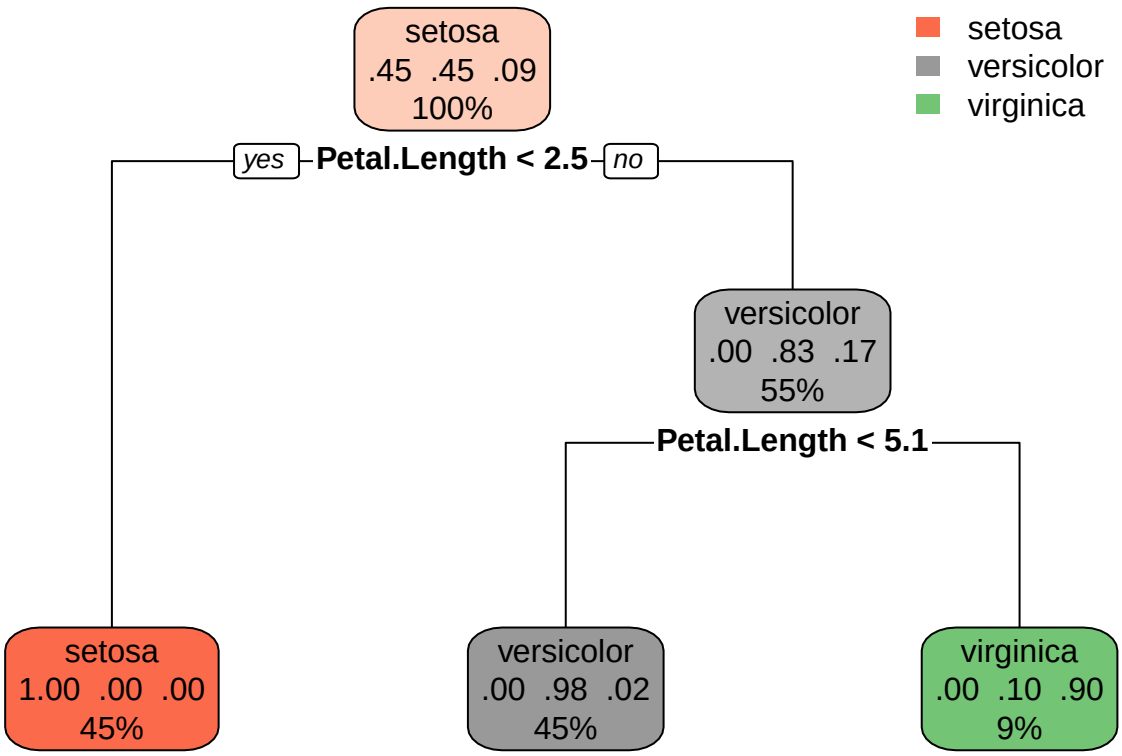
Iris Dataset

Shreya Kaul

```
# Load standard libraries
library(tidyverse)
library(datasets)
library(ggplot2)
library(MASS)
library(klaR)
library(stringr)
library(ISLR)
library(caret)
library(dummies)
library(class)
library(scales)
library(boot)
library(glmnet)
library(rpart)
library(rpart.plot)
```

```
#Dataset -Iris
data(iris)
my_x1 <- iris$Sepal.Length
my_x2 <- iris$Petal.Length
my_y <- as.factor(iris$Species)
```

```
#Classification Tress
tree_sample <- sample(1:110, replace=FALSE)
train_data <- iris[tree_sample,c('Species','Sepal.Length','Petal.Length')]
test_data <- iris[-tree_sample,c('Species','Sepal.Length','Petal.Length')]
train_tree <- rpart(train_data$Species~.,data = train_data, method = 'class')
rpart.plot(train_tree)
```




```
fit <- predict(train_tree, test_data, type='class')
```

```
table_mat <- table(test_data$Species , fit)
print(table_mat)
```

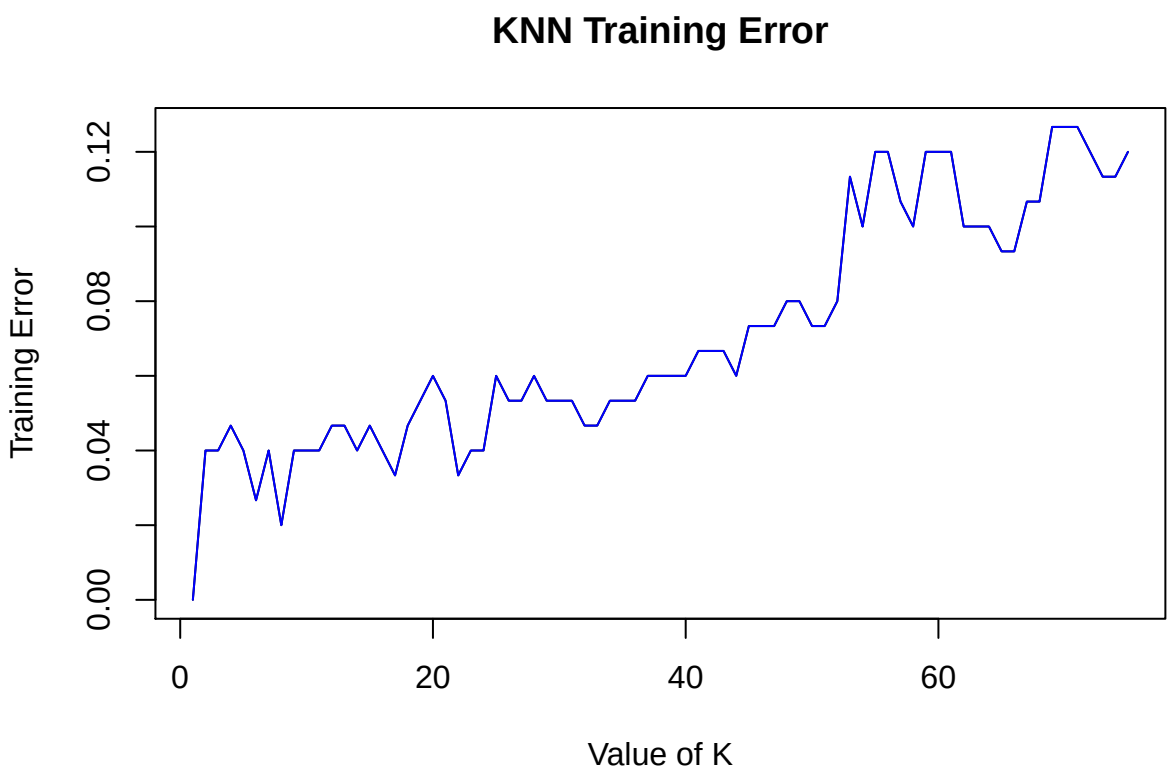
```
##             fit
##              setosa versicolor virginica
##   setosa          0          0         0
##   versicolor      0          0         0
##   virginica       0          8        32
```

```
accu <- sum(diag(table_mat)) / sum(table_mat) * 100
print(paste0("Accuracy of the model: ", accu," %"))
```

```
## [1] "Accuracy of the model: 80 %"
```

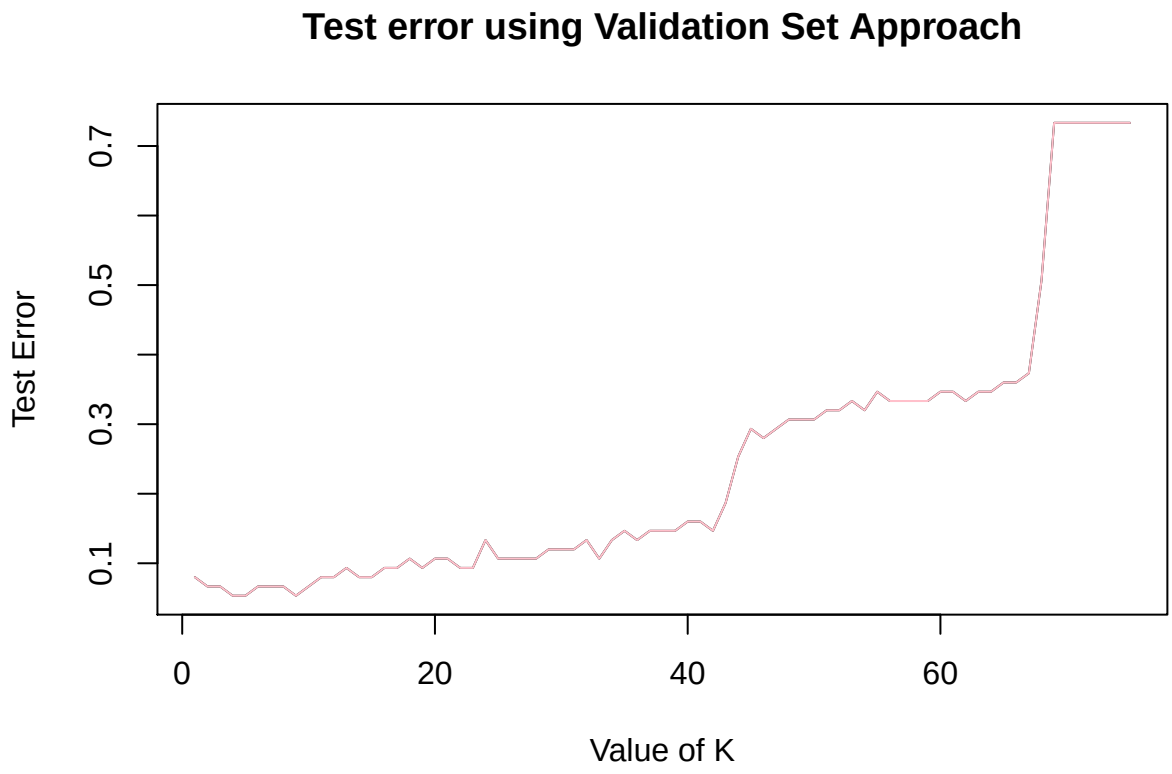
```
set.seed(123)
k_seq <- seq(1:((nrow(iris)/2)))

train_error_seq <- vector('list', length = length(k_seq))

for (i in k_seq)
{

k <- k_seq[i]

training_prediction <- knn(train=data.frame(my_x1,my_x2),test=data.frame(my_x1,my_x2),cl=my_y,k=k)

training_error <- mean(training_prediction!=my_y)

train_error_seq[i] <- training_error
```

```
}
#Plotting of KNN Training Error for different values of K
plot(x = k_seq, y=unlist(train_error_seq), type='l', xlab='Value of K', ylab='Training Error', main='KN
lines(x = k_seq, y=unlist(train_error_seq), col = "blue")
```



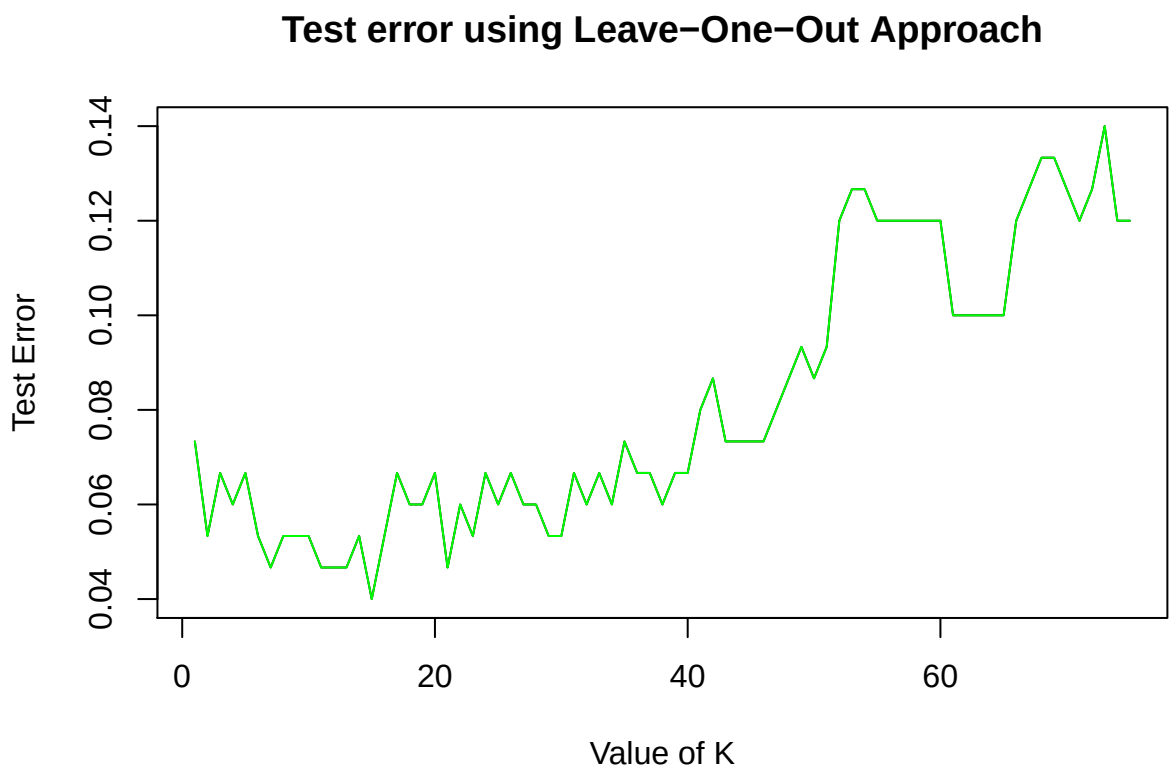

```
#Validation set approach

#Randomizing data and Splitting into test and train data
set.seed(123)
iris_random <- sample(1:nrow(iris), replace = FALSE)
iris_random_sample <- iris[iris_random,]
iris_random_train_class <- rbind(iris_random_sample[1:75,c('Species')])
iris_random_test_class <- rbind(iris_random_sample[76:150,c('Species')])
iris_random_train <- rbind(iris_random_sample[1:75,c('Sepal.Length','Petal.Length')])
iris_random_test <- rbind(iris_random_sample[76:150,c('Sepal.Length','Petal.Length')])

#Test error using Validation Set Approach
train_error_val <- vector()
k_seq <- seq(1:((nrow(iris)/2)))
for (i in k_seq)
{

k <- k_seq[i]

training_prediction_val <- knn(train=iris_random_train,test=iris_random_test,cl=iris_random_train_class

training_error_val <- mean(training_prediction_val!=iris_random_test_class)

train_error_val[i] <- training_error_val
```

```
}

#Plotting of test error using validation set approach
plot(x = k_seq, y=unlist(train_error_val), type='l', xlab='Value of K', ylab='Test Error', main='Test e
lines(x = k_seq, y=unlist(train_error_val), col="pink")
```



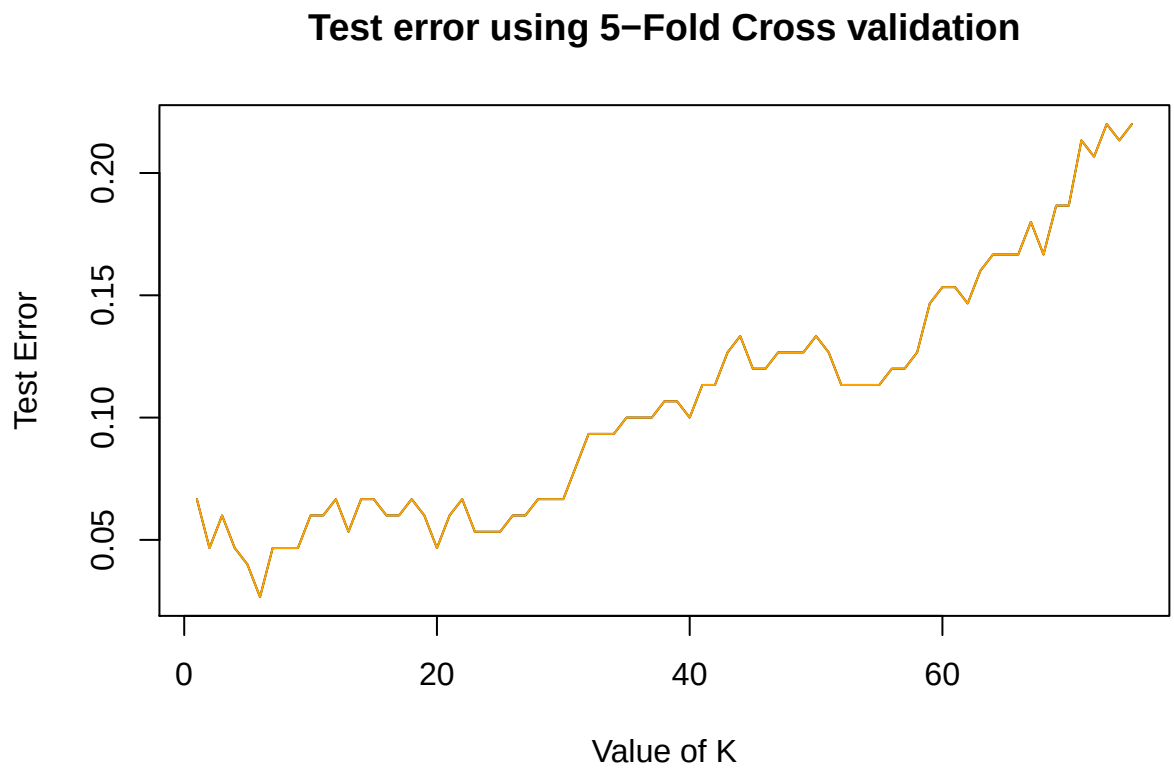

```
#Leave one out CV

train_error_loocv <- vector()
k_seq <- seq(1:((nrow(iris)/2)))
train_error <- vector()

#Loop for K-values
for (j in k_seq)
{
#Train and test dataset
for (i in 1:nrow(iris))
{
iris_loocv_train_class <- rbind(iris[-i,c('Species')])
iris_loocv_test_class <- rbind(iris[i,c('Species')])
iris_loocv_train <- rbind(iris[-i,c('Sepal.Length','Petal.Length')])
iris_loocv_test <- rbind(iris[i,c('Sepal.Length','Petal.Length')])


training_prediction_loocv <- knn(train=iris_loocv_train,test=iris_loocv_test,cl=iris_loocv_train_class,

training_error_loocv <- mean(training_prediction_loocv!=iris_loocv_test_class)

train_error_loocv[i] <- training_error_loocv
```

```
}
  train_error[j] <- mean(train_error_loocv)
}

#Plotting of test error using Leave one out approach
plot(x = k_seq, y=unlist(train_error), type='l', xlab='Value of K', ylab='Test Error', main='Test error
lines(x = k_seq, y=unlist(train_error), col="green")
```



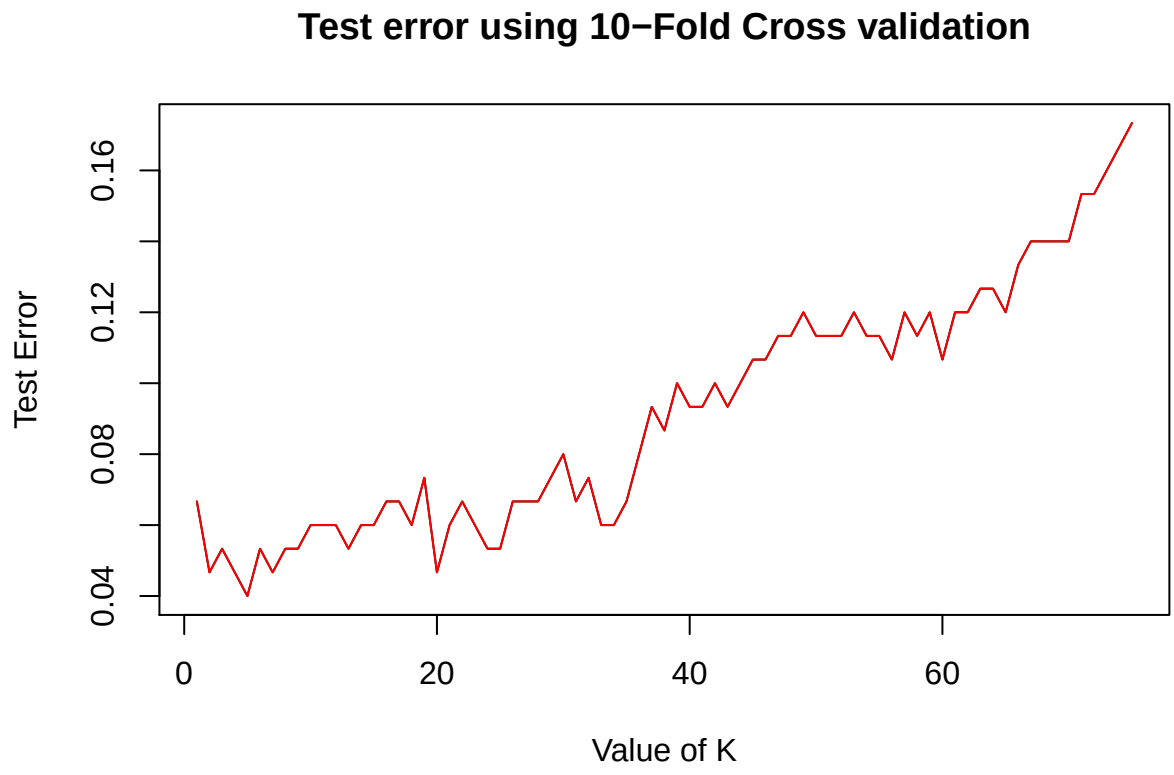

```
#5-Fold CV
training_error_kfold <- vector()
train_error_kfold <- vector()
kfold_error <- vector()
k_seq <- seq(1:((nrow(iris)/2)))

#Shuffling data randomly
set.seed(123)
iris_random_kfold <- sample(1:nrow(iris), replace = FALSE)
iris_sample_kfold <- iris[iris_random_kfold,]

#Creating 5 folds
folds <- cut(seq(1,nrow(iris_sample_kfold)),breaks=5,labels=FALSE)


for (j in k_seq) {


#Perform CV
for (i in 1:5) {
  test_index <- which(folds==i,arr.ind = TRUE)
```

```
  kfold_test_data <- iris_sample_kfold[test_index,c('Sepal.Length','Petal.Length')]
  kfold_train_data <- iris_sample_kfold[-test_index,c('Sepal.Length','Petal.Length')]
  kfold_test_class <- iris_sample_kfold[test_index,c('Species')]
  kfold_train_class <- iris_sample_kfold[-test_index,c('Species')]

  training_prediction_kfold <- knn(train=kfold_train_data,test=kfold_test_data,cl=kfold_train_class,k=j
  training_error_kfold <- mean(training_prediction_kfold!=kfold_test_class)

train_error_kfold[i] <- training_error_kfold

}
  kfold_error[j] <- mean(train_error_kfold)
}
#Plotting of test error using 5-fold cross validation for different values of K
plot(x = k_seq, y=unlist(kfold_error), type='l', xlab='Value of K', ylab='Test Error', main='Test error
lines(x = k_seq, y=unlist(kfold_error), col="orange")
```



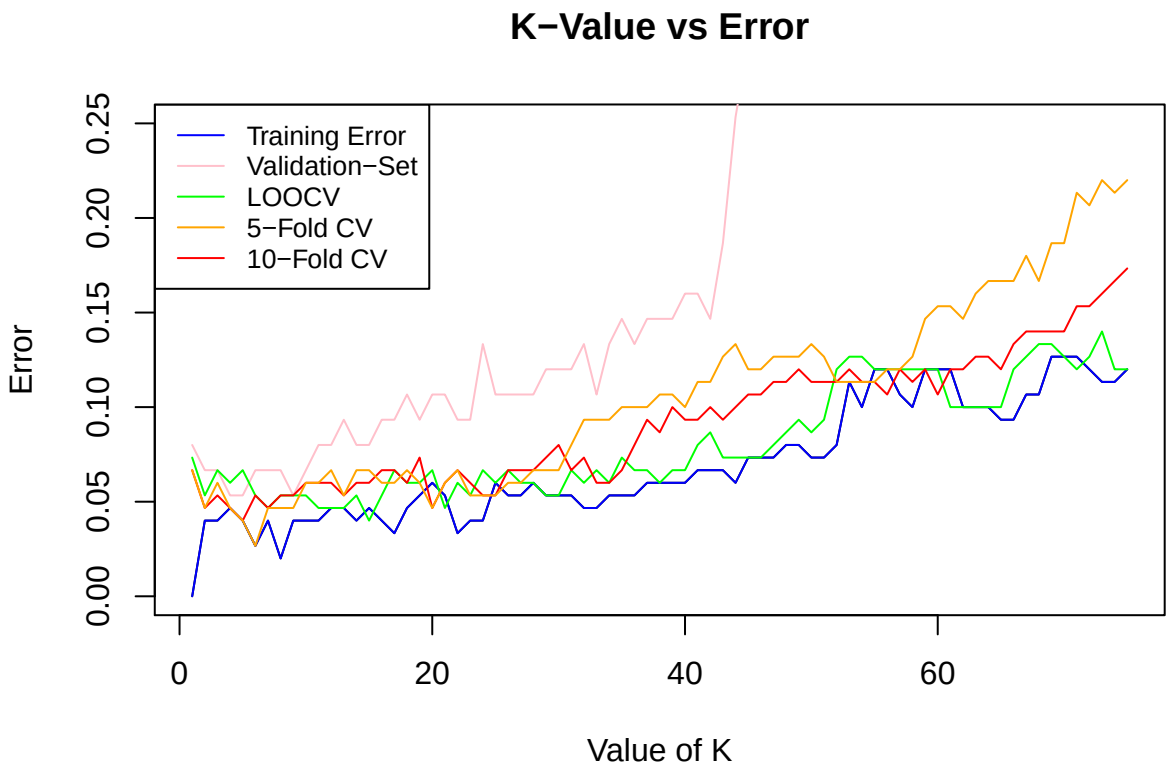

```
#10-Fold CV
training_error_kfold <- vector()
train_error_kfold <- vector()
kfold_error2 <- vector()
k_seq <- seq(1:((nrow(iris)/2)))

#Shuffling data randomly
set.seed(123)
iris_random_kfold <- sample(1:nrow(iris), replace = FALSE)
iris_sample_kfold <- iris[iris_random_kfold,]

#Creating  folds
```

```r
folds <- cut(seq(1,nrow(iris_sample_kfold)),breaks=10,labels=FALSE)

for (j in seq_along(k_seq)) {


#Perform CV
for (i in 1:10) {
  test_index <- which(folds==i,arr.ind = TRUE)
  kfold_test_data <- iris_sample_kfold[test_index,c('Sepal.Length','Petal.Length')]
  kfold_train_data <- iris_sample_kfold[-test_index,c('Sepal.Length','Petal.Length')]
  kfold_test_class <- iris_sample_kfold[test_index,c('Species')]
  kfold_train_class <- iris_sample_kfold[-test_index,c('Species')]

  training_prediction_kfold <- knn(train=kfold_train_data,test=kfold_test_data,cl=kfold_train_class,k=j
  training_error_kfold <- mean(training_prediction_kfold!=kfold_test_class)

train_error_kfold[i] <- training_error_kfold

}
  kfold_error2[j] <- mean(train_error_kfold)
}
#Plotting of test error using 10-fold cross validation for different values of K
plot(x = k_seq, y=unlist(kfold_error2), type='l', xlab='Value of K', ylab='Test Error', main='Test erro
lines(x = k_seq, y=unlist(kfold_error2), col="red")
```




```r
#Combining all the plots together
plot(x = k_seq, y=unlist(train_error_seq), type='l', xlab='Value of K', ylab='Error', main='K-Value vs
lines(x = k_seq, y=unlist(train_error_seq), col = "blue")
lines(x = k_seq, y=unlist(train_error_val), col="pink")
```

```
lines(x = k_seq, y=unlist(train_error), col="green")
lines(x = k_seq, y=unlist(kfold_error2), col="red")
lines(x = k_seq, y=unlist(kfold_error), col="orange")
legend("topleft",legend=c("Training Error", "Validation-Set","LOOCV","5-Fold CV","10-Fold CV"),
       col=c("blue", "pink","green","orange","red"), lty=1:1, cex=0.8)
```



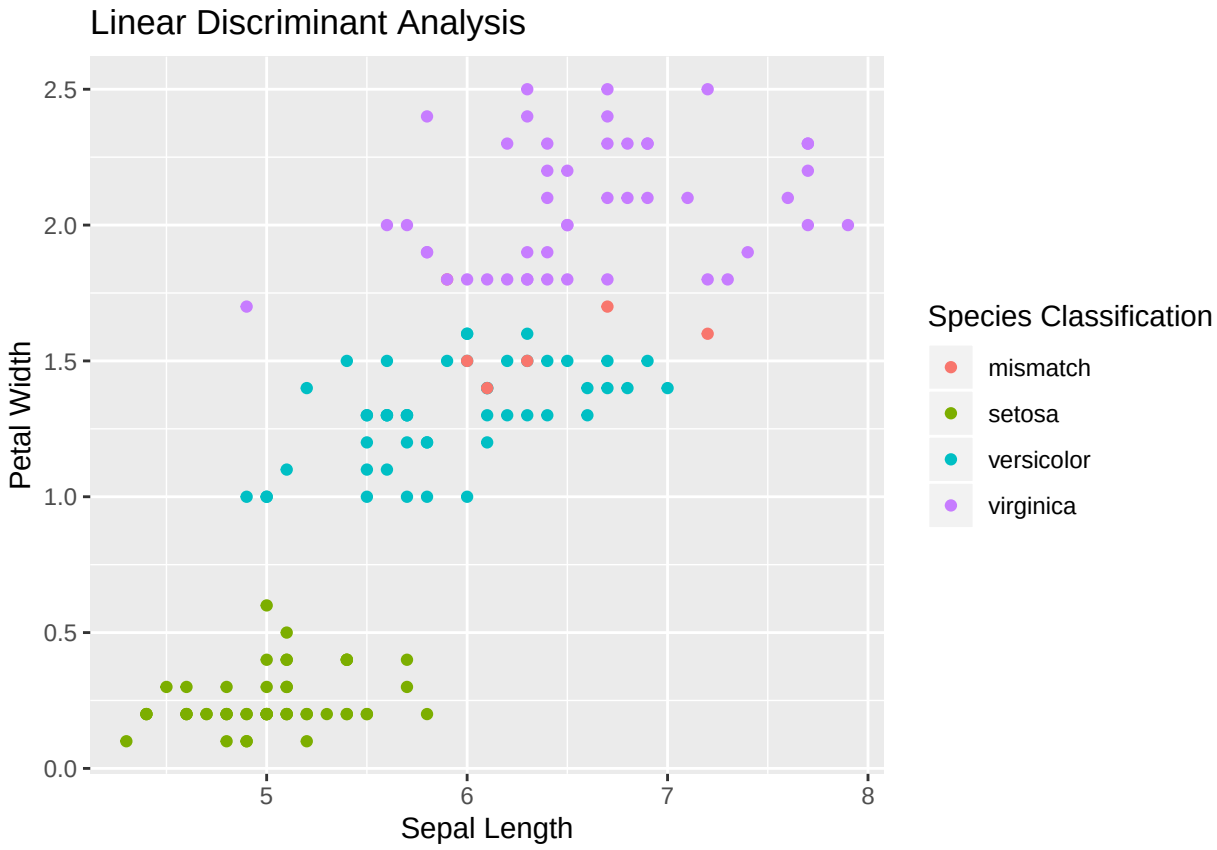
From the above plots, it can be seen that lowest test error will be for the values of K between 0-20.

```
#LDA

iris_df <- rbind(iris[,c('Sepal.Length', 'Petal.Width','Species')])
lda_iris <- lda(iris_df$Species~., iris_df)
lda_iris
```

```
## Call:
## lda(iris_df$Species ~ ., data = iris_df)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Petal.Width
## setosa            5.006       0.246
## versicolor        5.936       1.326
## virginica         6.588       2.026
##
## Coefficients of linear discriminants:
##                      LD1       LD2
## Sepal.Length -0.02715312  2.085884
```

```
## Petal.Width   4.91087360 -1.848948
##
## Proportion of trace:
##    LD1    LD2
## 0.9999 0.0001
```

```
iris_lda_value <- predict(lda_iris)
iris_df_plot_lda <- iris_df
for (i in 1:150) {
    iris_df_plot_lda$classification <- ifelse(iris_df$Species != iris_lda_value$class,"mismatch",
                                ifelse
                                (iris_df$Species == "setosa","setosa",
                                  ifelse
                                  (iris_df$Species == "versicolor","versicolor",
                                    ifelse
                                    (iris_df$Species == "virginica","virginica",NA))))

}

gg <- ggplot(iris_df_plot_lda, aes(iris_df_plot_lda$Sepal.Length, iris_df_plot_lda$Petal.Width, col=iris
print(gg)
```




```
confusionMatrix(iris_lda_value$class, iris_df$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
```

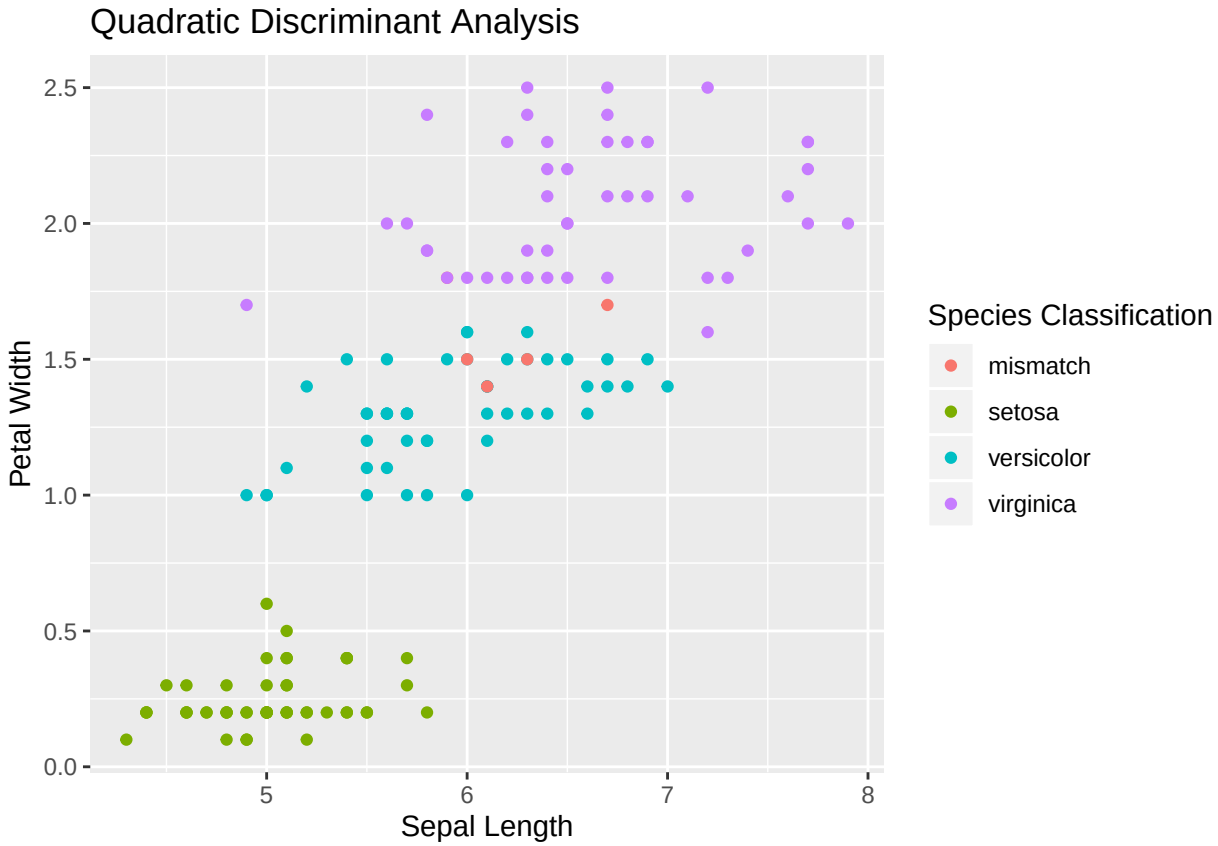
```
##   setosa         50          0         0
##   versicolor      0         48         4
##   virginica       0          2        46
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9600           0.9200
## Specificity                 1.0000            0.9600           0.9800
## Pos Pred Value              1.0000            0.9231           0.9583
## Neg Pred Value              1.0000            0.9796           0.9608
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3200           0.3067
## Detection Prevalence        0.3333            0.3467           0.3200
## Balanced Accuracy           1.0000            0.9600           0.9500
```

```
iris_df <- rbind(iris[,c('Sepal.Length', 'Petal.Width','Species')])
qda_iris <- qda(iris_df$Species~., iris_df)
qda_iris
```

```
## Call:
## qda(iris_df$Species ~ ., data = iris_df)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Petal.Width
## setosa            5.006       0.246
## versicolor        5.936       1.326
## virginica         6.588       2.026
```

```
iris_qda_value <- predict(qda_iris)
for (i in 1:150) {
    iris_qda_value$classification <- ifelse(iris_df$Species != iris_qda_value$class,"mismatch",
                                ifelse
                                (iris_df$Species == "setosa","setosa",
                                  ifelse
                                  (iris_df$Species == "versicolor","versicolor",
                                    ifelse
                                    (iris_df$Species == "virginica","virginica",NA))))

}
```

```
iris_df_plot_qda <- iris_df
iris_df_plot_qda$classification <- iris_qda_value$classification
gg <- ggplot(iris_df, aes(iris_df_plot_qda$Sepal.Length, iris_df_plot_qda$Petal.Width, col=iris_df_plot
print(gg)
```




```
confusionMatrix(iris_df$Species, iris_qda_value$class)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          3        47
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.9239, 0.9891)
##     No Information Rate : 0.34
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9412           0.9592
## Specificity                 1.0000            0.9798           0.9703
## Pos Pred Value              1.0000            0.9600           0.9400
## Neg Pred Value              1.0000            0.9700           0.9800
## Prevalence                  0.3333            0.3400           0.3267
## Detection Rate              0.3333            0.3200           0.3133
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9605           0.9647
```

Out of the LDA and QDA models, QDA gave the smaller training error. QDA has more variance and therefore is more flexible. So, it fits better on the training data. If we consider bias-variance trade-off, then as the variance increases the training error decreases.
LDA is less flexible than QDA as in LDA the number of parameters are less. Due to more parameters, QDA has high variance. If the number of observations in training data are more then LDA will give a smaller test error as QDA will have higher variance and may result in overfitting of model. But if the classes have a different covariance matrix, then LDA will face high bias on the dataset and QDA will have lower test error.