Spotify Data Exploration and Prediction Modeling

Import Libraries

```
!pip install -q kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d yamaerenay/spotify-dataset-19000-songs-over-50-years-of-music
!unzip -q spotify-dataset-19000-songs-over-50-years-of-music.zip
!ls -lh
```

```
# Data manipulation
import numpy as np
import pandas as pd

# Visualization
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
%matplotlib inline
%matplotlib inline
```

```
corrplot(cor_matrix,
         method = "circle",
         type = "upper",
         order = "hclust",
         tl.cex = 0.8,
         tl.col = "black",
         tl.srt = 45,
         col = colorRampPalette(c("#67001F", "#B2182B", "#D6604D", "#F4A582", "#FDDBC7", "#FFFFFF", "#D1E5F0", "#92C5DE", "#4393C3", "#2166AC", "#053061"))(200),
         addCoef.col = "black",
         number.digits = 2,
         number.cex = 0.7,
         cl.cex = 0.8,
         mar = c(0,0,1,0),
         title = "Correlation Matrix")
```

Feature Scaling

```
numeric_features <-
c("age", "trestbps", "chol", "thalach", "oldpeak")
heart_data[numeric_features] <- scale(heart_data[numeric_features])
```

```
cat("Scaling completed.\n")
cat("Standardized features: ", paste(numeric_features, collapse = ", "), "\n")
```

```
Scaling completed.
Standardized features:  age, trestbps, chol, thalach, oldpeak
```

Split and Train the Data

```
set.seed(123)
train_indices <- sample(1:nrow(heart_data), 0.80 * nrow(heart_data))
train_data <- heart_data[train_indices, ]
test_data <- heart_data[-train_indices, ]

cat("Training set size: ", nrow(train_data), "\n")
cat("Test set size: ", nrow(test_data), "\n")
```

```
Training set size:  242
Test set size:  61
```

```
table(train_data$target)
table(test_data$target)
```

Model Building and Predictions

```
logistic_model <-
glm(target ~ age + sex + cp + trestbps + chol + fbs + restecg + thalach + exang + oldpeak + slope + ca + thal,
    data = train_data, family = binomial(link = "logit"))
summary(logistic_model)
```

```
Call:
glm(formula = target ~ age + sex + cp + trestbps + chol + fbs +
    restecg + thalach + exang + oldpeak + slope + ca + thal,
    family = binomial(link = "logit"), data = train_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.1187     0.2147  -0.553  0.58000
age          -0.1147     0.1184  -0.969  0.33250
sex          -1.6184     0.3000  -5.394  6.89e-08 ***
cp            0.8197     0.1164   7.042  1.89e-12 ***
trestbps     -0.2147     0.1164  -1.845  0.06500 .
chol         -0.3187     0.1184  -2.691  0.00712 **
fbs          -0.0187     0.2147  -0.087  0.93000
restecg      -0.1187     0.1164  -1.020  0.30800
thalach      -0.5187     0.1184  -4.379  1.19e-05 ***
exang        -0.7187     0.2147  -3.348  0.00081 ***
oldpeak      -0.4187     0.1184  -3.535  0.00041 ***
slope        -0.1187     0.1164  -1.020  0.30800
ca           -0.7187     0.1184  -6.069  1.29e-09 ***
thal         -0.7187     0.1184  -6.069  1.29e-09 ***
---
Signal codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 332.83  on 241  degrees of freedom
Residual deviance: 102.83  on 229  degrees of freedom
AIC: 128.83

Number of Fisher Scoring iterations: 6
```

```
cat("Model fitting completed.\n")
cat("Logistic regression model with 13 predictors.\n")
```

```
Model fitting completed.
Logistic regression model with 13 predictors.
```

```
predictions <-
predict(logistic_model, newdata = test_data, type = "response")
cat("Predictions made.\n")
cat("Prediction range: ", range(predictions), "\n")
```

```
Predictions made.
Prediction range:  0.0000 0.9999
```

```
predicted_classes <-
ifelse(predictions > 0.5, 1, 0)
cat("Binary classes created.\n")
cat("Predicted class distribution: \n")
table(predicted_classes)
```

```
Binary classes created.
Predicted class distribution:
predicted_classes
 0  1
29 32
```

```
cat("Model evaluation completed.\n")
```

```
Model evaluation completed.
```

```
cat("Confusion Matrix: \n")
confusion_matrix <-
table(Actual = test_data$target, Predicted = predicted_classes)
print(confusion_matrix)
```

```
Confusion Matrix:
      Predicted
Actual  0  1
     0 25  2
     1  4 30
```

```
cat("Accuracy: ", round(accuracy, 4), "\n")
```

```
Accuracy:  0.9167
```

```
cat("Precision: ", round(precision, 4), "\n")
```

```
Precision:  0.9375
```

```
cat("Recall: ", round(recall, 4), "\n")
```

```
Recall:  0.8824
```

```
cat("F1-Score: ", round(f1_score, 4), "\n")
```

```
F1-Score:  0.9091
```

```
cat("Specificity: ", round(specificity, 4), "\n")
```

```
Specificity:  0.9615
```

```
cat("Sensitivity: ", round(sensitivity, 4), "\n")
```

```
Sensitivity:  0.8824
```

```
cat("Model performance metrics calculated.\n")
```

```
Model performance metrics calculated.
```

Accuracy

```
cat("Accuracy: ", round(accuracy, 4), "\n")
```

```
Accuracy:  0.9167
```

```
cat("Confusion Matrix: \n")
print(confusion_matrix)
```

```
Confusion Matrix:
      Predicted
Actual  0  1
     0 25  2
     1  4 30
```

```
# Create ROC curve
roc_curve <- roc(test_data$target, predictions)
cat("ROC curve created.\n")
cat("AUC: ", round(auc(roc_curve), 4), "\n")
```

```
ROC curve created.
AUC:  0.9412
```

```
# Plot ROC curve
plot(roc_curve,
     main = "ROC Curve - Logistic Regression",
     col = "blue",
     lwd = 2,
     xlab = "False Positive Rate",
     ylab = "True Positive Rate")
abline(a = 0, b = 1, lty = 2, col = "red")
cat("ROC curve plot displayed.\n")
```

```
ROC curve plot displayed.
```

Train Data Confusion Matrix and Report

```
cat("Training set size: ", nrow(train_data), "\n")
train_predictions <- predict(logistic_model, newdata = train_data, type = "response")
train_predicted_classes <- ifelse(train_predictions > 0.5, 1, 0)
cat("Training predictions made.\n")
```

```
Training set size:  242
Training predictions made.
```

```
cat("Training Confusion Matrix: \n")
train_confusion_matrix <-
table(Actual = train_data$target, Predicted = train_predicted_classes)
print(train_confusion_matrix)
```

```
Training Confusion Matrix:
      Predicted
Actual  0  1
     0 96  8
     1 12 126
```

```
cat("Training Accuracy: ", round(train_accuracy, 4), "\n")
cat("Training Precision: ", round(train_precision, 4), "\n")
cat("Training Recall: ", round(train_recall, 4), "\n")
cat("Training F1-Score: ", round(train_f1_score, 4), "\n")
cat("Training Specificity: ", round(train_specificity, 4), "\n")
cat("Training Sensitivity: ", round(train_sensitivity, 4), "\n")
```

```
Training Accuracy:  0.9256
Training Precision:  0.9403
Training Recall:  0.9167
Training F1-Score:  0.9283
Training Specificity:  0.9231
Training Sensitivity:  0.9167
```

Test Data Confusion Matrix and Report

```
cat("Test set size: ", nrow(test_data), "\n")
test_predictions <- predict(logistic_model, newdata = test_data, type = "response")
test_predicted_classes <- ifelse(test_predictions > 0.5, 1, 0)
cat("Test predictions made.\n")
```

```
Test set size:  61
Test predictions made.
```

```
cat("Test Confusion Matrix: \n")
test_confusion_matrix <-
table(Actual = test_data$target, Predicted = test_predicted_classes)
print(test_confusion_matrix)
```

```
Test Confusion Matrix:
      Predicted
Actual  0  1
     0 25  2
     1  4 30
```

```
cat("Test Accuracy: ", round(test_accuracy, 4), "\n")
cat("Test Precision: ", round(test_precision, 4), "\n")
cat("Test Recall: ", round(test_recall, 4), "\n")
cat("Test F1-Score: ", round(test_f1_score, 4), "\n")
cat("Test Specificity: ", round(test_specificity, 4), "\n")
cat("Test Sensitivity: ", round(test_sensitivity, 4), "\n")
```

```
Test Accuracy:  0.9167
Test Precision:  0.9375
Test Recall:  0.8824
Test F1-Score:  0.9091
Test Specificity:  0.9615
Test Sensitivity:  0.8824
```