  1.670e+05  -0.009    0.993
## mpg           6.853e+01  6.973e+03   0.010    0.992
## cylinders    -2.511e-02  2.583e+03   0.000    1.000
## acceleration -2.885e-01  2.392e+03   0.000    1.000
## year         -2.213e-01  2.056e+03   0.000    1.000
## origin        2.616e+00  1.646e+04   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4.3467e+02  on 313  degrees of freedom
## Residual deviance: 2.9747e-07  on 308  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

Predicting for new values

```
test_predict = predict(model, newdata = test_data, type = "response")
y_predict_test = round(test_predict)
```

Calculating the error

```
test_error <- sum(abs(y_predict_test- y_test_data))/length(y_test_data)
print(test_error)
```

```
## [1] 0
```

Computing the confusion matrix

```
conf <- confusionMatrix(as.factor(y_predict_test), as.factor(y_test_data))
names(conf)
```

```
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
```

```
conf$table
```

```
##           Reference
## Prediction  0  1
##          0 46  0
##          1  0 32
```

```
conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   1.000000e+00   1.000000e+00   9.538076e-01   1.000000e+00   5.897436e-01
## AccuracyPValue  McnemarPValue
##   1.293397e-18            NaN
```

The model has predicted all of the test errors correctly

Calculating the accuracy using formula

```
false_result = which(y_predict_test==y_test_data)
accuracy=length(false_result) / length(y_test_data)
print(accuracy)
```

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

(h) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

Modelling KNN with the values k = 1,3,5,7,9,11,13,15

```
require(class)
```

```
## Loading required package: class
```

```
kvalues = c(1,3,5,7,9,11,13,15)
model_knn = c()
error_knn = c()
knn_accuracy = c()
for(i in 1:8) {
  predict_knn <-knn(train_data[,-1],test_data[,-1],y_train_data,k=kvalues[i])
  #model_knn[i] = predict_knn
  error_knn[i]<- mean(predict_knn != y_test_data)

  false_result_knn = which(predict_knn == y_test_data)
  knn_accuracy[i] = length(false_result_knn)/length(y_test_data)
}
```

Error at each k value

```
print(error_knn)
```

```
## [1] 0.02564103 0.02564103 0.02564103 0.05128205 0.03846154 0.05128205 0.05128205
## [8] 0.06410256
```

```
print(min(error_knn))
```

```
## [1] 0.02564103
```

Accuracy at each k value

```
print(knn_accuracy)
```

```
## [1] 0.9743590 0.9743590 0.9743590 0.9487179 0.9615385 0.9487179 0.9487179
## [8] 0.9358974
```

```
print(max(knn_accuracy))
```

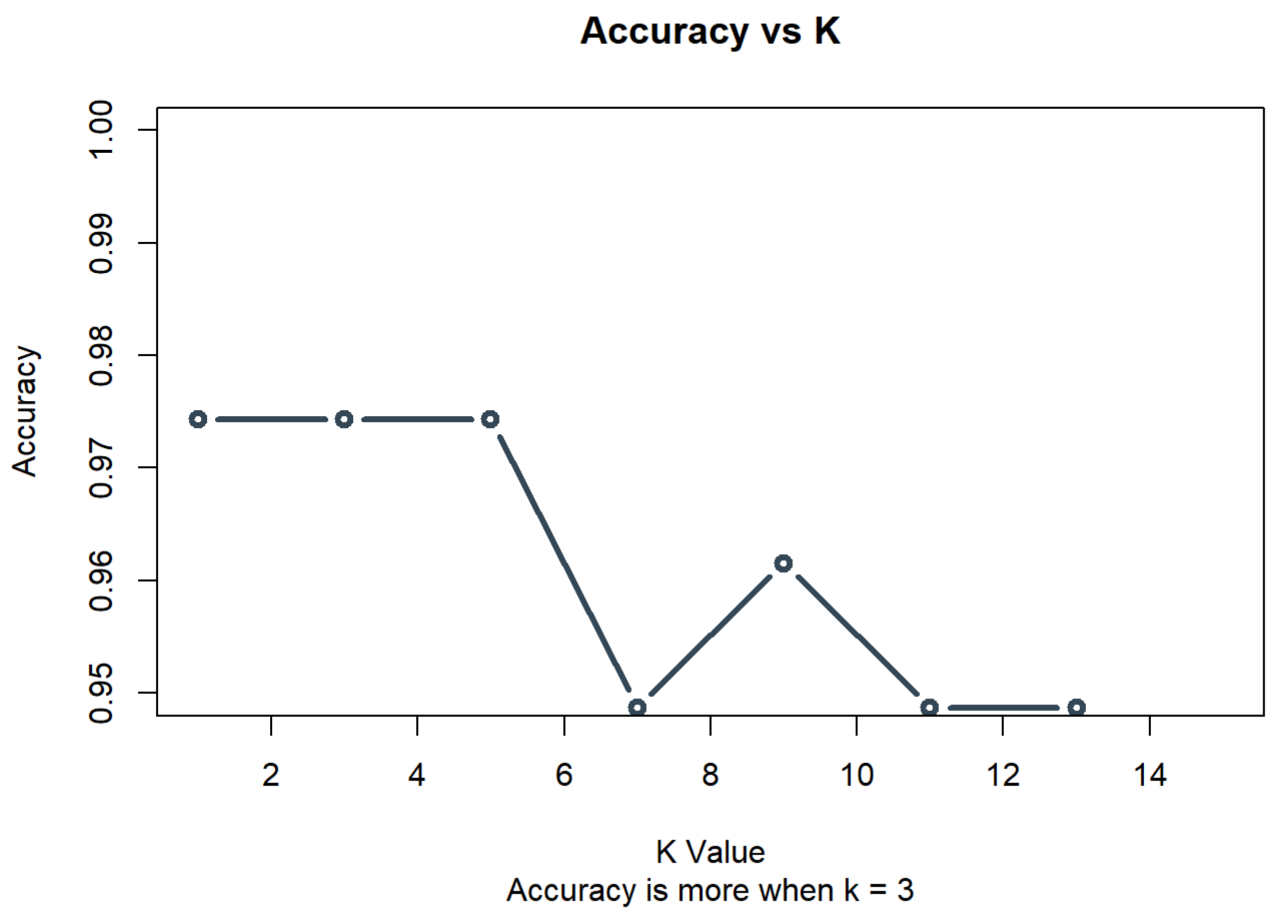
```
## [1] 0.974359
```

From the given k values k = 1,3,5,7,9,11,13,15, k =3 gives better accuracy

When K= 3 the error is 0.02564103

Plotting K and Accuracy

```
plot(kvalues,knn_accuracy, col="#334756", type = "b", xlab = "K Value", ylab = "Accuracy", yl
im = c(0.95,1.0), main = "Accuracy vs K", sub="Accuracy is more when k = 3", lwd = 3.0)
```


Accuracy vs K
Accuracy
1.00
0.99
0.98
0.97
0.96
0.95
2
4
6
8
10
12
14
K Value
Accuracy is more when k = 3