Project 1

Mudit Arora

2023-11-02

Necessary libraries

```
library(ggplot2)
library(caTools)
library(class)
library(caret)
```

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

```
library(ROCR)
```

Problem Statement

The diabetes dataset has been taken from Kaggle which consists of various factors that causes Diabetes. I'll be using KNN Classification to understand which of the factors are more accurate.

K-Nearest Neighbors (KNN) is a simple and commonly used machine learning algorithm for classification and regression tasks. It is a type of instance-based or lazy learning algorithm, meaning it doesn't explicitly build a model during the training phase but stores the entire training dataset in memory for making predictions.

```
dataset <-  read.csv("diabetes.csv")
summary(dataset)
```

```
##   Pregnancies        Glucose      BloodPressure   SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.0   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.0   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.0   Median :23.00
##  Mean   : 3.849   Mean   :120.9   Mean   : 69.1   Mean   :20.52
##  3rd Qu.: 6.000   3rd Qu.:140.5   3rd Qu.: 80.0   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.0   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2435           1st Qu.:24.00
##  Median : 32.0   Median :32.00   Median :0.3740           Median :29.00
##  Mean   : 79.9   Mean   :31.99   Mean   :0.4721           Mean   :33.25
##  3rd Qu.:127.5   3rd Qu.:36.60   3rd Qu.:0.6265           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
```

```
##     Outcome
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3494
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

Dataset

The dataset consists of 9 columns, 1-8 columns are the factors, and Outcome being the result.

```
head(dataset)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

Data Preprocessing

Checking if the dataset consist of any null values, fortunately it does not so it will we can start with the prediction model.

```
null_values <- sum(is.na(dataset))
null_values
```

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

Visualizing various graphs

After comparing several factors with each other, these are the 5 important graph to depict the outcome of Diabetes.

1. Scatter plot of Glucose vs. BMI with Outcome

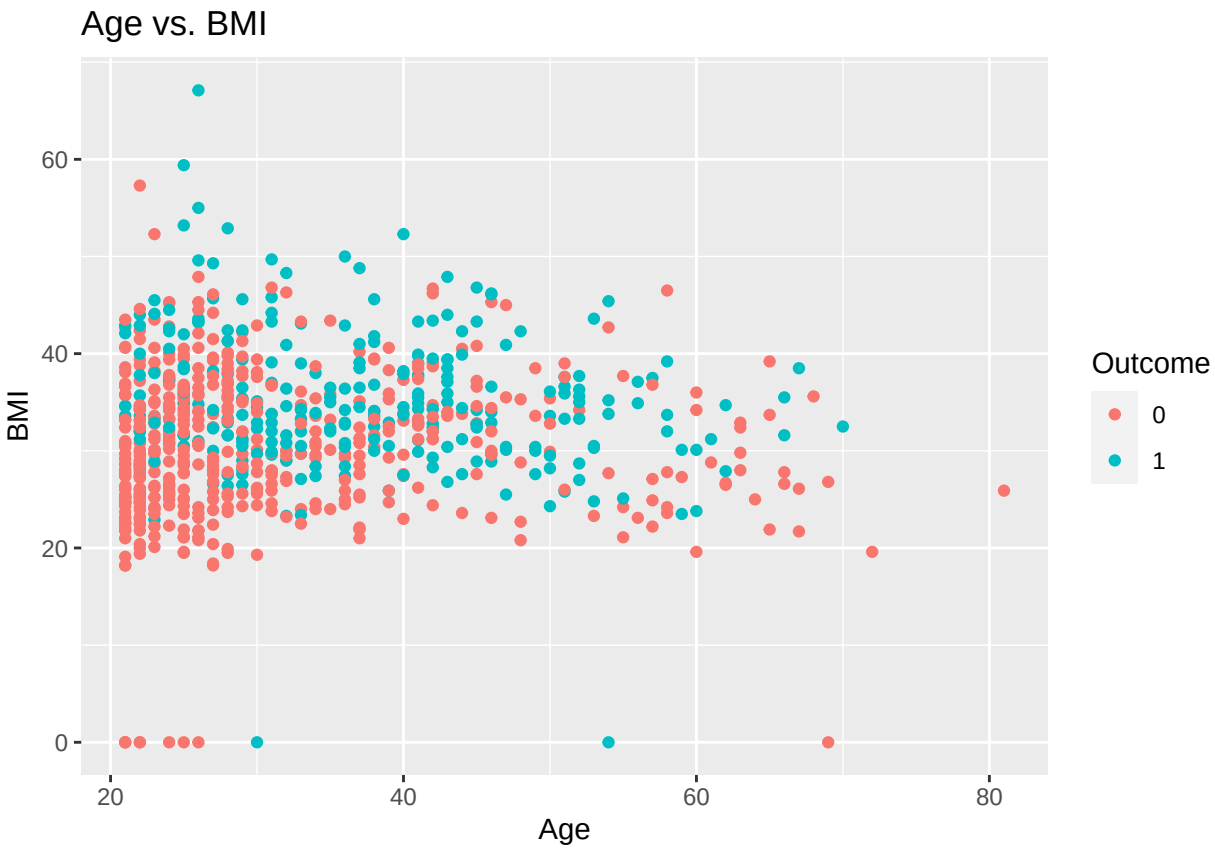
This plot shows that when glucose content is > 100 there is a high chance of getting diabetes.

```
ggplot(dataset, aes(x = BMI, y = Glucose, color = factor(Outcome))) +
  geom_point() +
  labs(title = "BMI vs. Glucose", color = "Outcome")
```

2. Scatter plot of Age vs BMI with Outcome

This plot shows that it's unlikely to get diabetes if your age is < 30, whereas there is high chance of getting diabetes > 30 with BMI ranging 25-45.

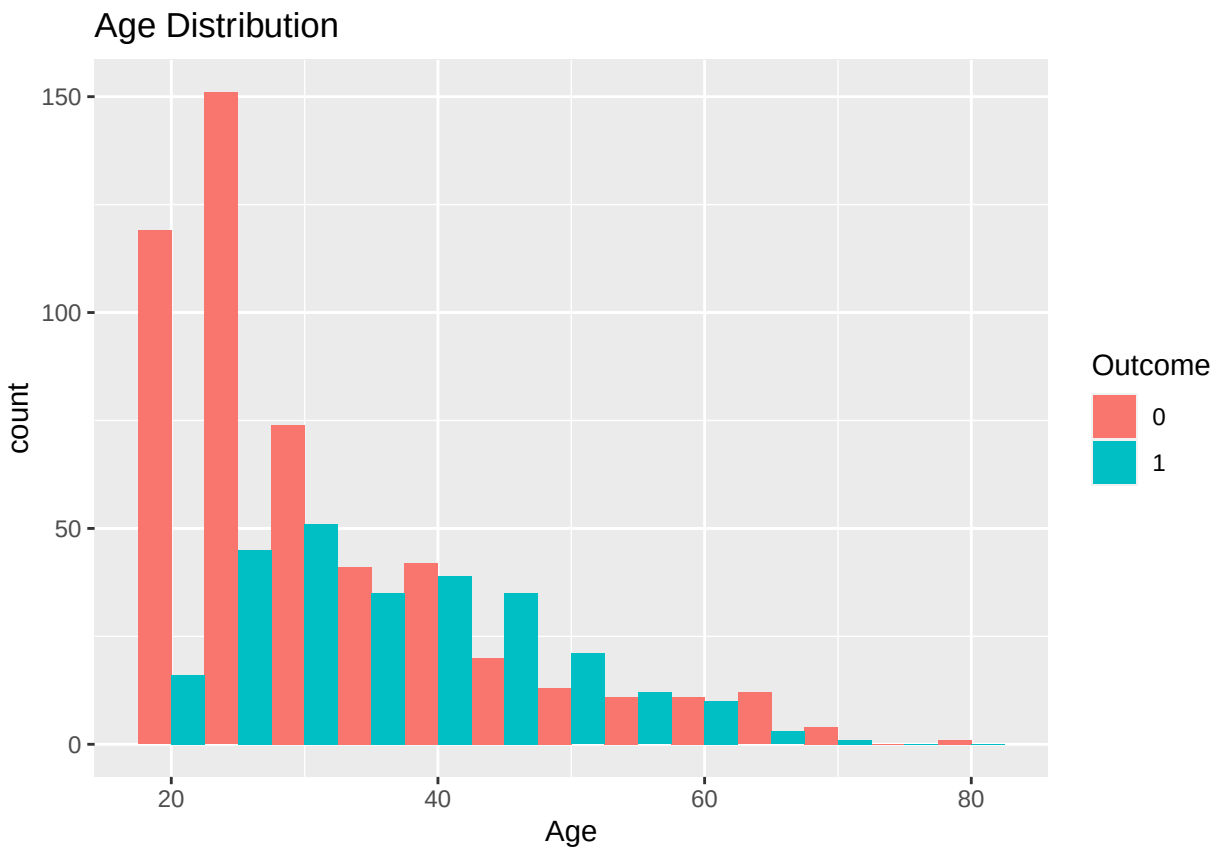
```
ggplot(dataset, aes(x = Age, y = BMI, color = factor(Outcome))) +
  geom_point() +
  labs(title = "Age vs. BMI", color = "Outcome")
```



3. Histogram of Age Distribution

The histogram suggests that people between the age of 30-50 have high risk of getting diabetes.

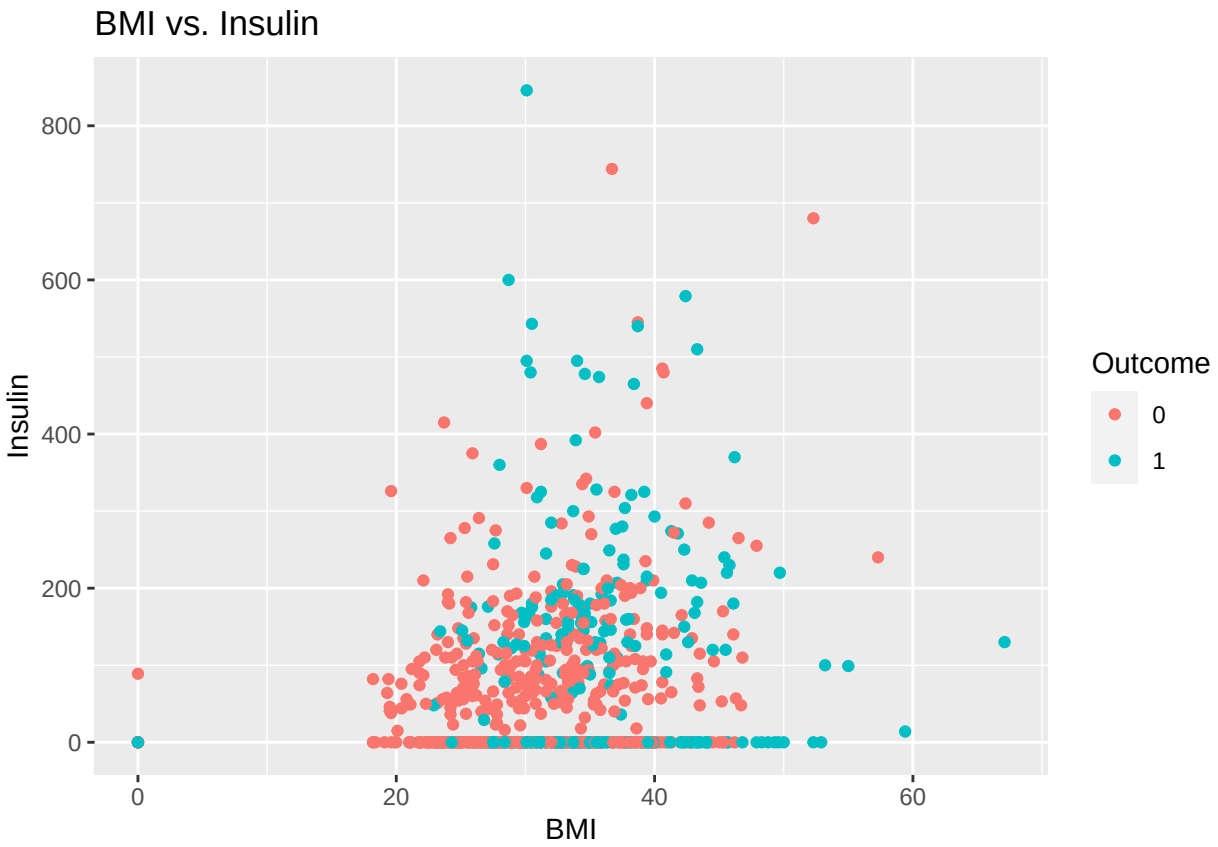
```
ggplot(dataset, aes(x = Age, fill = factor(Outcome))) +
  geom_histogram(binwidth = 5, position = "dodge") +
  labs(title = "Age Distribution", fill = "Outcome")
```



4 Scatter Plot of BMI vs Insulin

This plot suggests that there is high risk level of getting diabetes on high insulin level.

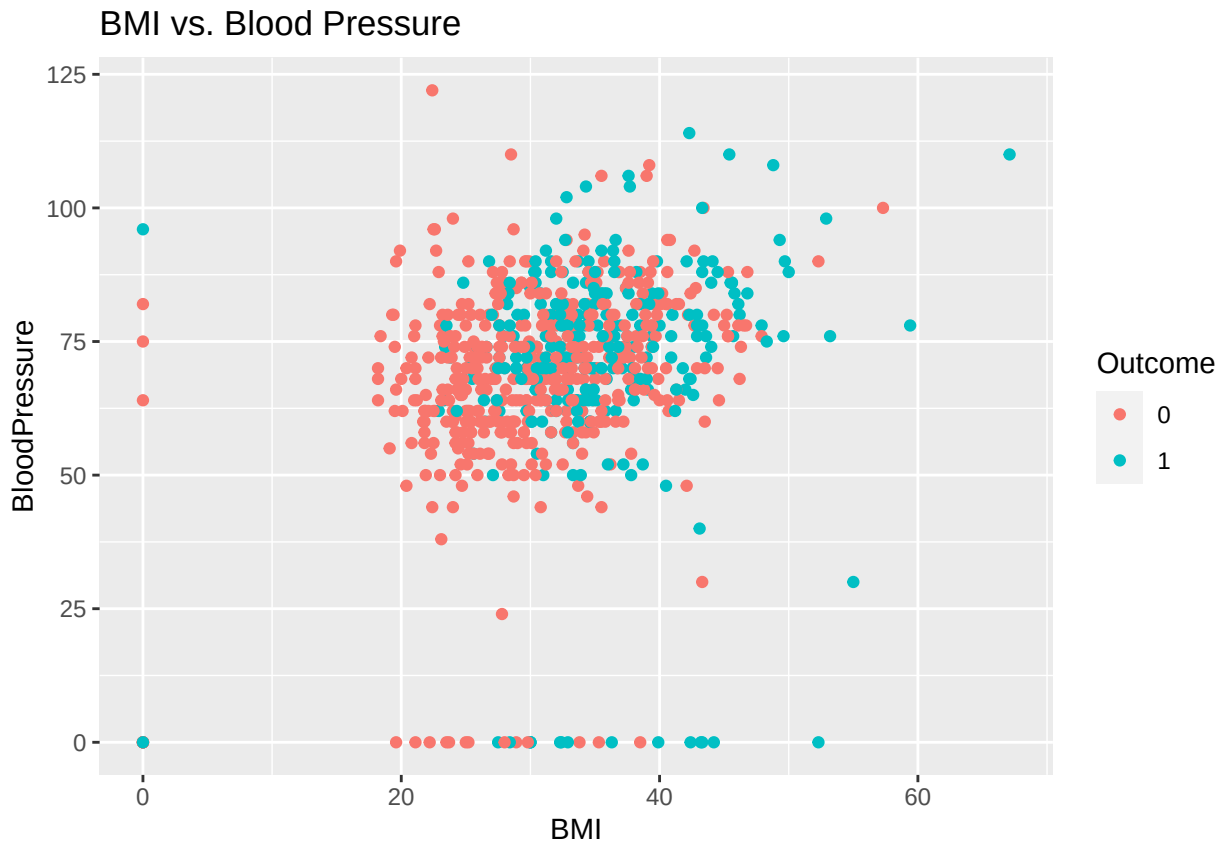
```
ggplot(dataset, aes(x = BMI, y = Insulin, color = factor(Outcome))) +
  geom_point() +
  labs(title = "BMI vs. Insulin", color = "Outcome")
```



5 Scatter plot of BMI vs Blood Pressure

High Blood Pressure is significant factor of having Diabetes.

```
ggplot(dataset, aes(x = BMI, y = BloodPressure, color = factor(Outcome))) +
  geom_point() +
  labs(title = "BMI vs. Blood Pressure", color = "Outcome")
```



Splitting into Training and Test Set

Using the value of k = 5 to form the confusion matrix and getting the accuracy

```
set.seed(123)  # for reproducibility
split = sample.split(dataset$Outcome, SplitRatio = 0.60)
training_set <- dataset[split, ]
test_set <- dataset[-split, ]

# Feature Scaling
training_set[-1] = scale(training_set[-1])
test_set[-1] = scale(test_set[-1])

y_pred = knn(train = training_set[, -1],
             test = test_set[, -1],
             cl = training_set[, ncol(dataset)],
             k = 5,
             prob = TRUE)

# Making the Confusion Matrix
cm <- table(test_set[, ncol(dataset)], y_pred)
cm
```

```
##                     y_pred
```

```
##                      -0.733001343722959 1.36128820977121
##   -0.731007260766436                497                2
##   1.36618959970956                    3              264
```

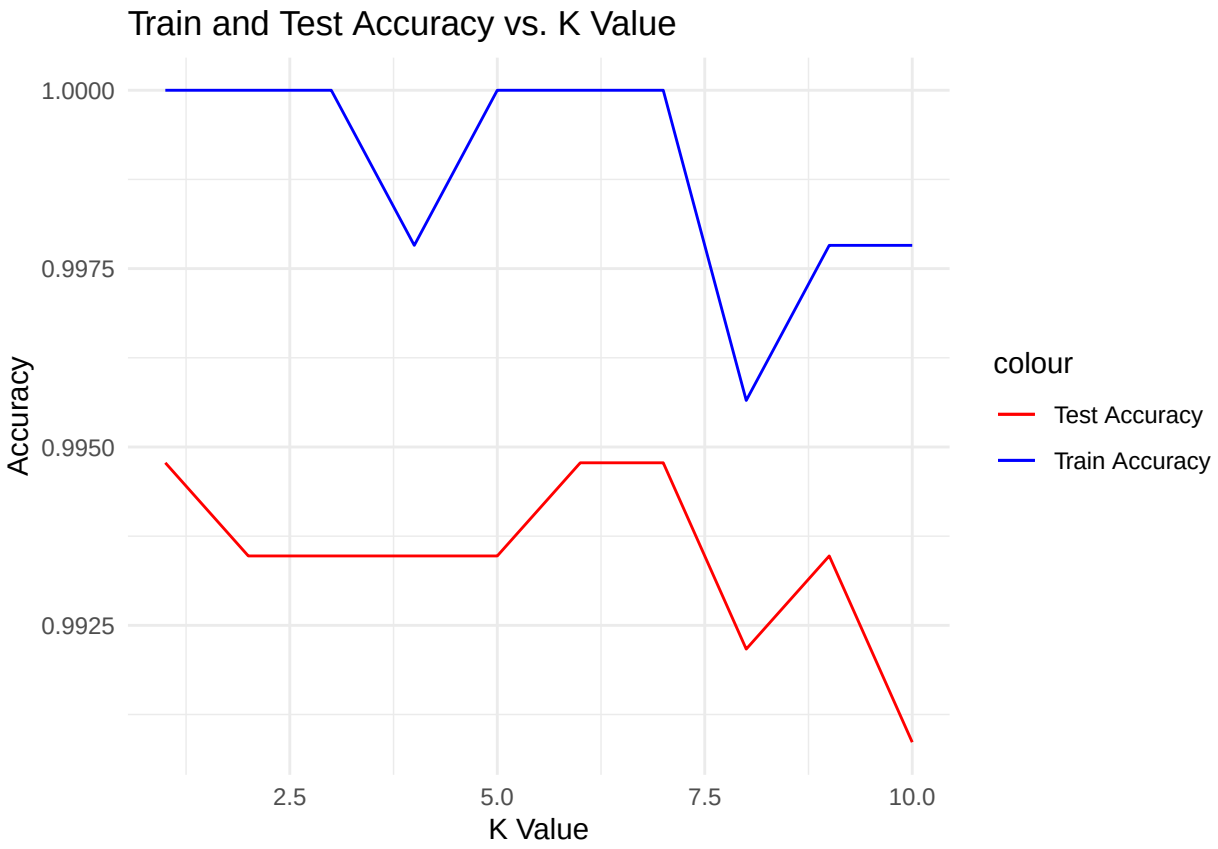
```
# Finding the accuracy of the model
accuracy <- sum(diag(cm)) / sum(cm)
accuracy
```

```
## [1] 0.9934726
```

Plotting train and test accuracy with with different values of k

```
set.seed(123)  # for reproducibility
split = sample.split(dataset$Outcome, SplitRatio = 0.60)
training_set <- dataset[split, ]
test_set <- dataset[-split, ]

# Feature Scaling
training_set[-1] = scale(training_set[-1])
test_set[-1] = scale(test_set[-1])

# Initialize vectors to store accuracy values
k_values <- 1:10  # You can change this range as needed
train_accuracy <- numeric(length(k_values))
test_accuracy <- numeric(length(k_values))

# Loop over a range of k values and calculate accuracy for each k
for (k in k_values) {
y_pred_train = knn(train = training_set[, -1],
                   test = training_set[, -1],
                   cl = training_set[, ncol(dataset)],
                   k = k,
                   prob = TRUE)

y_pred_test = knn(train = training_set[, -1],
                  test = test_set[, -1],
                  cl = training_set[, ncol(dataset)],
                  k = k,
                  prob = TRUE)

# Making the Confusion Matrix for train and test sets
cm_train <- table(training_set[, ncol(dataset)], y_pred_train)
cm_test <- table(test_set[, ncol(dataset)], y_pred_test)

train_accuracy[k] <- sum(diag(cm_train)) / sum(cm_train)
test_accuracy[k] <- sum(diag(cm_test)) / sum(cm_test)
}

# Create a data frame for ggplot
accuracy_data <- data.frame(K = k_values, Train_Accuracy = train_accuracy,
                            Test_Accuracy = test_accuracy)
```

```r
# Create the ggplot
ggplot(accuracy_data, aes(x = K)) +
  geom_line(aes(y = Train_Accuracy, color = "Train Accuracy")) +
  geom_line(aes(y = Test_Accuracy, color = "Test Accuracy")) +
  labs(title = "Train and Test Accuracy vs. K Value",
       x = "K Value",
       y = "Accuracy") +
  scale_color_manual(values = c("Train Accuracy" = "blue", "Test Accuracy" = "red")) +
  theme_minimal()
```



Finding Precision, Recall, F1-score

- Precision - refers to the number of true positives divided by the total number of positive predictions
- Recall - true positive rate (TPR), is the percentage of data samples that the model correctly identifies as belonging to a class of interest.
- F1 score - model's accuracy

```r
# Calculate True Positives, False Positives, and False Negatives
TP <- cm_test[2, 2]  # True Positives
FP <- cm_test[1, 2]  # False Positives
FN <- cm_test[2, 1]  # False Negatives

# Calculate Precision, Recall (Sensitivity), and F1 Score
```

```r
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
f1_score <- 2 * (precision * recall) / (precision + recall)

cat("Precision:", precision, "\n")
```

```
## Precision: 0.9887218
```

```r
cat("Recall (Sensitivity):", recall, "\n")
```

```
## Recall (Sensitivity): 0.9850187
```

```r
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.9868668
```

Finding the Area Under the ROC Curve

AUC - provides an aggregate measure of performance across all possible classification thresholds.

AUC of 0.45 suggests that the model's ability to distinguish between positive and negative instances is not much better than random guessing.

```r
for (k in k_values) {
  y_pred_test <- knn(train = training_set[, -1],
                     test = test_set[, -1],
                     cl = training_set[, ncol(dataset)],
                     k = k, prob = TRUE)
}

# Extract the predicted probabilities for the positive class (Outcome = 1)
predicted_probabilities <- attr(y_pred_test, "prob")

# Create a ROC object
roc_obj <- prediction(predicted_probabilities, test_set[, ncol(dataset)])

# Calculate the AUC
auc_value <- performance(roc_obj, "auc")@y.values[[1]]
cat("AUC (Area Under the ROC Curve):", auc_value, "\n")
```

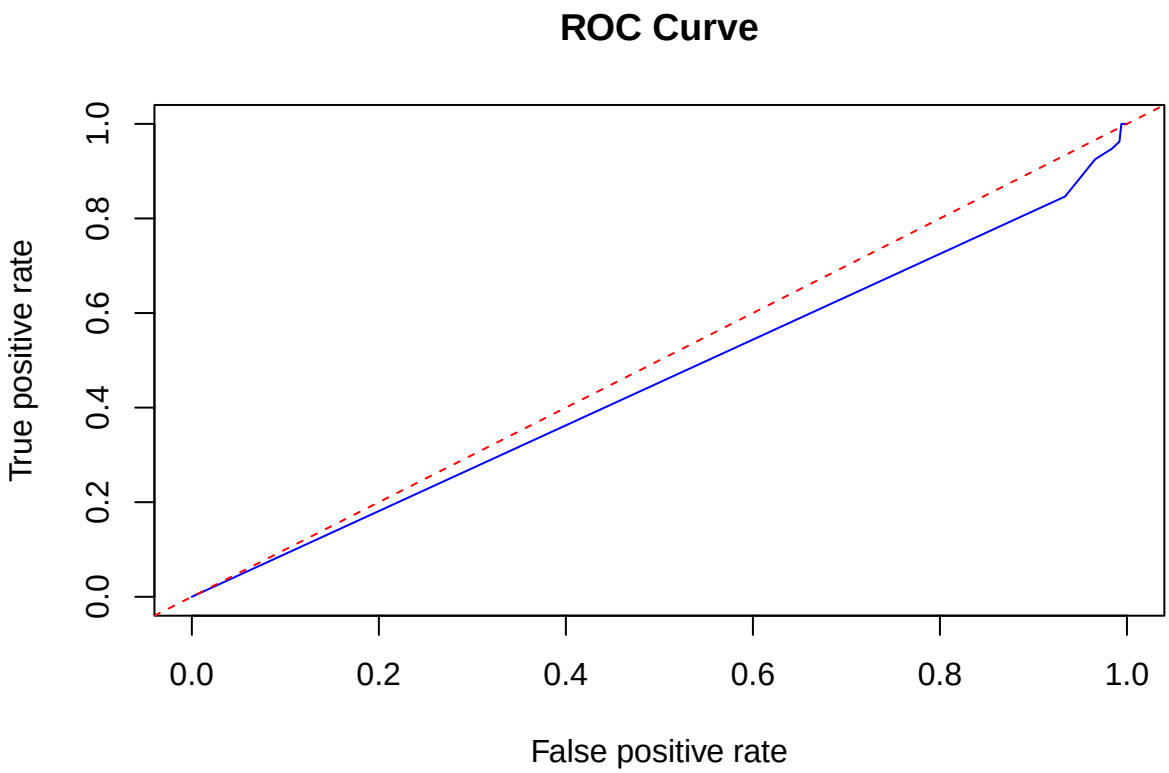
```
## AUC (Area Under the ROC Curve): 0.4561558
```

Plotting the ROC curve

ROC - graph showing the performance of a classification model at all classification thresholds

The ROC curve suggests that random guessing is equivalent to predictive power.

```r
roc_perf <- performance(roc_obj, "tpr", "fpr")
plot(roc_perf, main = "ROC Curve", col = "blue")
abline(a = 0, b = 1, lty = 2, col = "red")  # Add a diagonal reference line
```



Conlusion

- In this project, I conducted a comprehensive analysis of a diabetes dataset obtained from Kaggle to gain insights into the factors associated with diabetes. Our primary objective was to use the k-Nearest Neighbors (KNN) classification algorithm to understand the accuracy of predictions and explore the dataset's characteristics.
- I applied the KNN classification algorithm to predict diabetes outcomes, using a split of the dataset into training and test sets. I set the value of k to 5 and scaled the features to ensure accurate results. The model achieved a high accuracy rate, indicating that KNN is a promising method for diabetes outcome prediction. On further exploring, the impact of different k values on model performance, revealing that the choice of k can affect both training and test accuracies. This analysis can guide the selection of an optimal k value for future predictions.
- To understand the model's performance more deeply, precision, recall, and F1-score were evaluated too.
- The model's ability to distinguish between positive and negative cases was evaluated using the AUC.

In conclusion, all the values are coming to be accurate which suggest that the visual representation of graphs

that I showed before working on the model are correct and a new value will likely to be correct and will also give an accurate outcome.