Practical 7

Model evaluation

Joan Navarro Bellido

```r
# Load required packages
library(caret)
library(dplyr)
library(ggplot2)
library(lattice)
library(ROCR)
```

Exercice 1

```r
# Load the dataset
tic_tac_toe <- read.csv("./dataset/tic-tac-toe.txt", header = FALSE)
head(tic_tac_toe)
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9      V10
## 1  x  x  x  x  o  o  x  o  o positive
## 2  x  x  x  x  o  o  o  x  o positive
## 3  x  x  x  x  o  o  o  o  x positive
## 4  x  x  x  x  o  o  o  b  b positive
## 5  x  x  x  x  o  o  b  o  b positive
## 6  x  x  x  x  o  o  b  b  o positive
```

```r
# Assign column names
names(tic_tac_toe) <- c("top_left", "top_middle", "top_right",
                        "middle_left", "middle_middle", "middle_right",
                        "bottom_left", "bottom_middle", "bottom_right",
                        "Class")

# Check structure of the dataset and look for missing values
str(tic_tac_toe)
```

```
## 'data.frame':    958 obs. of  10 variables:
##  $ top_left     : chr  "x" "x" "x" "x" ...
##  $ top_middle   : chr  "x" "x" "x" "x" ...
##  $ top_right    : chr  "x" "x" "x" "x" ...
##  $ middle_left  : chr  "x" "x" "x" "x" ...
##  $ middle_middle: chr  "o" "o" "o" "o" ...
##  $ middle_right : chr  "o" "o" "o" "o" ...
##  $ bottom_left  : chr  "x" "o" "o" "o" ...
##  $ bottom_middle: chr  "o" "x" "o" "b" ...
##  $ bottom_right : chr  "o" "o" "x" "b" ...
##  $ Class        : chr  "positive" "positive" "positive" "positive" ...
```

```r
any(is.na(tic_tac_toe))
```

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

Exercice 2

```r
# Create the split: 70% training, 30% testing
train_index <- createDataPartition(tic_tac_toe$Class, p = 0.7, list = FALSE)

# Split the data into training and testing sets
train_data <- tic_tac_toe[train_index, ]
test_data <- tic_tac_toe[-train_index, ]

# Verify the structure of the training and testing data
str(train_data)
```

```
## 'data.frame':    672 obs. of  10 variables:
##  $ top_left     : chr  "x" "x" "x" "x" ...
##  $ top_middle   : chr  "x" "x" "x" "x" ...
##  $ top_right    : chr  "x" "x" "x" "x" ...
##  $ middle_left  : chr  "x" "x" "x" "x" ...
##  $ middle_middle: chr  "o" "o" "o" "o" ...
##  $ middle_right : chr  "o" "o" "o" "b" ...
##  $ bottom_left  : chr  "o" "o" "b" "o" ...
##  $ bottom_middle: chr  "x" "o" "o" "o" ...
##  $ bottom_right : chr  "o" "x" "b" "b" ...
##  $ Class        : chr  "positive" "positive" "positive" "positive" ...
```

```r
str(test_data)
```

```
## 'data.frame':    286 obs. of  10 variables:
##  $ top_left     : chr  "x" "x" "x" "x" ...
##  $ top_middle   : chr  "x" "x" "x" "x" ...
##  $ top_right    : chr  "x" "x" "x" "x" ...
##  $ middle_left  : chr  "x" "x" "x" "x" ...
##  $ middle_middle: chr  "o" "o" "o" "b" ...
##  $ middle_right : chr  "o" "o" "o" "o" ...
##  $ bottom_left  : chr  "x" "o" "b" "b" ...
##  $ bottom_middle: chr  "o" "b" "b" "o" ...
##  $ bottom_right : chr  "o" "b" "o" "o" ...
##  $ Class        : chr  "positive" "positive" "positive" "positive" ...
```

Exercice 3

```r
# Set up 10-fold cross-validation
control <- trainControl(method = "cv", number = 10)

# Naive Bayes Model
nb_model <- train(Class ~ ., data = train_data, method = "nb", trControl = control)

# Decision Tree Model
dt_model <- train(Class ~ ., data = train_data, method = "rpart", trControl = control)
```

```r
# Neural Network Model
nn_model <- train(Class ~ ., data = train_data, method = "nnet", trControl = control, trace = FALSE)

# k-Nearest Neighbors Model
knn_model <- train(Class ~ ., data = train_data, method = "knn", trControl = control)

# Support Vector Machine Model
svm_model <- train(Class ~ ., data = train_data, method = "svmLinear", trControl = control)
```

Exercice 4

```r
# Ensure Class is a factor with consistent levels in training and test data
train_data$Class <- factor(train_data$Class)
test_data$Class <- factor(test_data$Class, levels = levels(train_data$Class))

# Naive Bayes Predictions and Evaluation
nb_pred <- factor(predict(nb_model, newdata = test_data), levels = levels(test_data$Class))
nb_confusion <- confusionMatrix(nb_pred, test_data$Class)
print(nb_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       52       47
##   positive       47      140
##
##                Accuracy : 0.6713
##                  95% CI : (0.6136, 0.7255)
##     No Information Rate : 0.6538
##     P-Value [Acc > NIR] : 0.2895
##
##                   Kappa : 0.2739
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.5253
##             Specificity : 0.7487
##          Pos Pred Value : 0.5253
##          Neg Pred Value : 0.7487
##              Prevalence : 0.3462
##          Detection Rate : 0.1818
##    Detection Prevalence : 0.3462
##       Balanced Accuracy : 0.6370
##
##        'Positive' Class : negative
##
```

```r
# Decision Tree Predictions and Evaluation
dt_pred <- factor(predict(dt_model, newdata = test_data), levels = levels(test_data$Class))
dt_confusion <- confusionMatrix(dt_pred, test_data$Class)
print(dt_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       42       24
##   positive       57      163
##
##                Accuracy : 0.7168
##                  95% CI : (0.6608, 0.7683)
##     No Information Rate : 0.6538
##     P-Value [Acc > NIR] : 0.0138143
##
##                   Kappa : 0.3211
##
##  Mcnemar's Test P-Value : 0.0003772
##
##             Sensitivity : 0.4242
##             Specificity : 0.8717
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.7409
##              Prevalence : 0.3462
##          Detection Rate : 0.1469
##    Detection Prevalence : 0.2308
##       Balanced Accuracy : 0.6480
##
##        'Positive' Class : negative
##
```

```
# Neural Network Predictions and Evaluation
nn_pred <- factor(predict(nn_model, newdata = test_data), levels = levels(test_data$Class))
nn_confusion <- confusionMatrix(nn_pred, test_data$Class)
print(nn_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       94        0
##   positive        5      187
##
##                Accuracy : 0.9825
##                  95% CI : (0.9597, 0.9943)
##     No Information Rate : 0.6538
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9609
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.9495
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9740
##              Prevalence : 0.3462
##          Detection Rate : 0.3287
```

```
##    Detection Prevalence : 0.3287
##       Balanced Accuracy : 0.9747
##
##        'Positive' Class : negative
##
```

```
# k-Nearest Neighbors Predictions and Evaluation
knn_pred <- factor(predict(knn_model, newdata = test_data), levels = levels(test_data$Class))
knn_confusion <- confusionMatrix(knn_pred, test_data$Class)
print(knn_confusion)
```

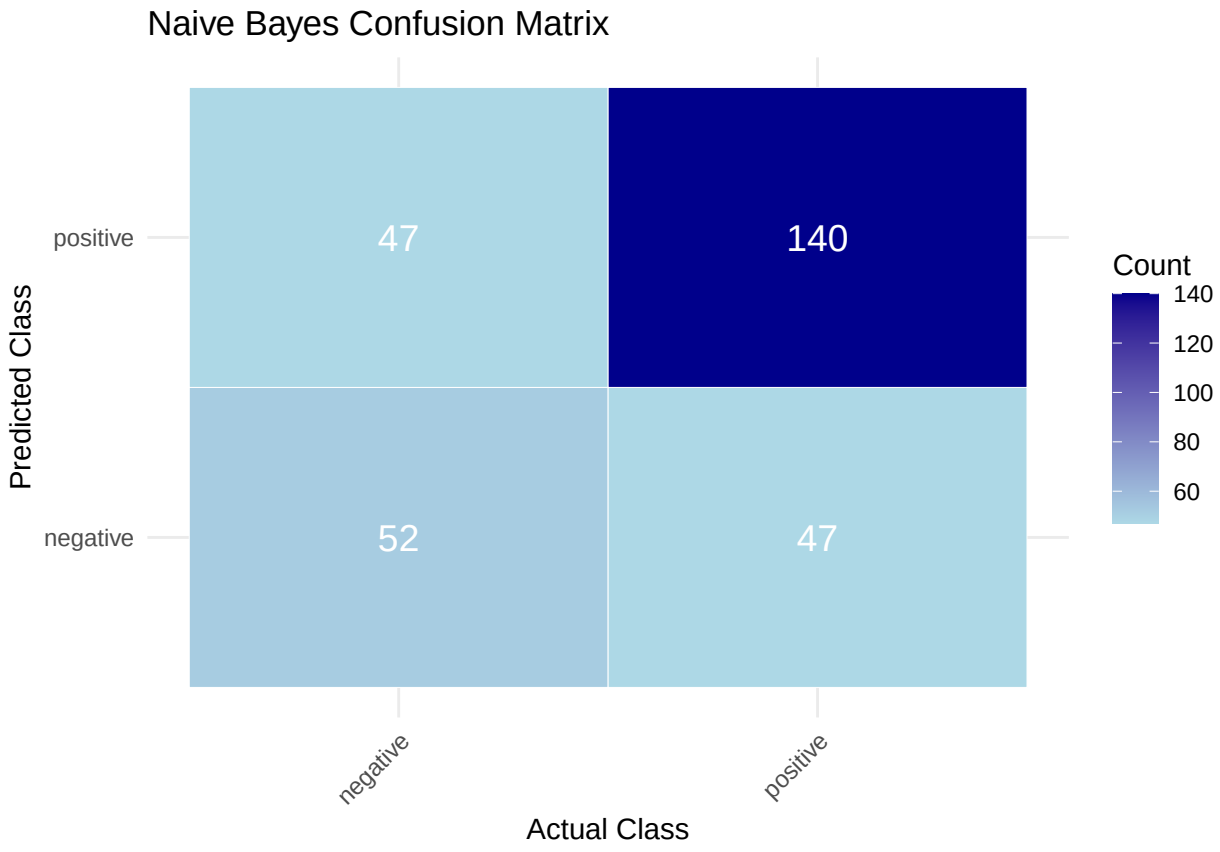
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       94        3
##   positive        5      184
##
##                Accuracy : 0.972
##                  95% CI : (0.9456, 0.9878)
##     No Information Rate : 0.6538
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9379
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9495
##             Specificity : 0.9840
##          Pos Pred Value : 0.9691
##          Neg Pred Value : 0.9735
##              Prevalence : 0.3462
##          Detection Rate : 0.3287
##    Detection Prevalence : 0.3392
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : negative
##
```

```
# SVM Predictions and Evaluation
svm_pred <- factor(predict(svm_model, newdata = test_data), levels = levels(test_data$Class))
svm_confusion <- confusionMatrix(svm_pred, test_data$Class)
print(svm_confusion)
```

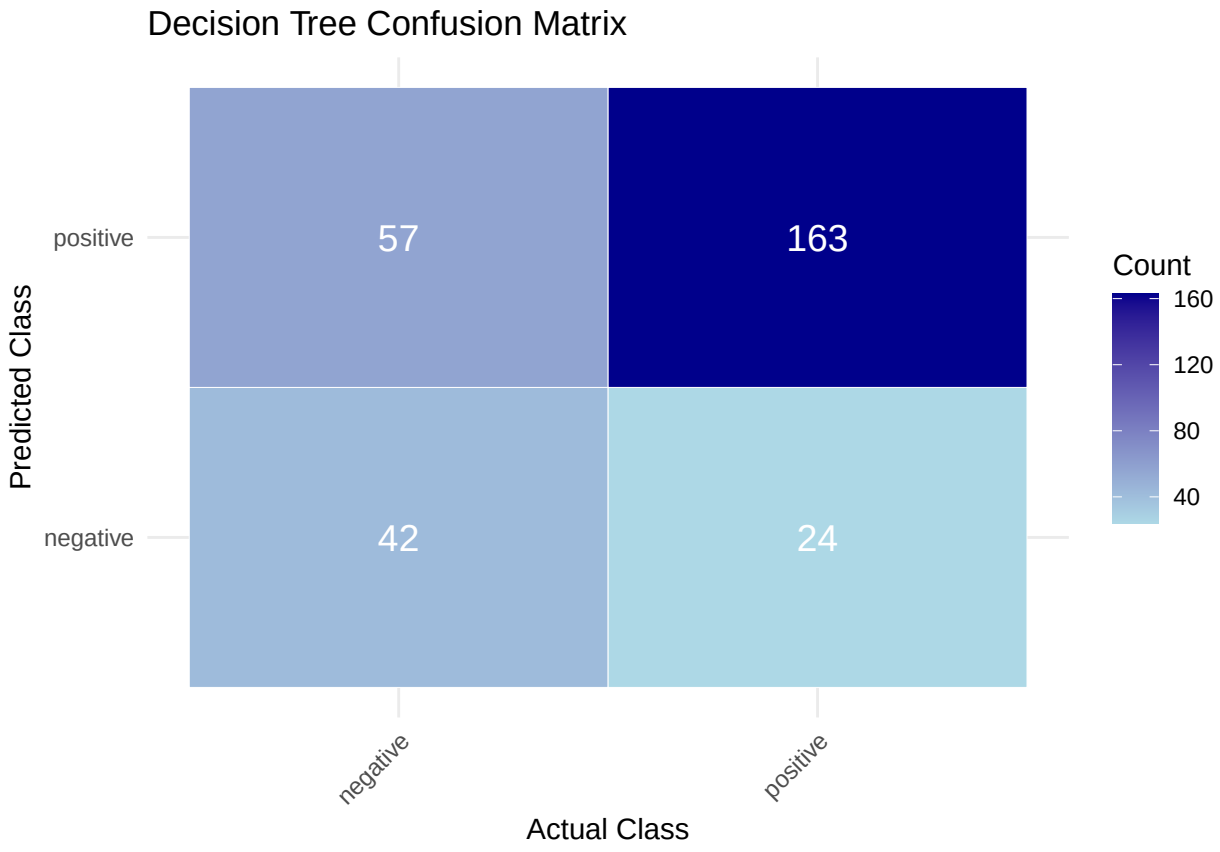
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       94        0
##   positive        5      187
##
##                Accuracy : 0.9825
##                  95% CI : (0.9597, 0.9943)
##     No Information Rate : 0.6538
##     P-Value [Acc > NIR] : < 2e-16
##
```

```
##                   Kappa : 0.9609
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.9495
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9740
##              Prevalence : 0.3462
##          Detection Rate : 0.3287
##    Detection Prevalence : 0.3287
##       Balanced Accuracy : 0.9747
##
##        'Positive' Class : negative
##
```

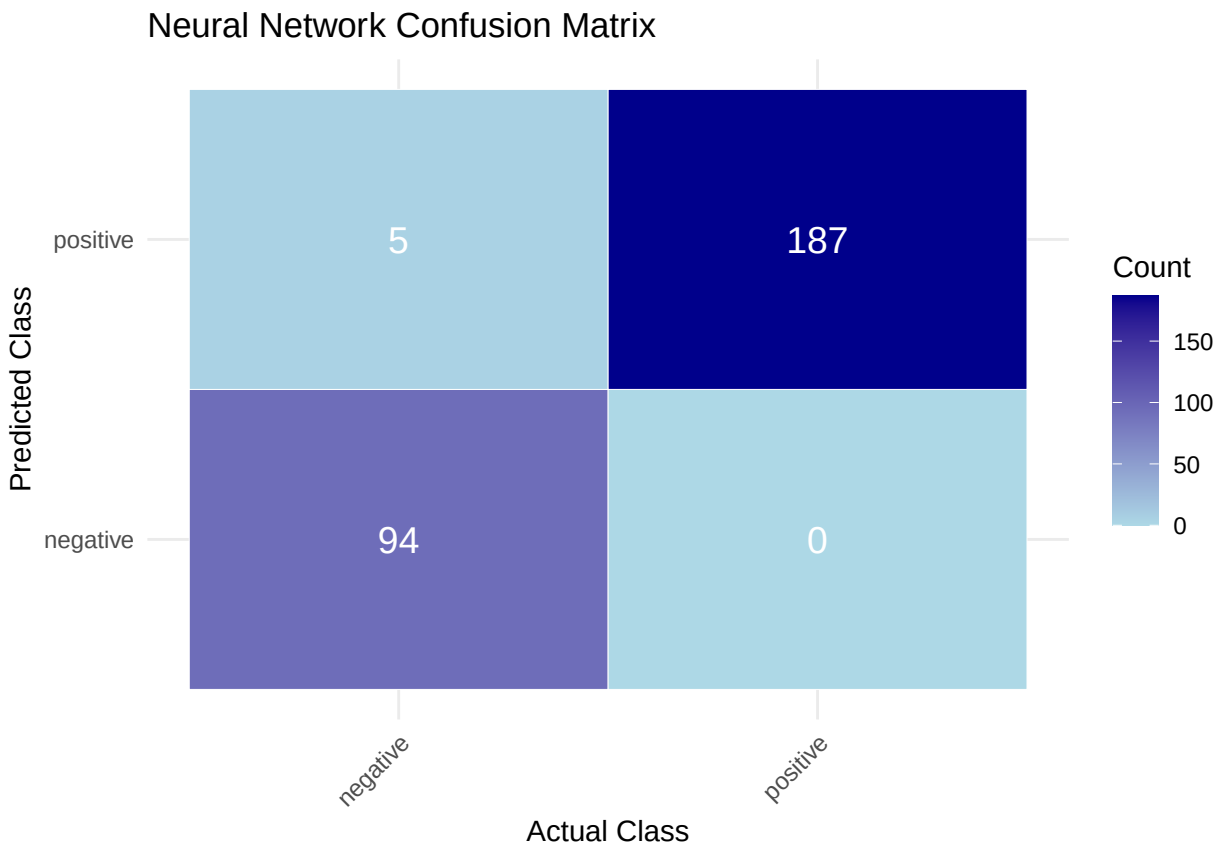
```r
# Function to visualize a confusion matrix as a heatmap
plot_confusion_matrix <- function(conf_matrix, title = "Confusion Matrix") {
    cm_df <- as.data.frame(conf_matrix$table)
    colnames(cm_df) <- c("Prediction", "Reference", "Count")

    ggplot(cm_df, aes(x = Reference, y = Prediction, fill = Count)) +
        geom_tile(color = "white") +
        scale_fill_gradient(low = "lightblue", high = "darkblue") +
        geom_text(aes(label = Count), color = "white", size = 5) +
        labs(title = title, x = "Actual Class", y = "Predicted Class") +
        theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))
}

# Use the function to plot each confusion matrix
plot_confusion_matrix(nb_confusion, title = "Naive Bayes Confusion Matrix")
```



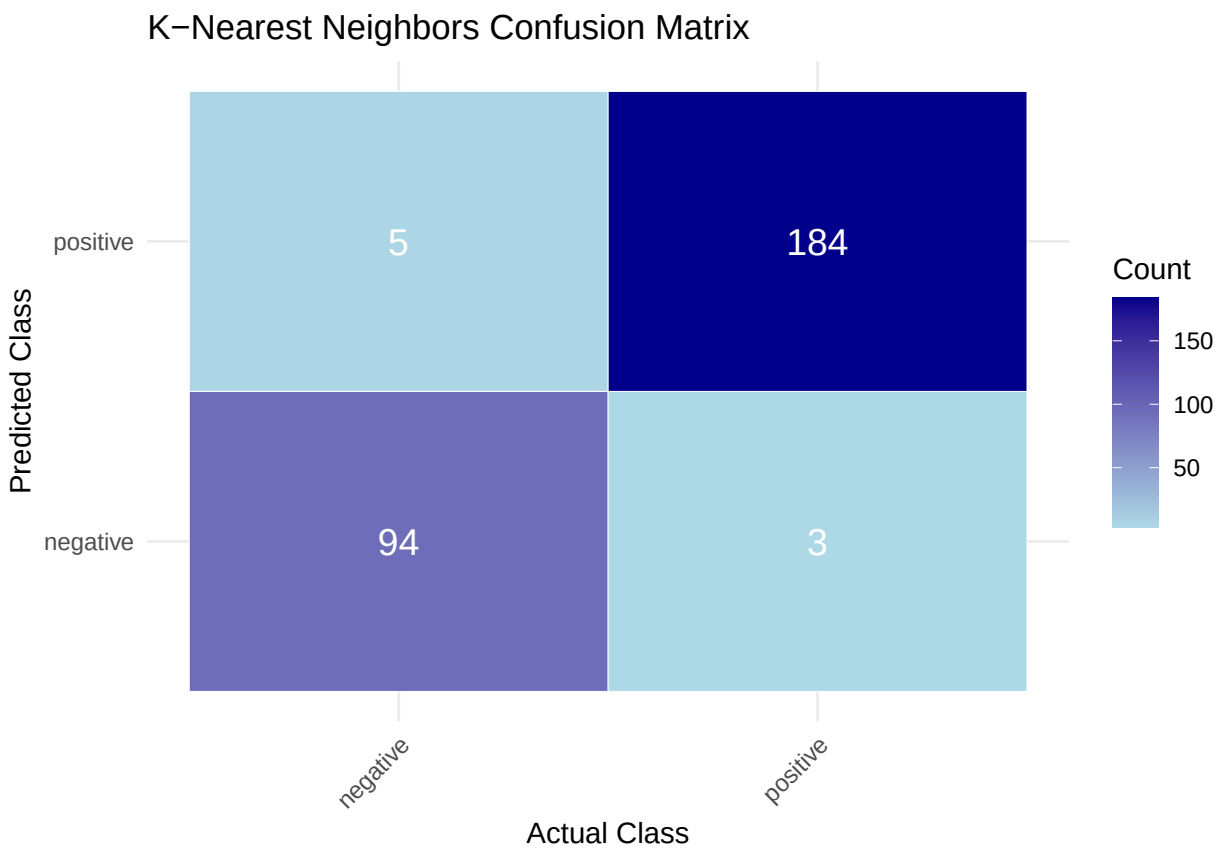

```
plot_confusion_matrix(dt_confusion, title = "Decision Tree Confusion Matrix")
```



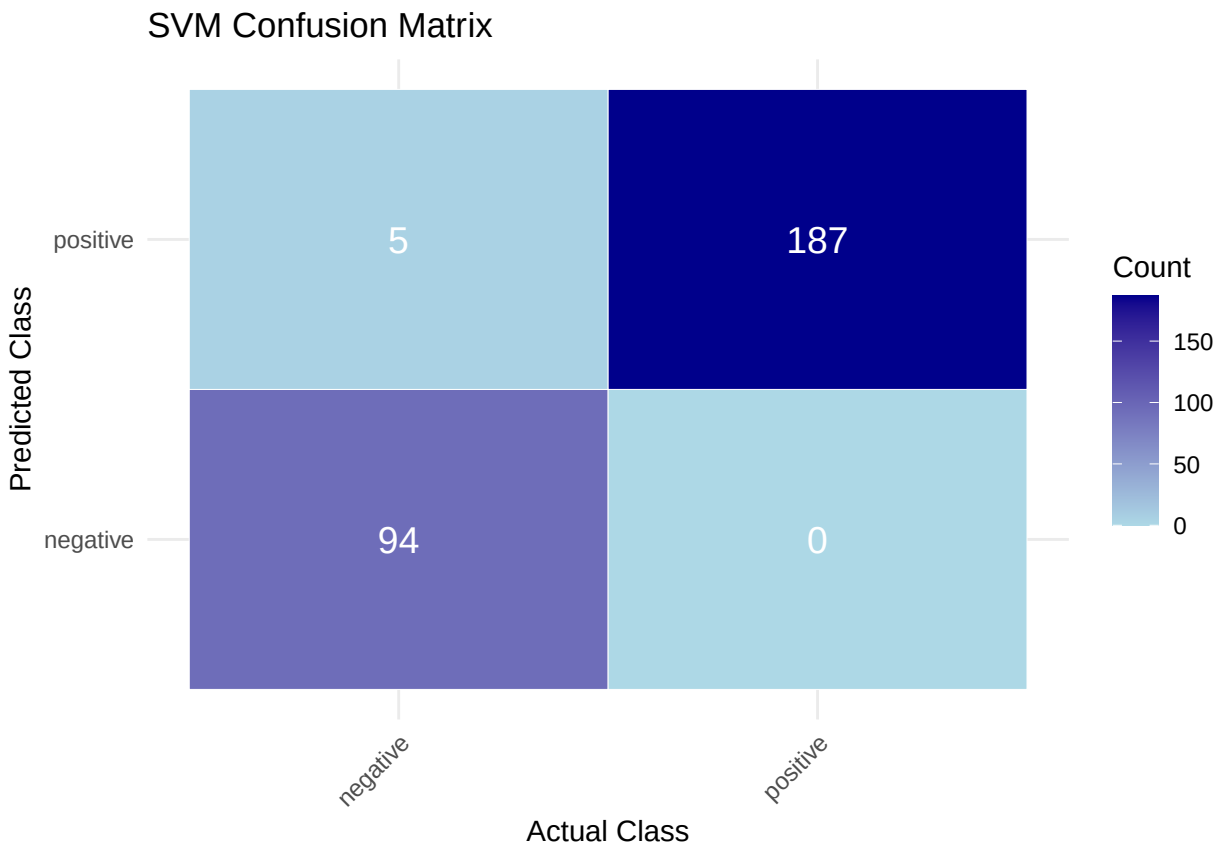

```
plot_confusion_matrix(nn_confusion, title = "Neural Network Confusion Matrix")
```




```
plot_confusion_matrix(knn_confusion, title = "K-Nearest Neighbors Confusion Matrix")
```

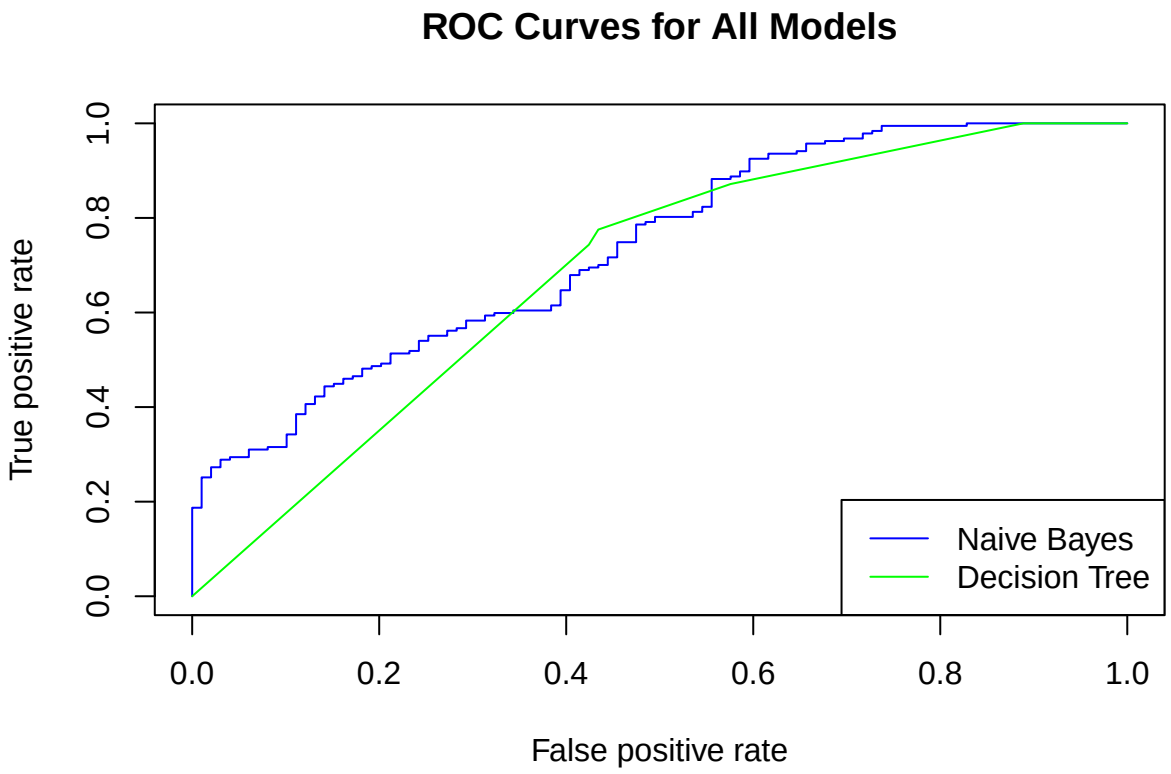



```
plot_confusion_matrix(svm_confusion, title = "SVM Confusion Matrix")
```



Exercice 5

```r
# Naive Bayes ROC Curve
nb_prob <- predict(nb_model, newdata = test_data, type = "prob")[,2]
nb_pred_roc <- prediction(nb_prob, as.numeric(test_data$Class) - 1)  # Convert class to 0/1
nb_perf <- performance(nb_pred_roc, "tpr", "fpr")

# Decision Tree ROC Curve
dt_prob <- predict(dt_model, newdata = test_data, type = "prob")[,2]
dt_pred_roc <- prediction(dt_prob, as.numeric(test_data$Class) - 1)
dt_perf <- performance(dt_pred_roc, "tpr", "fpr")

# Plot ROC Curves
plot(nb_perf, col = "blue", main = "ROC Curves for All Models")
plot(dt_perf, col = "green", add = TRUE)
# Repeat similar lines for each model's ROC curve
legend("bottomright", legend = c("Naive Bayes", "Decision Tree"), col = c("blue", "green"), lty = 1)
```


ROC Curves for All Models
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes
Decision Tree