Implementation of Random Forest using Iris Dataset

```
 iris<-read.csv("D:/R programming/iris_dataset.csv")

# Loading data
 data(iris)
 head(iris)
 tail(iris)

# Structure
str(iris)

 # Installing package
 install.packages(“caTools”)
 library(“caTools”)
 install.packages(“randomForest”)
 library(“randomForest”)
 install.packages(“caret”)
 library("caret")
 # Splitting data in train and test data
 split <- sample.split(iris, SplitRatio = 0.7)
 split

 train <- subset(iris, split == "TRUE")
 test <- subset(iris, split == "FALSE")
# Fitting Random Forest to the train dataset
 control <- trainControl(method="repeatedcv", number=10, repeats=3)
 seed <- 7
 metric <- "Accuracy"
 set.seed(seed)
 rf <- train(Species~., data=iris, method="rf", metric=metric, tuneLength=15,
trControl=control)
 print(rf)
```

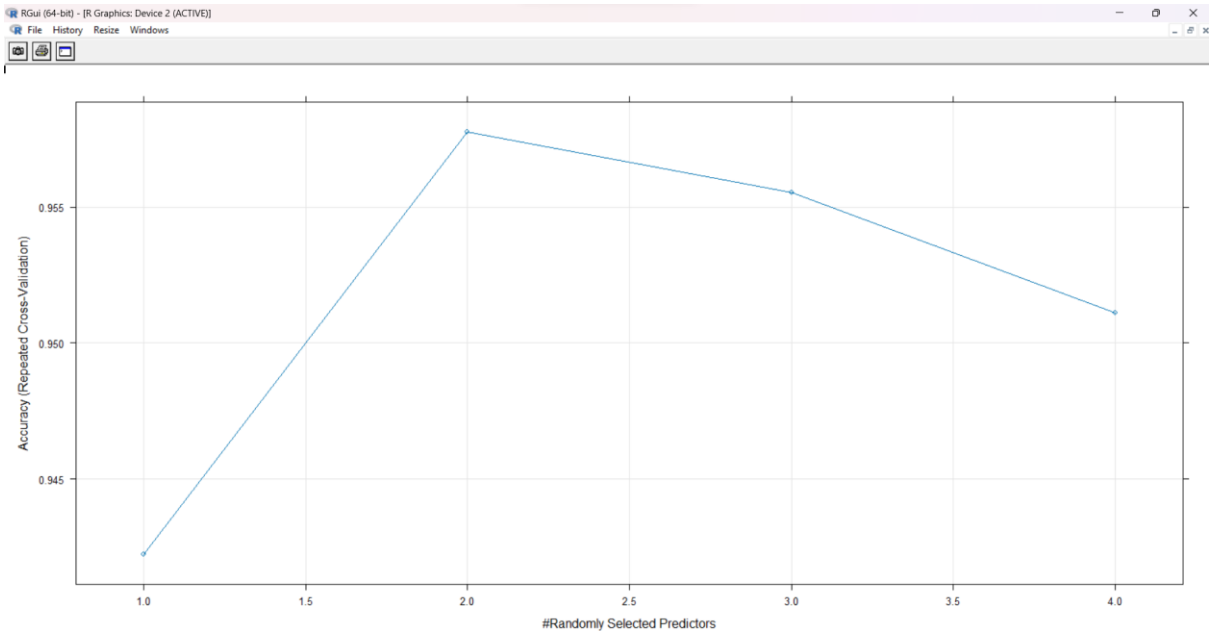
```
Random Forest

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9555556  0.9333333
  3     0.9577778  0.9366667
  4     0.9600000  0.9400000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 4.
```

Grid Search

```
tunegrid <- expand.grid(.mtry=c(1:4))
rf_gridsearch <- train(Species~., data=iris, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)
print(rf_gridsearch)
```

```
Random Forest

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.9422222  0.9133333
  2     0.9577778  0.9366667
  3     0.9555556  0.9333333
  4     0.9511111  0.9266667

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

```
plot(rf_gridsearch)
```


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
0.955
0.950
0.945
Accuracy (Repeated Cross-Validation)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
#Randomly Selected Predictors

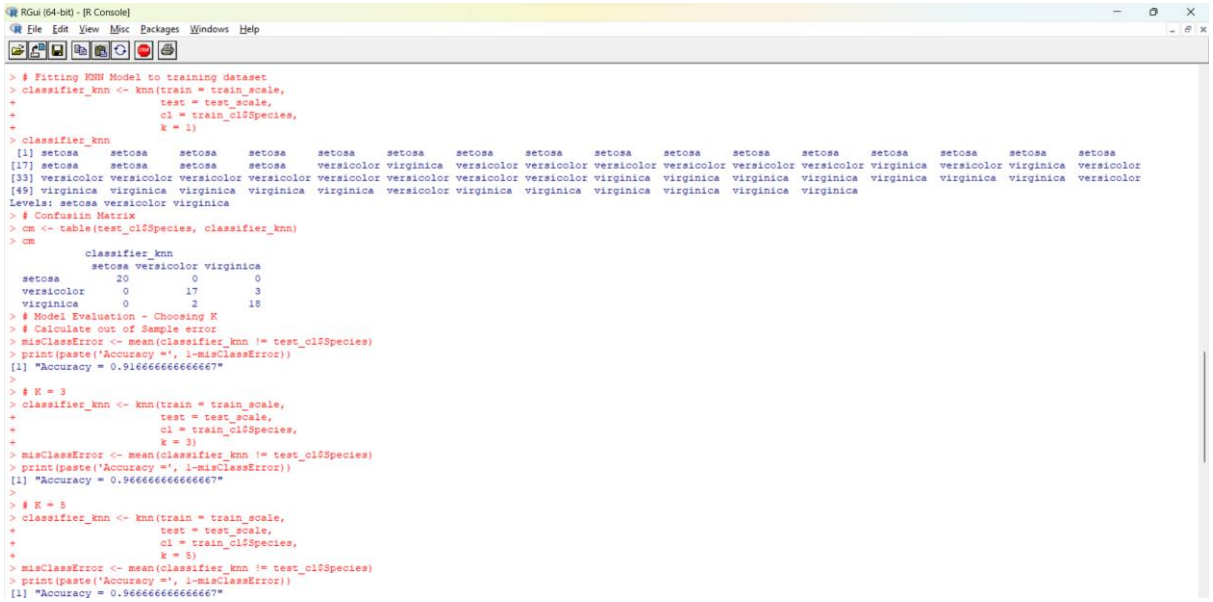
Implementation of KNN using Iris Dataset

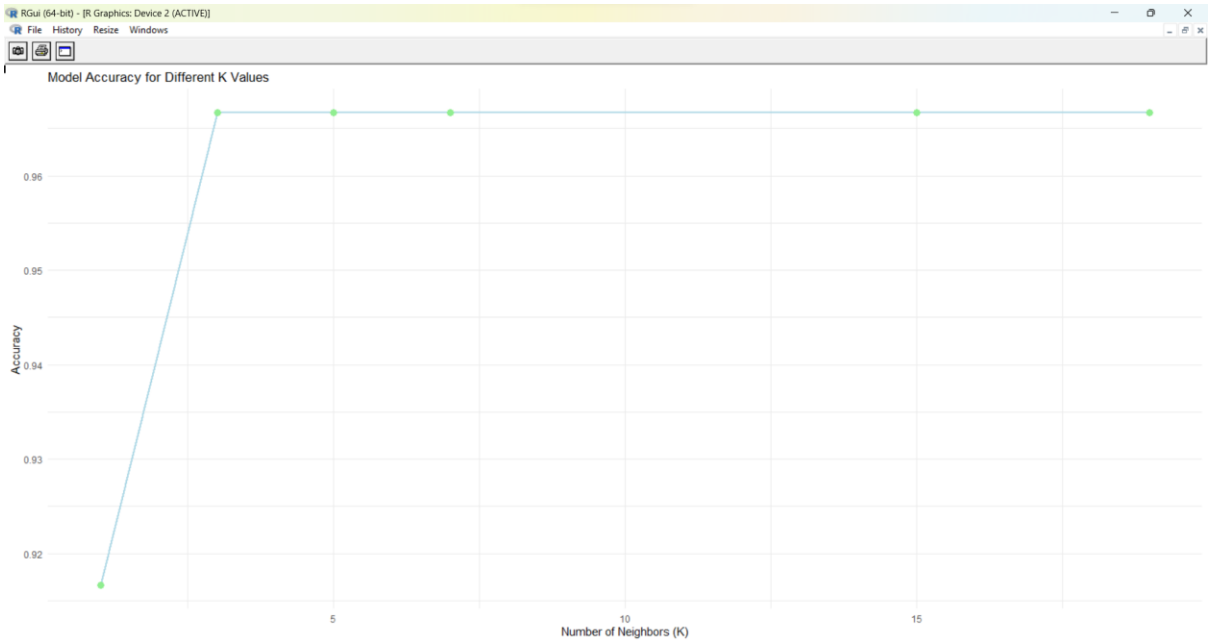
```
iris<-read.csv("D:/R programming/iris_dataset.csv")

# Loading data
data(iris)
 head(iris)
 tail(iris)

# Structure
str(iris)


# Installing Packages
install.packages("e1071")
install.packages("caTools")
install.packages("class")

# Loading package
library(e1071)
library(caTools)
library(class)

# Splitting data into train and test data
split <- sample.split(iris, SplitRatio = 0.7)
train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_cl[, 1:4])
test_scale <- scale(test_cl[, 1:4])

head(train_scale)
head(test_scale)


# Fitting KNN Model to training dataset
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 1)
classifier_knn


# Confusion Matrix
cm <- table(test_cl$Species, classifier_knn)
cm
```

```
# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 3
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 3)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 5
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 5)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 7
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 7)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 15
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 15)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 19
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 19)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))


library(ggplot2)

# Data preparation
k_values <- c(1, 3, 5, 7, 15, 19)
```

```
# Calculate accuracy for each k value
accuracy_values <- sapply(k_values, function(k) {
  classifier_knn <- knn(train = train_scale,
                        test = test_scale,
                        cl = train_cl$Species,
                        k = k)
  1 - mean(classifier_knn != test_cl$Species)
})

# Create a data frame for plotting
accuracy_data <- data.frame(K = k_values, Accuracy = accuracy_values)

# Plotting
ggplot(accuracy_data, aes(x = K, y = Accuracy)) +
  geom_line(color = "lightblue", size = 1) +
  geom_point(color = "lightgreen", size = 3) +
  labs(title = "Model Accuracy for Different K Values",
       x = "Number of Neighbors (K)",
       y = "Accuracy") +
  theme_minimal()
```

OUTPUT




RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Model Accuracy for Different K Values
Accuracy
0.96
0.95
0.94
0.93
0.92
5
10
15
Number of Neighbors (K)

```
install.packages('dplyr')
library(dplyr)
install.packages('caret')
library(caret)
install.packages('e1071')
library("e1071")
install.packages('ggplot2')
library("ggplot2")
install.packages('rpart.plot')
library(rpart.plot)
titanic<-read.csv("C:/Users/Arul Kumaran P/Desktop/titanic_data.csv")
head(titanic)
tail(titanic)
titanic = select(titanic,survived,pclass,sex,sibsp,parch)
titanic=na.omit(titanic)
str(titanic)


titanic$survived = factor(titanic$survived)
titanic$pclass = factor(titanic$pclass, order=TRUE,levels = c(3, 2, 1))
ggplot(titanic,aes(x = survived))+  geom_bar(width=0.5, fill = "coral") +
geom_text(stat='count', aes(label=stat(count)), vjust=-0.5) + theme_classic()
train_test_split = function(data, fraction = 0.8, train = TRUE)
  {
      total_rows = nrow(data)
      train_rows = fraction * total_rows
```

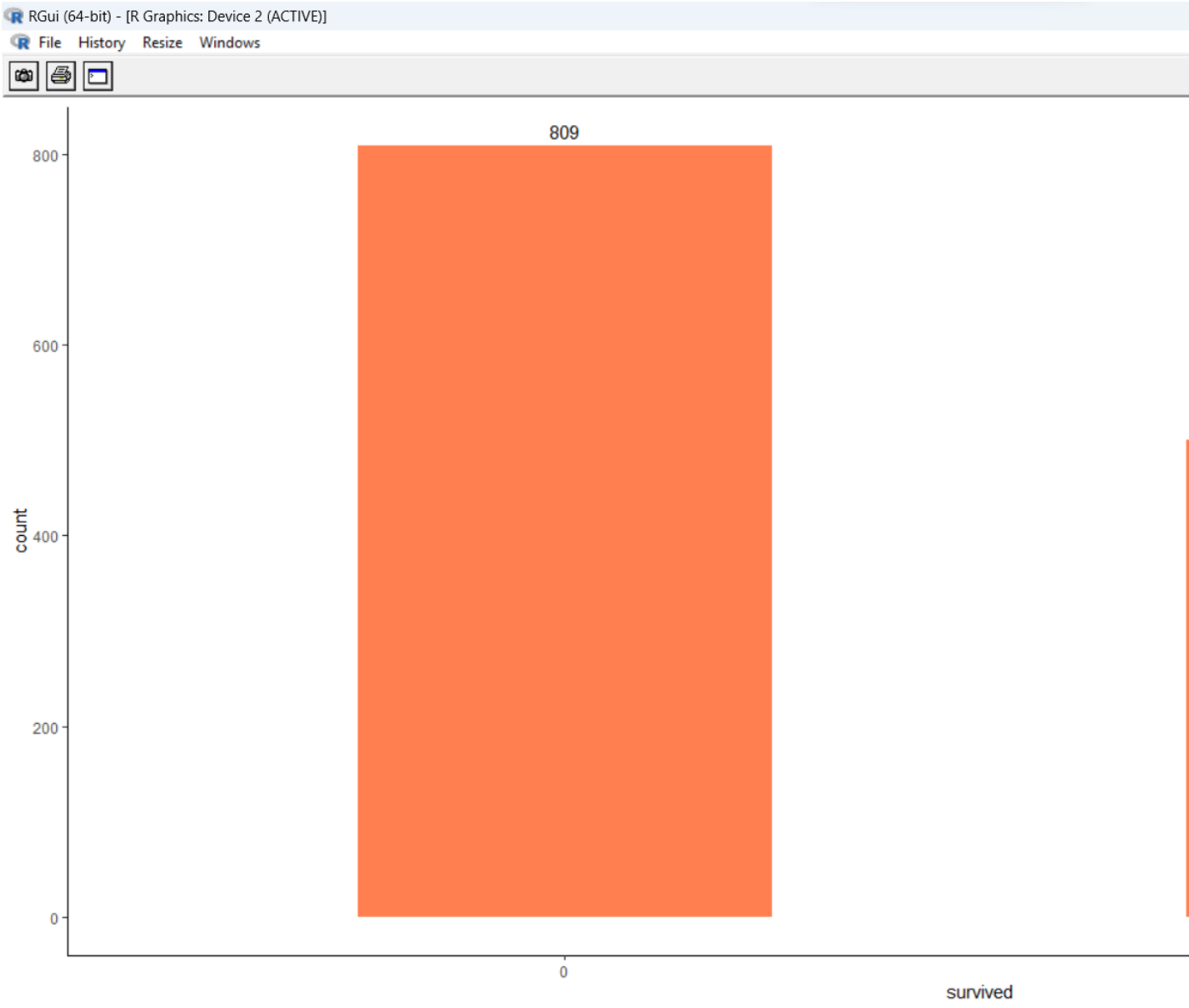
```
        sample = 1:train_rows
        if (train == TRUE){
          return (data[sample, ])
          }else{
            return (data[-sample, ])
          }
  }
train <- train_test_split(titanic, 0.8, train = TRUE)
test <- train_test_split(titanic, 0.8, train = FALSE)
nb_model = naiveBayes(survived ~., data=train)
nb_predict = predict(nb_model,test)
table_mat = table(nb_predict, test$survived)
table_mat

nb_accuracy = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", nb_accuracy)
```

```
install.packages('dplyr')
 library(dplyr)
 install.packages('caret')
 library(caret)
 install.packages('ggplot2')
 library(ggplot2)
 install.packages('rpart.plot')
 library(rpart.plot)
 titanic<-read.csv("C:/Users/Arul Kumaran P/Desktop/titanic_data.csv")
 head(titanic)
 tail(titanic)
titanic = select(titanic,survived,pclass,sex,sibsp,parch)
 titanic=na.omit(titanic)
 str(titanic)
 titanic$survived = factor(titanic$survived)
 titanic$pclass = factor(titanic$pclass, order=TRUE,   levels = c(3, 2, 1))

 ggplot(titanic,aes(x = survived))+
  geom_bar(width=0.5, fill = "coral") +
  geom_text(stat='count', aes(label=stat(count)), vjust=-0.5) +
  theme_classic()

 train_test_split = function(data, fraction = 0.8, train = TRUE) {
  total_rows = nrow(data)
  train_rows = fraction * total_rows
  sample = 1:train_rows
```

```
if (train == TRUE) {
   return (data[sample, ])
  } else {
   return (data[-sample, ])
  }
}
 train <- train_test_split(titanic, 0.8, train = TRUE)
 test <- train_test_split(titanic, 0.8, train = FALSE)
 fit <- rpart(survived~., data = train, method ='class')
 rpart.plot(fit, extra = 106)
 View(titanic)
 predicted = predict(fit, test, type = 'class')
 table = table(test$survived, predicted)
 table
 accuracy_Test <- sum(diag(table)) / sum(table)
 print(paste('Accuracy for test', accuracy_Test))
 accuracy_tune <- function(fit) {
  +    predict_unseen <- predict(fit, data_test, type = 'class')
  +    table_mat <- table(data_test$survived, predict_unseen)
  +    accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
  +    accuracy_Test
 }
 control <- rpart.control(minsplit = 4,
                          minbucket = round(5 / 3),
                          maxdepth = 3,
                          cp = 0)
```

```
tune_fit <- rpart(survived~., data = train, method = 'class', control = control)
dt_predict = predict(tune_fit, test, type = 'class')
table_mat = table(test$survived, dt_predict)
dt_accuracy_2 = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", dt_accuracy_2)
```

```
[1] "The accuracy is :  0.793893129770992"
```


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
809
800
600
400
200
0
count
0
survived