Diabetes

Kiran

2025-01-20

# Lets load the dataset into the dataframe "diabetes\_data". " This dataset is downloaded from kaggle and this dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases  
  
getwd()

## [1] "/Users/kiran/Downloads/Diabetes Prediction app"

diabetes\_data <- read.csv("diabetes.csv")

# Lets know the data   
# Using head and tail functions to see the first and last few records in the dataframe   
head(diabetes\_data)

## 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

tail(diabetes\_data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI  
## 763 9 89 62 0 0 22.5  
## 764 10 101 76 48 180 32.9  
## 765 2 122 70 27 0 36.8  
## 766 5 121 72 23 112 26.2  
## 767 1 126 60 0 0 30.1  
## 768 1 93 70 31 0 30.4  
## DiabetesPedigreeFunction Age Outcome  
## 763 0.142 33 0  
## 764 0.171 63 0  
## 765 0.340 27 0  
## 766 0.245 30 0  
## 767 0.349 47 1  
## 768 0.315 23 0

# Using "dim" function to get the dimensions of the dataframe "diabetes\_data"  
dim(diabetes\_data)

## [1] 768 9

# There are 768 rows of data with 9 columns

# Lets see all the column names   
names(diabetes\_data)

## [1] "Pregnancies" "Glucose"   
## [3] "BloodPressure" "SkinThickness"   
## [5] "Insulin" "BMI"   
## [7] "DiabetesPedigreeFunction" "Age"   
## [9] "Outcome"

# Lets change the "Outcome" column name to "Result"  
# Rename the "Outcome" column to "Result"  
colnames(diabetes\_data)[colnames(diabetes\_data) == "Outcome"] <- "Result"  
  
# Verify the change  
names(diabetes\_data)

## [1] "Pregnancies" "Glucose"   
## [3] "BloodPressure" "SkinThickness"   
## [5] "Insulin" "BMI"   
## [7] "DiabetesPedigreeFunction" "Age"   
## [9] "Result"

# Lets check the summary and structure of the dataframe "diabetes\_data" to check for any missing values and any data inconsistencies   
  
str(diabetes\_data)

## 'data.frame': 768 obs. of 9 variables:  
## $ Pregnancies : int 6 1 8 1 0 5 3 10 2 8 ...  
## $ Glucose : int 148 85 183 89 137 116 78 115 197 125 ...  
## $ BloodPressure : int 72 66 64 66 40 74 50 0 70 96 ...  
## $ SkinThickness : int 35 29 0 23 35 0 32 0 45 0 ...  
## $ Insulin : int 0 0 0 94 168 0 88 0 543 0 ...  
## $ BMI : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...  
## $ DiabetesPedigreeFunction: num 0.627 0.351 0.672 0.167 2.288 ...  
## $ Age : int 50 31 32 21 33 30 26 29 53 54 ...  
## $ Result : int 1 0 1 0 1 0 1 0 1 1 ...

summary(diabetes\_data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 62.00 1st Qu.: 0.00   
## Median : 3.000 Median :117.0 Median : 72.00 Median :23.00   
## Mean : 3.845 Mean :120.9 Mean : 69.11 Mean :20.54   
## 3rd Qu.: 6.000 3rd Qu.:140.2 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.0 Max. :122.00 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.:0.2437 1st Qu.:24.00   
## Median : 30.5 Median :32.00 Median :0.3725 Median :29.00   
## Mean : 79.8 Mean :31.99 Mean :0.4719 Mean :33.24   
## 3rd Qu.:127.2 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00   
## Result   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.349   
## 3rd Qu.:1.000   
## Max. :1.000

# The columns Glucose, BloodPressure, SkinThickness, Insulin, and BMI contain 0 values, which are biologically unrealistic and likely represent missing data.  
  
# Lets replace 0 with NA  
# Replace 0 values in these columns with NA to handle them as missing values:  
  
# Replace 0 values with NA for relevant columns  
cols\_with\_missing <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")  
diabetes\_data[cols\_with\_missing] <- lapply(diabetes\_data[cols\_with\_missing], function(x) ifelse(x == 0, NA, x))  
  
# Verify the changes  
summary(diabetes\_data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 44.0 Min. : 24.00 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 64.00 1st Qu.:22.00   
## Median : 3.000 Median :117.0 Median : 72.00 Median :29.00   
## Mean : 3.845 Mean :121.7 Mean : 72.41 Mean :29.15   
## 3rd Qu.: 6.000 3rd Qu.:141.0 3rd Qu.: 80.00 3rd Qu.:36.00   
## Max. :17.000 Max. :199.0 Max. :122.00 Max. :99.00   
## NA's :5 NA's :35 NA's :227   
## Insulin BMI DiabetesPedigreeFunction Age   
## Min. : 14.00 Min. :18.20 Min. :0.0780 Min. :21.00   
## 1st Qu.: 76.25 1st Qu.:27.50 1st Qu.:0.2437 1st Qu.:24.00   
## Median :125.00 Median :32.30 Median :0.3725 Median :29.00   
## Mean :155.55 Mean :32.46 Mean :0.4719 Mean :33.24   
## 3rd Qu.:190.00 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.00 Max. :67.10 Max. :2.4200 Max. :81.00   
## NA's :374 NA's :11   
## Result   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.349   
## 3rd Qu.:1.000   
## Max. :1.000   
##

# Lets see the total sum of missing values   
sum(is.na(diabetes\_data))

## [1] 652

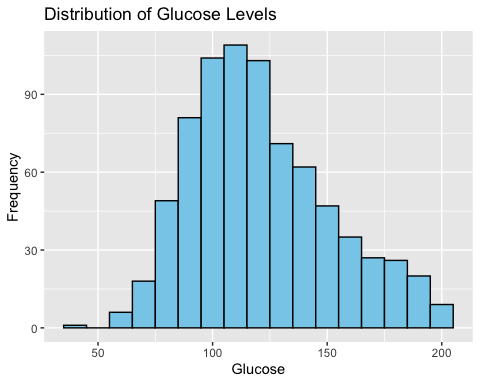
# There is no way we are going to delete the missing records

# Lets handle the missing values

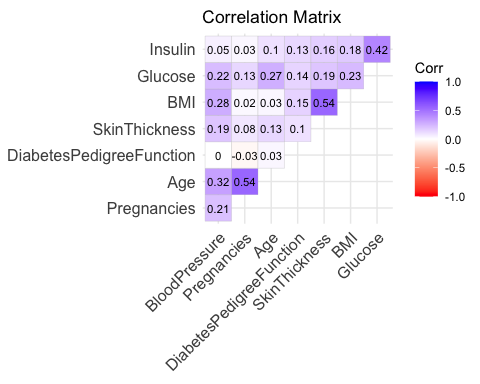
# Now lets use one of the imputating techniques   
# Impute missing values with median  
diabetes\_data[cols\_with\_missing] <- lapply(diabetes\_data[cols\_with\_missing], function(x) ifelse(is.na(x), median(x, na.rm = TRUE), x))  
  
# Verify the changes  
summary(diabetes\_data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 44.00 Min. : 24.00 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.: 99.75 1st Qu.: 64.00 1st Qu.:25.00   
## Median : 3.000 Median :117.00 Median : 72.00 Median :29.00   
## Mean : 3.845 Mean :121.66 Mean : 72.39 Mean :29.11   
## 3rd Qu.: 6.000 3rd Qu.:140.25 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.00 Max. :122.00 Max. :99.00   
## Insulin BMI DiabetesPedigreeFunction Age   
## Min. : 14.0 Min. :18.20 Min. :0.0780 Min. :21.00   
## 1st Qu.:121.5 1st Qu.:27.50 1st Qu.:0.2437 1st Qu.:24.00   
## Median :125.0 Median :32.30 Median :0.3725 Median :29.00   
## Mean :140.7 Mean :32.46 Mean :0.4719 Mean :33.24   
## 3rd Qu.:127.2 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00   
## Result   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.349   
## 3rd Qu.:1.000   
## Max. :1.000

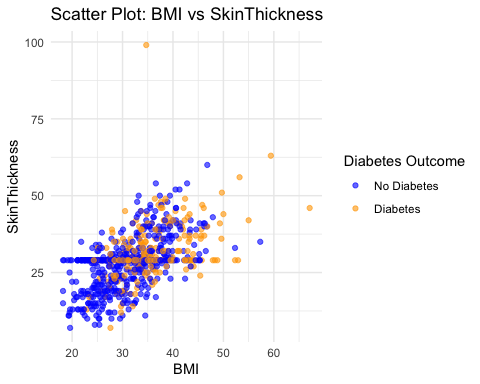
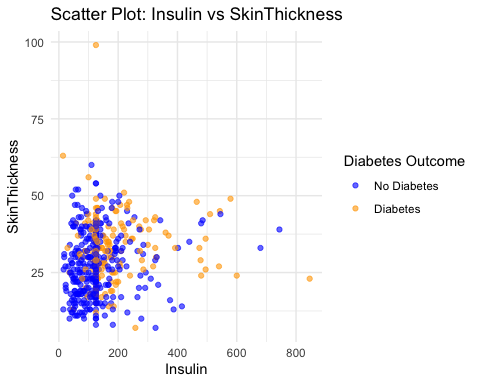
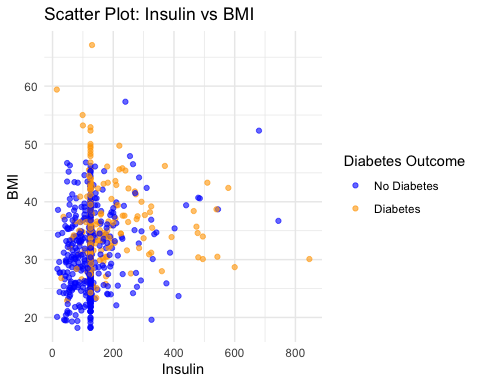
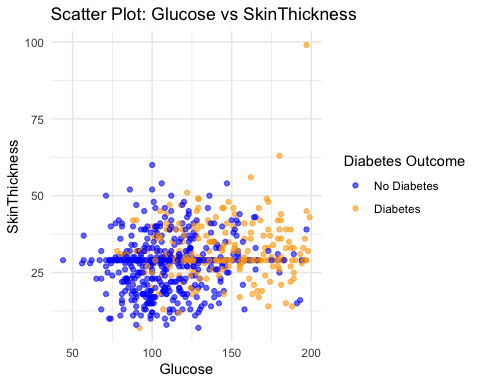
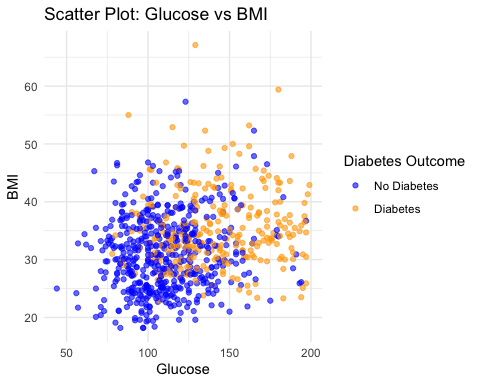
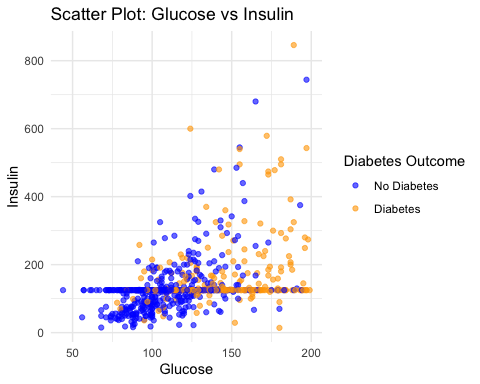
# Before moving to model training, lets visualize the data to understand its distribution and relationships.  
# Lets load the visualization library "ggplot2"  
  
library(ggplot2)  
  
histogram <- ggplot(diabetes\_data, aes(x = Glucose)) +  
 geom\_histogram(binwidth = 10, fill = "skyblue", color = "black") +  
 labs(title = "Distribution of Glucose Levels", x = "Glucose", y = "Frequency")  
  
print(histogram)



# Correlation Matrix:  
# Lets use a heatmap to analyze correlations between features.  
 # install.packages("ggcorrplot")  
library(ggcorrplot)  
  
# Compute the correlation matrix  
cor\_matrix <- cor(diabetes\_data[, -9]) # Exclude 'Result'  
  
# correlation plot  
ggcorrplot(cor\_matrix,  
 hc.order = TRUE, # Hierarchical clustering order  
 type = "upper", # Show only upper triangle  
 lab = TRUE, # Add correlation coefficients  
 lab\_size = 3, # Size of the labels  
 colors = c("red", "white", "blue"), # Gradient colors  
 title = "Correlation Matrix", # Add a title  
 ggtheme = theme\_minimal() # Clean ggplot theme  
)

 # The correlation matrix provides valuable insights into the relationships between the numeric variables in the diabetes dataset. Moderate positive correlations are observed between variables like BMI and SkinThickness (0.54), Pregnancies and Age (0.54), and Glucose and Insulin (0.42), suggesting these pairs may share underlying patterns relevant to diabetes prediction. Conversely, weak correlations among features like DiabetesPedigreeFunction and others indicate little to no linear relationship. These findings highlight the importance of certain features, such as Glucose, BMI, and Insulin, for further analysis and model building while suggesting minimal overlap in predictive value for weakly correlated variables.

library(ggplot2)  
  
# Convert 'Result' to a factor  
diabetes\_data$Result <- as.factor(diabetes\_data$Result)  
  
# Selected columns for analysis  
selected\_vars <- c("Glucose", "Insulin", "BMI", "SkinThickness")  
  
# Pairwise scatter plots with Result as color  
for (i in 1:(length(selected\_vars) - 1)) {  
 for (j in (i + 1):length(selected\_vars)) {  
 plot <- ggplot(diabetes\_data, aes(x = .data[[selected\_vars[i]]], y = .data[[selected\_vars[j]]], color = Result)) +  
 geom\_point(alpha = 0.6) +  
 scale\_color\_manual(values = c("blue", "orange"), labels = c("No Diabetes", "Diabetes")) +  
 labs(  
 title = paste("Scatter Plot:", selected\_vars[i], "vs", selected\_vars[j]),  
 x = selected\_vars[i],  
 y = selected\_vars[j],  
 color = "Diabetes Outcome"  
 ) +  
 theme\_minimal()  
 print(plot)  
 }  
}



# Feature Scaling

# Scaling selected features using Min-Max normalization  
cols\_to\_scale <- c("Glucose", "Insulin", "BMI", "SkinThickness")  
diabetes\_data[cols\_to\_scale] <- lapply(diabetes\_data[cols\_to\_scale], function(x) (x - min(x)) / (max(x) - min(x)))  
  
# Check scaled data  
summary(diabetes\_data[cols\_to\_scale])

## Glucose Insulin BMI SkinThickness   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3597 1st Qu.:0.1292 1st Qu.:0.1902 1st Qu.:0.1957   
## Median :0.4710 Median :0.1334 Median :0.2883 Median :0.2391   
## Mean :0.5010 Mean :0.1522 Mean :0.2915 Mean :0.2403   
## 3rd Qu.:0.6210 3rd Qu.:0.1361 3rd Qu.:0.3763 3rd Qu.:0.2717   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000

# Since the data is scaled lets split the dataset into two parts (training and testing sets )  
  
library(caret)

## Loading required package: lattice

# Set seed for reproducibility  
set.seed(123)  
  
# Split the dataset (80% training, 20% testing)  
train\_index <- createDataPartition(diabetes\_data$Result, p = 0.8, list = FALSE)  
train\_data <- diabetes\_data[train\_index, ] # 80 % train data  
test\_data <- diabetes\_data[-train\_index, ] # 20 % test data

# Model Training

library(caret)  
  
  
# Convert the Result column to a factor if it isn't already  
train\_data$Result <- factor(train\_data$Result, levels = c(0, 1), labels = c("No", "Yes"))  
test\_data$Result <- factor(test\_data$Result, levels = c(0, 1), labels = c("No", "Yes"))  
  
# Verify the changes  
levels(train\_data$Result)

## [1] "No" "Yes"

levels(test\_data$Result)

## [1] "No" "Yes"

# Create a control object for cross-validation  
control <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)  
  
# Logistic Regression  
lr\_model <- train(Result ~ ., data = train\_data, method = "glm", family = "binomial", trControl = control, metric = "ROC")  
  
# K-Nearest Neighbors  
knn\_model <- train(Result ~ ., data = train\_data, method = "knn", tuneLength = 10, trControl = control, metric = "ROC")  
  
# Decision Tree  
dt\_model <- train(Result ~ ., data = train\_data, method = "rpart", trControl = control, metric = "ROC")  
  
# Support Vector Machine  
svm\_model <- train(Result ~ ., data = train\_data, method = "svmRadial", trControl = control, metric = "ROC")  
  
# Naïve Bayes  
nb\_model <- train(Result ~ ., data = train\_data, method = "naive\_bayes", trControl = control, metric = "ROC")

# Evaluate Model Performance

# Predict probabilities on the test set  
test\_predictions <- predict(lr\_model, test\_data, type = "prob")  
  
# Predict classes based on probabilities  
test\_pred\_classes <- predict(lr\_model, test\_data)  
  
# Confusion matrix  
conf\_matrix <- confusionMatrix(test\_pred\_classes, test\_data$Result, positive = "Yes")  
  
# Print confusion matrix and metrics  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 91 21  
## Yes 9 32  
##   
## Accuracy : 0.8039   
## 95% CI : (0.7321, 0.8636)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 3.3e-05   
##   
## Kappa : 0.5426   
##   
## Mcnemar's Test P-Value : 0.04461   
##   
## Sensitivity : 0.6038   
## Specificity : 0.9100   
## Pos Pred Value : 0.7805   
## Neg Pred Value : 0.8125   
## Prevalence : 0.3464   
## Detection Rate : 0.2092   
## Detection Prevalence : 0.2680   
## Balanced Accuracy : 0.7569   
##   
## 'Positive' Class : Yes   
##

# ROC Curve  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

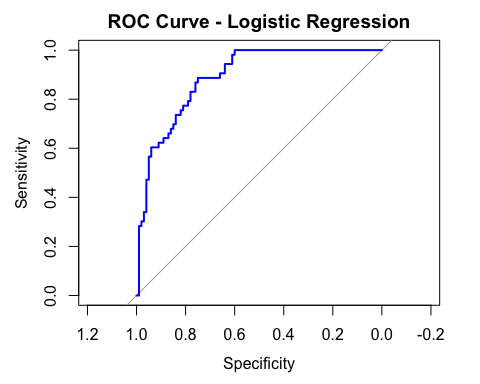
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

roc\_curve <- roc(test\_data$Result, test\_predictions$Yes) # Probability of "Yes"

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(roc\_curve, col = "blue", main = "ROC Curve - Logistic Regression")



auc\_value <- auc(roc\_curve)  
cat("AUC:", auc\_value, "\n")

## AUC: 0.8903774

# Comparing with other models   
# Train K-NN Model  
knn\_model <- train(Result ~ ., data = train\_data, method = "knn", trControl = control, metric = "ROC")  
  
# Evaluate K-NN Model  
knn\_predictions <- predict(knn\_model, test\_data, type = "prob")  
knn\_pred\_classes <- predict(knn\_model, test\_data)  
knn\_conf\_matrix <- confusionMatrix(knn\_pred\_classes, test\_data$Result, positive = "Yes")  
print(knn\_conf\_matrix)

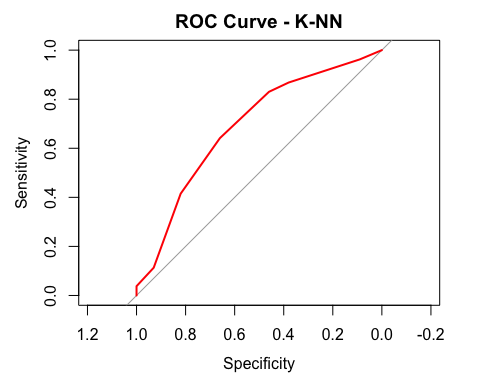
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 82 31  
## Yes 18 22  
##   
## Accuracy : 0.6797   
## 95% CI : (0.5996, 0.7528)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.27800   
##   
## Kappa : 0.2495   
##   
## Mcnemar's Test P-Value : 0.08648   
##   
## Sensitivity : 0.4151   
## Specificity : 0.8200   
## Pos Pred Value : 0.5500   
## Neg Pred Value : 0.7257   
## Prevalence : 0.3464   
## Detection Rate : 0.1438   
## Detection Prevalence : 0.2614   
## Balanced Accuracy : 0.6175   
##   
## 'Positive' Class : Yes   
##

# ROC Curve for K-NN  
knn\_roc <- roc(test\_data$Result, knn\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(knn\_roc, col = "red", main = "ROC Curve - K-NN")



auc\_knn <- auc(knn\_roc)  
cat("K-NN AUC:", auc\_knn, "\n")

## K-NN AUC: 0.6876415

# Train and evaluate decison tree

# Train Decision Tree  
dt\_model <- train(Result ~ ., data = train\_data, method = "rpart", trControl = control, metric = "ROC")  
  
# Predict and Evaluate  
dt\_predictions <- predict(dt\_model, test\_data, type = "prob")  
dt\_pred\_classes <- predict(dt\_model, test\_data)  
dt\_conf\_matrix <- confusionMatrix(dt\_pred\_classes, test\_data$Result, positive = "Yes")  
print(dt\_conf\_matrix)

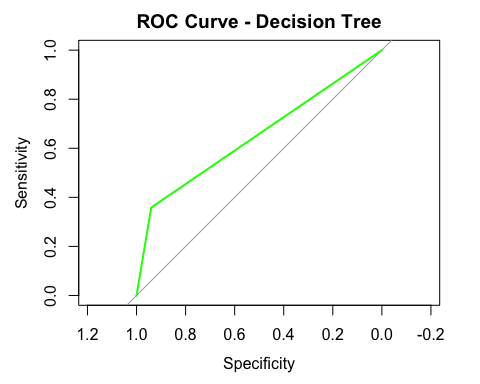
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 94 34  
## Yes 6 19  
##   
## Accuracy : 0.7386   
## 95% CI : (0.6615, 0.8062)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.01536   
##   
## Kappa : 0.3408   
##   
## Mcnemar's Test P-Value : 1.963e-05   
##   
## Sensitivity : 0.3585   
## Specificity : 0.9400   
## Pos Pred Value : 0.7600   
## Neg Pred Value : 0.7344   
## Prevalence : 0.3464   
## Detection Rate : 0.1242   
## Detection Prevalence : 0.1634   
## Balanced Accuracy : 0.6492   
##   
## 'Positive' Class : Yes   
##

# ROC Curve  
dt\_roc <- roc(test\_data$Result, dt\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(dt\_roc, col = "green", main = "ROC Curve - Decision Tree")



cat("Decision Tree AUC:", auc(dt\_roc), "\n")

## Decision Tree AUC: 0.6492453

# Train and Evaluate Naïve Bayes

# Train Naïve Bayes  
nb\_model <- train(Result ~ ., data = train\_data, method = "naive\_bayes", trControl = control, metric = "ROC")  
  
# Predict and Evaluate  
nb\_predictions <- predict(nb\_model, test\_data, type = "prob")  
nb\_pred\_classes <- predict(nb\_model, test\_data)  
nb\_conf\_matrix <- confusionMatrix(nb\_pred\_classes, test\_data$Result, positive = "Yes")  
print(nb\_conf\_matrix)

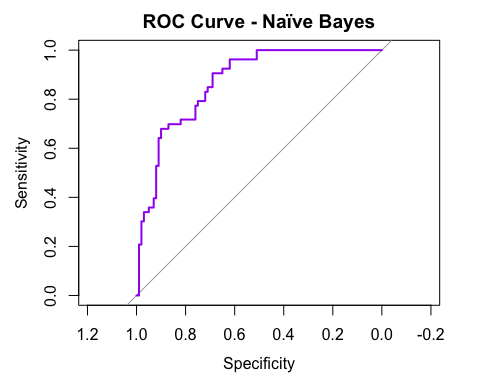
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 84 16  
## Yes 16 37  
##   
## Accuracy : 0.7908   
## 95% CI : (0.7178, 0.8523)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.0001499   
##   
## Kappa : 0.5381   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.6981   
## Specificity : 0.8400   
## Pos Pred Value : 0.6981   
## Neg Pred Value : 0.8400   
## Prevalence : 0.3464   
## Detection Rate : 0.2418   
## Detection Prevalence : 0.3464   
## Balanced Accuracy : 0.7691   
##   
## 'Positive' Class : Yes   
##

# ROC Curve  
nb\_roc <- roc(test\_data$Result, nb\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(nb\_roc, col = "purple", main = "ROC Curve - Naïve Bayes")



cat("Naïve Bayes AUC:", auc(nb\_roc), "\n")

## Naïve Bayes AUC: 0.8696226

# Train and Evaluate SVM

# Train SVM  
svm\_model <- train(Result ~ ., data = train\_data, method = "svmRadial", trControl = control, metric = "ROC")  
  
# Predict and Evaluate  
svm\_predictions <- predict(svm\_model, test\_data, type = "prob")  
svm\_pred\_classes <- predict(svm\_model, test\_data)  
svm\_conf\_matrix <- confusionMatrix(svm\_pred\_classes, test\_data$Result, positive = "Yes")  
print(svm\_conf\_matrix)

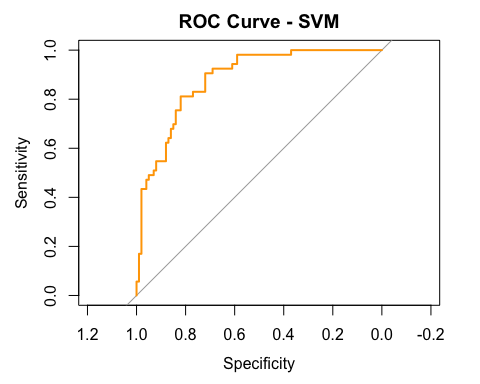
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 87 20  
## Yes 13 33  
##   
## Accuracy : 0.7843   
## 95% CI : (0.7106, 0.8466)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.0003018   
##   
## Kappa : 0.5084   
##   
## Mcnemar's Test P-Value : 0.2962699   
##   
## Sensitivity : 0.6226   
## Specificity : 0.8700   
## Pos Pred Value : 0.7174   
## Neg Pred Value : 0.8131   
## Prevalence : 0.3464   
## Detection Rate : 0.2157   
## Detection Prevalence : 0.3007   
## Balanced Accuracy : 0.7463   
##   
## 'Positive' Class : Yes   
##

# ROC Curve  
svm\_roc <- roc(test\_data$Result, svm\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(svm\_roc, col = "orange", main = "ROC Curve - SVM")



cat("SVM AUC:", auc(svm\_roc), "\n")

## SVM AUC: 0.8818868

# Decision Tree Evaluation

# Predict probabilities for Decision Tree  
dt\_predictions <- predict(dt\_model, test\_data, type = "prob")  
  
# Predict classes for Decision Tree  
dt\_pred\_classes <- predict(dt\_model, test\_data)  
  
# Confusion matrix for Decision Tree  
dt\_conf\_matrix <- confusionMatrix(dt\_pred\_classes, test\_data$Result, positive = "Yes")  
print(dt\_conf\_matrix)

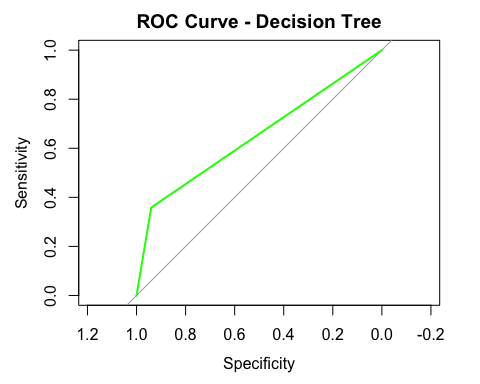
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 94 34  
## Yes 6 19  
##   
## Accuracy : 0.7386   
## 95% CI : (0.6615, 0.8062)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.01536   
##   
## Kappa : 0.3408   
##   
## Mcnemar's Test P-Value : 1.963e-05   
##   
## Sensitivity : 0.3585   
## Specificity : 0.9400   
## Pos Pred Value : 0.7600   
## Neg Pred Value : 0.7344   
## Prevalence : 0.3464   
## Detection Rate : 0.1242   
## Detection Prevalence : 0.1634   
## Balanced Accuracy : 0.6492   
##   
## 'Positive' Class : Yes   
##

# ROC Curve for Decision Tree  
dt\_roc\_curve <- roc(test\_data$Result, dt\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(dt\_roc\_curve, col = "green", main = "ROC Curve - Decision Tree")



auc\_dt <- auc(dt\_roc\_curve)  
cat("Decision Tree AUC:", auc\_dt, "\n")

## Decision Tree AUC: 0.6492453

# Support Vector Machine (SVM) Evaluation

# Predict probabilities for SVM  
svm\_predictions <- predict(svm\_model, test\_data, type = "prob")  
  
# Predict classes for SVM  
svm\_pred\_classes <- predict(svm\_model, test\_data)  
  
# Confusion matrix for SVM  
svm\_conf\_matrix <- confusionMatrix(svm\_pred\_classes, test\_data$Result, positive = "Yes")  
print(svm\_conf\_matrix)

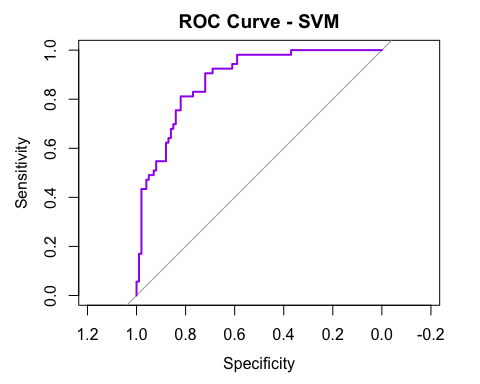
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 87 20  
## Yes 13 33  
##   
## Accuracy : 0.7843   
## 95% CI : (0.7106, 0.8466)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.0003018   
##   
## Kappa : 0.5084   
##   
## Mcnemar's Test P-Value : 0.2962699   
##   
## Sensitivity : 0.6226   
## Specificity : 0.8700   
## Pos Pred Value : 0.7174   
## Neg Pred Value : 0.8131   
## Prevalence : 0.3464   
## Detection Rate : 0.2157   
## Detection Prevalence : 0.3007   
## Balanced Accuracy : 0.7463   
##   
## 'Positive' Class : Yes   
##

# ROC Curve for SVM  
svm\_roc\_curve <- roc(test\_data$Result, svm\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(svm\_roc\_curve, col = "purple", main = "ROC Curve - SVM")



auc\_svm <- auc(svm\_roc\_curve)  
cat("SVM AUC:", auc\_svm, "\n")

## SVM AUC: 0.8818868

#Naïve Bayes Evaluation

# Predict probabilities for Naïve Bayes  
nb\_predictions <- predict(nb\_model, test\_data, type = "prob")  
  
# Predict classes for Naïve Bayes  
nb\_pred\_classes <- predict(nb\_model, test\_data)  
  
# Confusion matrix for Naïve Bayes  
nb\_conf\_matrix <- confusionMatrix(nb\_pred\_classes, test\_data$Result, positive = "Yes")  
print(nb\_conf\_matrix)

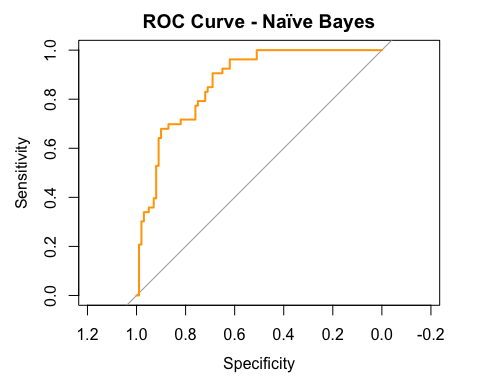
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 84 16  
## Yes 16 37  
##   
## Accuracy : 0.7908   
## 95% CI : (0.7178, 0.8523)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 0.0001499   
##   
## Kappa : 0.5381   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.6981   
## Specificity : 0.8400   
## Pos Pred Value : 0.6981   
## Neg Pred Value : 0.8400   
## Prevalence : 0.3464   
## Detection Rate : 0.2418   
## Detection Prevalence : 0.3464   
## Balanced Accuracy : 0.7691   
##   
## 'Positive' Class : Yes   
##

# ROC Curve for Naïve Bayes  
nb\_roc\_curve <- roc(test\_data$Result, nb\_predictions$Yes)

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(nb\_roc\_curve, col = "orange", main = "ROC Curve - Naïve Bayes")



auc\_nb <- auc(nb\_roc\_curve)  
cat("Naïve Bayes AUC:", auc\_nb, "\n")

## Naïve Bayes AUC: 0.8696226

# Performance comparison

# Initialize an empty data frame to store results  
results <- data.frame(  
 Model = character(),  
 Accuracy = numeric(),  
 AUC = numeric(),  
 Sensitivity = numeric(),  
 Specificity = numeric(),  
 stringsAsFactors = FALSE  
)  
  
# Function to calculate metrics  
get\_metrics <- function(model, test\_data, model\_name) {  
 # Predictions  
 predictions\_prob <- predict(model, test\_data, type = "prob")  
 predictions <- predict(model, test\_data)  
   
 # Confusion matrix  
 conf\_matrix <- confusionMatrix(predictions, test\_data$Result, positive = "Yes")  
   
 # ROC and AUC  
 roc\_curve <- roc(test\_data$Result, predictions\_prob$Yes)  
 auc\_value <- auc(roc\_curve)  
   
 # Add metrics to results data frame  
 results <<- rbind(results, data.frame(  
 Model = model\_name,  
 Accuracy = conf\_matrix$overall["Accuracy"],  
 AUC = auc\_value,  
 Sensitivity = conf\_matrix$byClass["Sensitivity"],  
 Specificity = conf\_matrix$byClass["Specificity"]  
 ))  
}  
  
# Evaluate all models  
get\_metrics(lr\_model, test\_data, "Logistic Regression")

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

get\_metrics(knn\_model, test\_data, "KNN")

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

get\_metrics(dt\_model, test\_data, "Decision Tree")

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

get\_metrics(svm\_model, test\_data, "SVM")

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

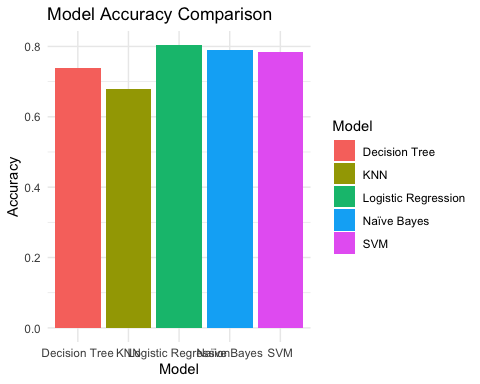
get\_metrics(nb\_model, test\_data, "Naïve Bayes")

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

# Print the results  
print(results)

## Model Accuracy AUC Sensitivity Specificity  
## Accuracy Logistic Regression 0.8039216 0.8903774 0.6037736 0.91  
## Accuracy1 KNN 0.6797386 0.6876415 0.4150943 0.82  
## Accuracy2 Decision Tree 0.7385621 0.6492453 0.3584906 0.94  
## Accuracy3 SVM 0.7843137 0.8818868 0.6226415 0.87  
## Accuracy4 Naïve Bayes 0.7908497 0.8696226 0.6981132 0.84

library(ggplot2)  
  
# Simple bar plot for accuracy comparison  
ggplot(results, aes(x = Model, y = Accuracy, fill = Model)) +  
 geom\_bar(stat = "identity") +  
 labs(  
 title = "Model Accuracy Comparison",  
 x = "Model",  
 y = "Accuracy"  
 ) +  
 theme\_minimal()

 # Training the model

library(caret)  
  
# Train Logistic Regression Model  
control <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)  
lr\_model <- train(Result ~ ., data = train\_data, method = "glm", family = "binomial", trControl = control, metric = "ROC")  
  
# Save the Model  
saveRDS(lr\_model, "best\_model.rds")

```