

# Final Group Project: Updated Checkpoint Submission

Team 8: Nicole Ma; Chenwen Dong; Xulin Wang; Fei Xia; Hanying Qiao

2024-04-10

## 1. Introduction

Per our discussion and in light of new insights and interests, we have decided to switch our research topic from **Bitcoin Analysis** to “**Exploring the Dynamics of Music Popularity and Genre Classification Through Spotify’s Audio Features**”.

This document contains the checkpoint submission for our Final Group Project. This document includes all the code and visualizations used in our analysis.

## 2. Research question

**Research Question:** In our daily lives, we are constantly immersed in a diverse array of music. We are wondering - Can the popularity and genre of a song be accurately predicted based on its audio features using data from the Spotify API? We believe the analysis can offer insights into musical trends, enabling the industry to adapt and innovate proactively.

- Objective 1: Predict song popularity by audio features from Spotify data
- Objective 2: Predict song genres by audio features from Spotify data

**Assumption:** Quantifiable Audio Features: For music related data analysis, one of the biggest problems historically has simply been a way to quantify music, but Spotify has been leading the way detailing the audio features.

## 3. Spotify Dataset

The Spotify Database API provides a comprehensive dataset, encompassing a wide array of audio features for songs. This dataset is ideal for our objectives due to several reasons:

1. **Free API:** The Spotify API is freely accessible to developers.
2. **Diverse Data Points:** Spotify has data of thousands of tracks spanning various genres, offers a rich foundation for our analysis
3. **Reliable Popularity Metric:** Spotify’s popularity score is derived from actual user engagement, offering a tangible metric for song success.

For data downloading part, we are using Python API to access Spotify data, please refer to [Spotify\\_Download\\_API.ipynb](#) for more details.

At the heart of our investigation are two primary dimensions: the popularity of tracks and their genres.

- **Popularity:** The value will be between 0 and 100, with 100 being the most popular. The popularity is calculated by Spotify algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Given the left-skewed distribution, with a vast majority of tracks clustering at the lower end, we define tracks with a popularity score greater than 16 as “popular”.
- **Genre:** Our dataset encompasses approximately 1000 tracks each from six distinct genres (total ~6k track data): Electronic, Hip-hop, Jazz, Pop, Country, and Rock.
- **Other Audio features:**
  1. Acousticness: A measure on a scale from 0 to 1 indicating the acoustic nature of a track.
  2. Danceability: A metric assessing how suitable a track is for dancing.
  3. Energy: Reflects the intensity and activity of a track.
  4. Instrumentalness: Indicates the presence of instrumental sounds in a track.
  5. Liveness: Differentiates live recordings from studio productions.
  6. Loudness: Measures the overall decibel level of a track.
  7. Speechiness: Quantifies the spoken word content in a track.
  8. Tempo: The speed at which a piece of music is played, measured in beats per minute (BPM).
  9. Time Signature: Describes how beats in a song are grouped, contributing to the song’s rhythm.
  10. Key: Identifies the tonal center and harmonic foundation of a track.
  11. Valence: Measures the musical positiveness conveyed by a track.

## 4. Data Loading

### Load Packages

```
library(readr)
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(corrplot)

library(dplyr)
library(tidyr)
library(caret)
library(rpart)

library(randomForest)
library(e1071)
library(nnet)
```

### Load Dataset

```
data <- read.csv("total.csv", stringsAsFactors = FALSE)
```

## 5. Data Pre-processing

Check if there are duplicate Track in our dataset

```
duplicates <- duplicated(data$track_id)
num_duplicates <- sum(duplicates)
cat("Number of duplicates in 'track_id' column:", num_duplicates, "\n")
```

```
## Number of duplicates in 'track_id' column: 0
```

```
if(num_duplicates > 0) {
  duplicate_rows <- data[duplicates, ]
  cat("Duplicate rows based on track_id:\n")
  print(duplicate_rows)
} else {
  cat("No duplicate rows based on track_id found.\n")
}
```

```
## No duplicate rows based on track_id found.
```

Check Missing Values

```
n_df <- nrow(data)

for(col in colnames(data)) {
  missing <- sum(is.na(data[[col]]))
  mis_perc <- (missing / n_df) * 100

  cat(sprintf("The missing percentage of %s is %.2f%%\n", col, mis_perc))
}
```

```
## The missing percentage of Name is 0.00%
## The missing percentage of track_id is 0.01%
## The missing percentage of Album is 0.00%
## The missing percentage of album_id is 0.00%
## The missing percentage of Artist is 0.00%
## The missing percentage of Release_date is 0.00%
## The missing percentage of Length is 0.00%
## The missing percentage of Popularity is 0.00%
## The missing percentage of Acousticness is 0.00%
## The missing percentage of Danceability is 0.00%
## The missing percentage of Energy is 0.00%
## The missing percentage of Instrumentalness is 0.00%
## The missing percentage of Liveness is 0.00%
```

```

## The missing percentage of Loudness is 0.00%
## The missing percentage of Speechiness is 0.00%
## The missing percentage of Tempo is 0.00%
## The missing percentage of Time_signature is 0.00%
## The missing percentage of key is 0.00%
## The missing percentage of valence is 0.00%
## The missing percentage of genre is 0.00%

```

## Feature Distribution

```

df_features = subset(data, select = -c(Album,Release_date,Artist,album_id,Name,track_id,Popularity) )
df_features <- na.omit(df_features)

n_rows <- nrow(df_features)
n_cols <- ncol(df_features)
cat("There are", n_rows, "rows and", n_cols, "columns\n")

## There are 7262 rows and 13 columns

summary_df <- t(summary(df_features))
cols <- colnames(df_features)
print(summary_df)

##          Length      Min.   1st Qu. Median   3rd Qu.    Max. 
##  Acousticness   Min. :0.0000018 1st Qu.:0.0274000 Median :0.1970000 
##  Danceability   Min. :0.00000   1st Qu.:0.50100 Median :0.61700 
##  Energy         Min. :0.00177  1st Qu.:0.42800 Median :0.62600 
##  Instrumentalness Min. :0.000000 1st Qu.:0.000000 Median :0.000233 
##  Liveness        Min. :0.01310  1st Qu.:0.09700 Median :0.12600 
##  Loudness        Min. :-34.149   1st Qu.:-11.552 Median : -8.242 
##  Speechiness     Min. :0.00000  1st Qu.:0.03570 Median :0.04980 
##  Tempo           Min. : 0.000  1st Qu.: 97.780 Median :120.000 
##  Time_signature   Min. :0.000  1st Qu.: 4.000  Median : 4.000 
##  key             Min. : 0.000  1st Qu.: 2.000  Median : 5.000 
##  valence          Min. :0.00000 1st Qu.:0.31900 Median :0.51700 
##  genre            Length:7262   Class :character Mode  :character 
##          Length      Mean   3rd Qu.    Max. 
##  Acousticness   Mean :0.3310017 3rd Qu.:0.6037500 Max. :0.9960000 
##  Danceability   Mean :0.60630   3rd Qu.:0.72400 Max. :0.97000 
##  Energy         Mean :0.59667   3rd Qu.:0.80000 Max. :0.99900 
##  Instrumentalness Mean :0.215055 3rd Qu.:0.378750 Max. :0.984000 
##  Liveness        Mean :0.19540   3rd Qu.:0.24100 Max. :0.99200 
##  Loudness        Mean : -9.317  3rd Qu.:-5.976 Max. : 1.103 
##  Speechiness     Mean :0.09493   3rd Qu.:0.09730 Max. :0.95800 
##  Tempo           Mean :120.010  3rd Qu.:137.020 Max. :249.85 
##  Time_signature   Mean :3.908    3rd Qu.:4.000  Max. : 5.000 
##  key             Mean : 5.356   3rd Qu.: 9.000 Max. :11.000 
##  valence          Mean :0.51310  3rd Qu.:0.71470 Max. :0.98600 
##  genre

```

\*To address our two primary research objectives, we have systematically structured our approach into EDA (visualization), feature selection and modeling. Each phase is conducted separately for both objectives.

## 6. Objective 1: Predict song popularity by audio features from Spotify data

### 6.1. EDA

#### 6.1.1. Predict Variables Distribution

```
df_y=data$Popularity
summary_dfy <- t(summary(df_y))
cols <- colnames(df_y)
print(summary_dfy)

##      Min. 1st Qu. Median  Mean 3rd Qu. Max.
## [1,] 0.00   0.00   3.00 16.59  24.00 96.00

# Calculate the mean of popularity by different genres
mean_by_category <- data %>%
  group_by(genre) %>%
  summarise(mean_value = mean(Popularity, na.rm = TRUE))

# number of rows in each genre
rows <- data %>% count(genre)

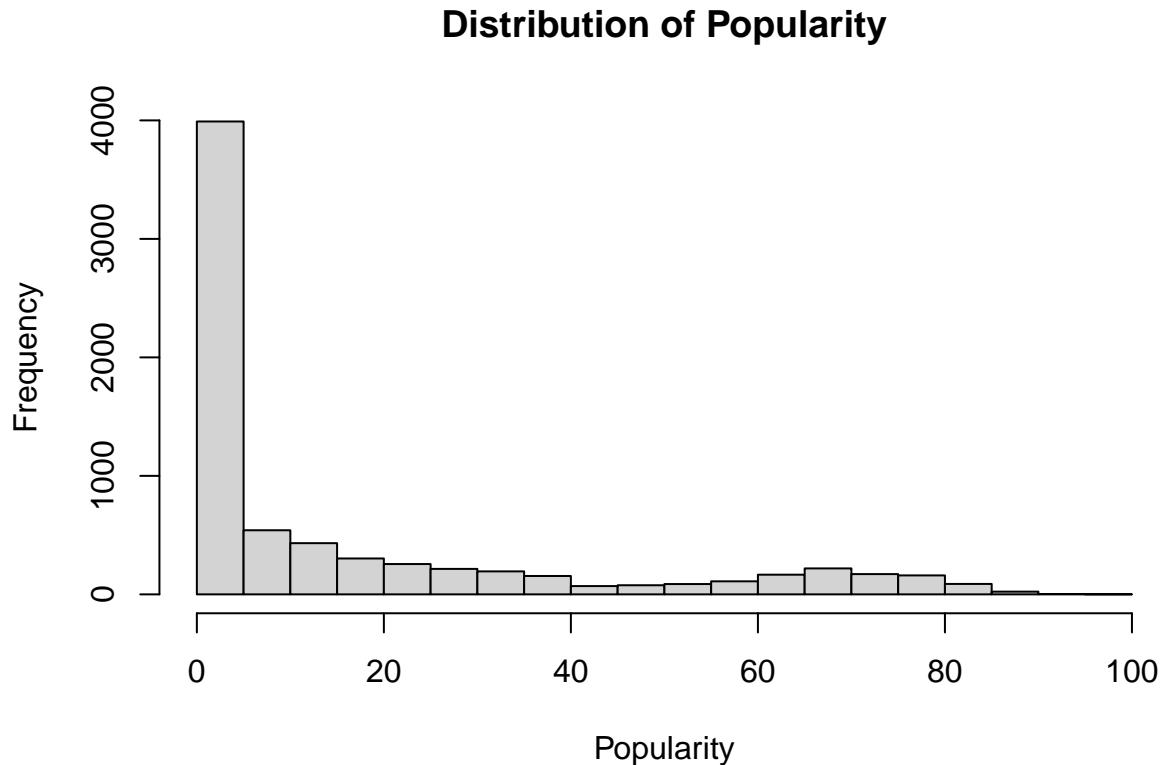
print(mean_by_category)

## # A tibble: 6 x 2
##   genre     mean_value
##   <chr>       <dbl>
## 1 country     11.9
## 2 electronic   12.5
## 3 hiphop      23.5
## 4 jazz        7.52
## 5 pop         21.1
## 6 rock        22.5

print(rows)

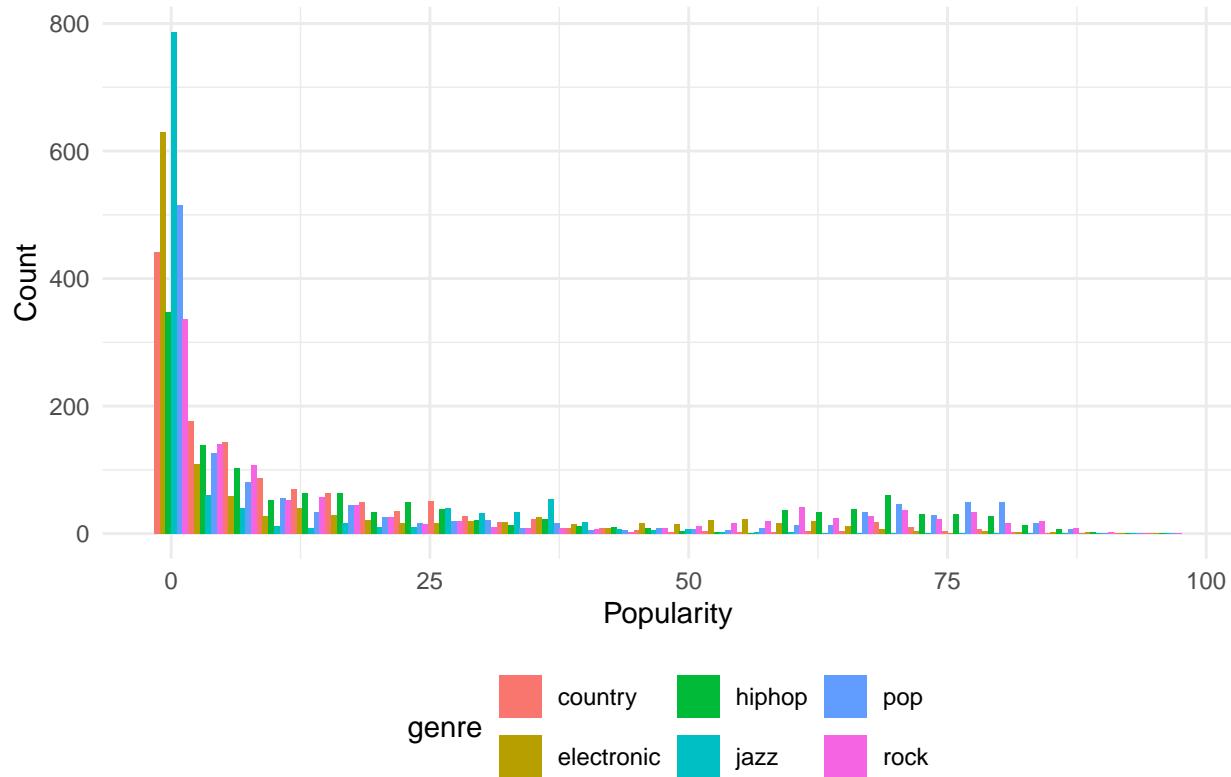
## # A tibble: 6 x 2
##   genre     n
##   <chr>   <dbl>
## 1 country 1274
## 2 electronic 1178
## 3 hiphop 1268
## 4 jazz    1150
## 5 pop     1268
## 6 rock    1124
```

```
# Plot distributions of numerical features  
hist(df_y, main="Distribution of Popularity", xlab="Popularity")
```



```
# Separate histograms for each genre  
# make genre as factor  
data$genre <- as.factor(data$genre)  
  
ggplot(data, aes(x = Popularity, fill = genre)) +  
  geom_histogram(position = "dodge", bins = 30) +  
  theme_minimal() +  
  labs(x = "Popularity", y = "Count", title = "Distribution of Popularity by Genre") +  
  theme(legend.position = "bottom")
```

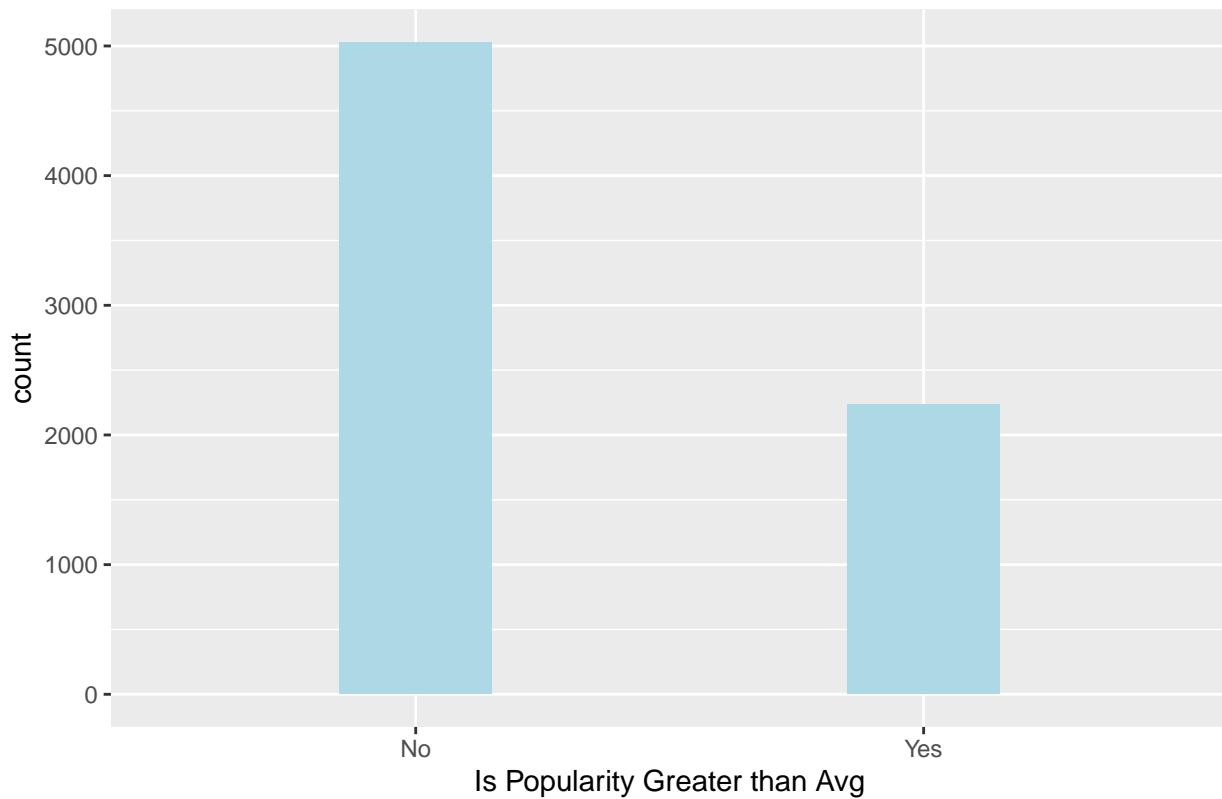
## Distribution of Popularity by Genre



```
ggplot(data = data) +stat_count ( aes(x= ifelse(data$Popularity > 16, "Yes", "No")) , fill = "lightblue")  
  labs(title = "Distribution of Popularity Greater than Avg",x = "Is Popularity Greater than Avg")
```

```
## Warning: Use of 'data$Popularity' is discouraged.  
## i Use 'Popularity' instead.
```

Distribution of Popularity Greater than Avg



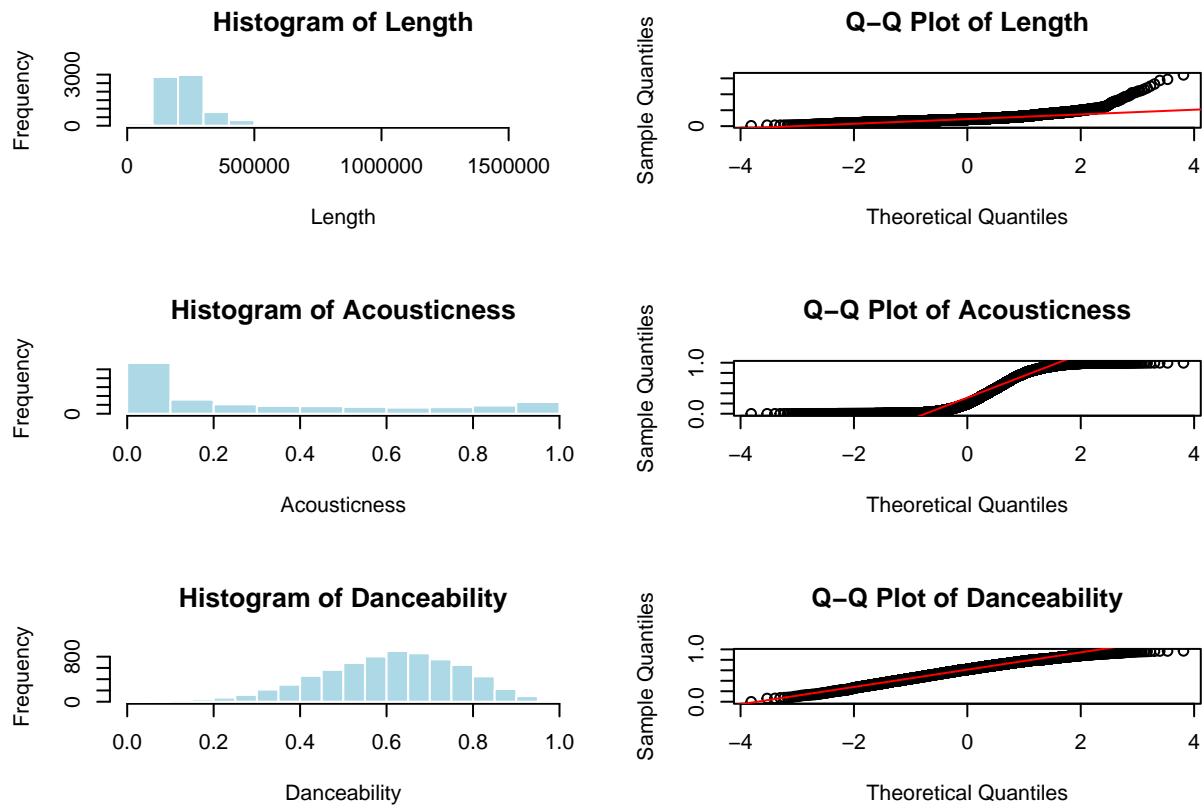
#### 6.1.2. Feature Distribution Visualization

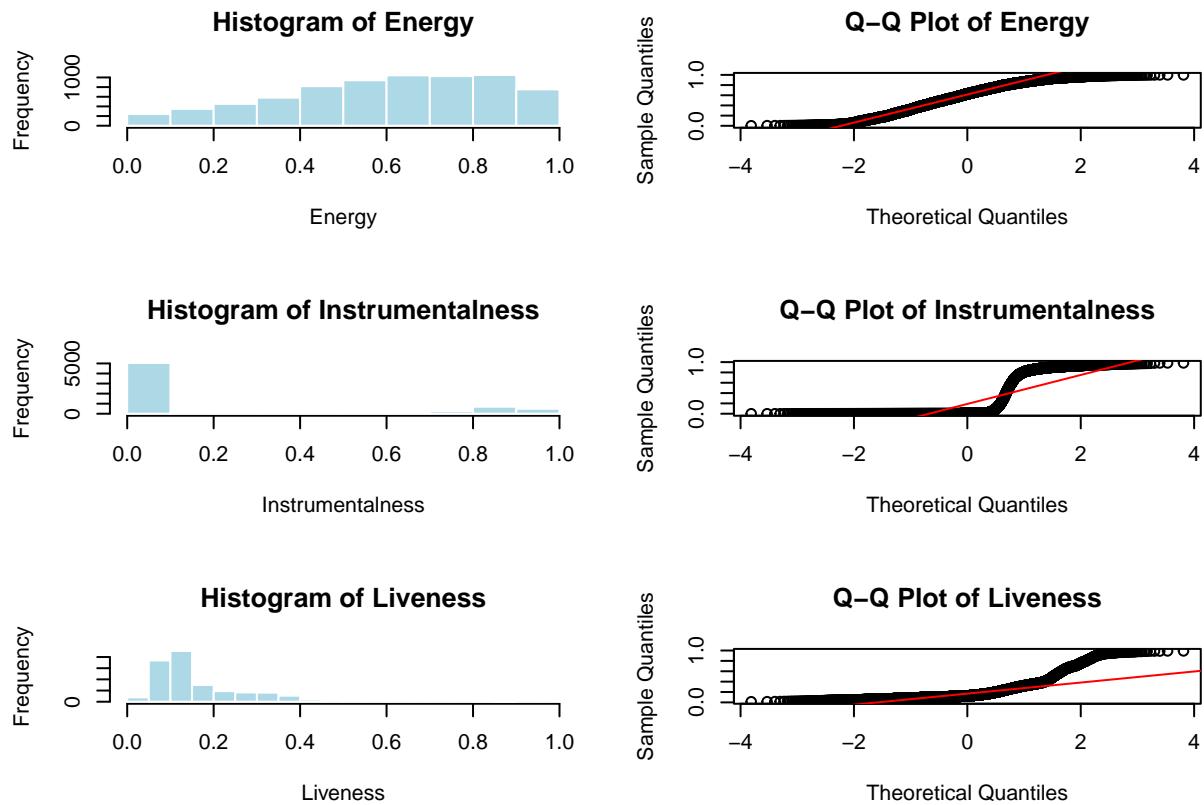
```
# Filter cols to include only numeric columns to avoid errors in plotting
numeric_cols <- sapply(df_features, is.numeric)
cols <- names(df_features)[numeric_cols]

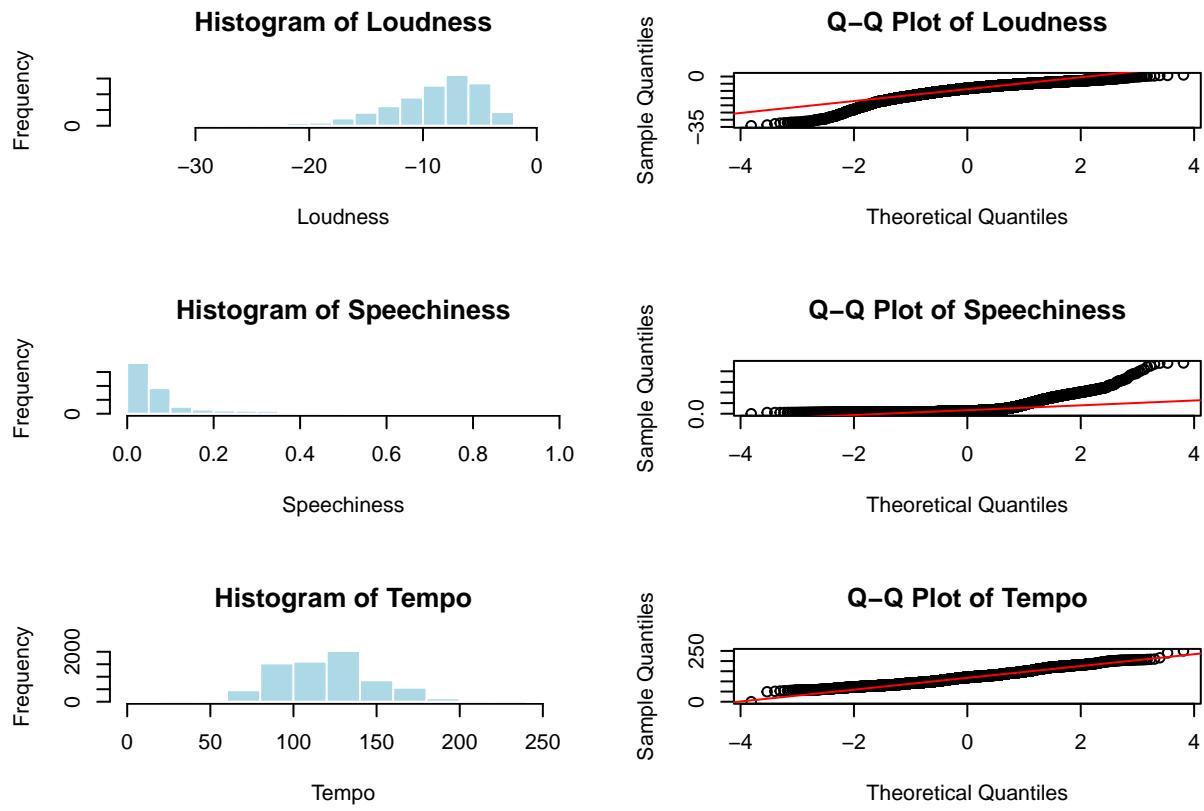
par(mfrow = c(3, 2))

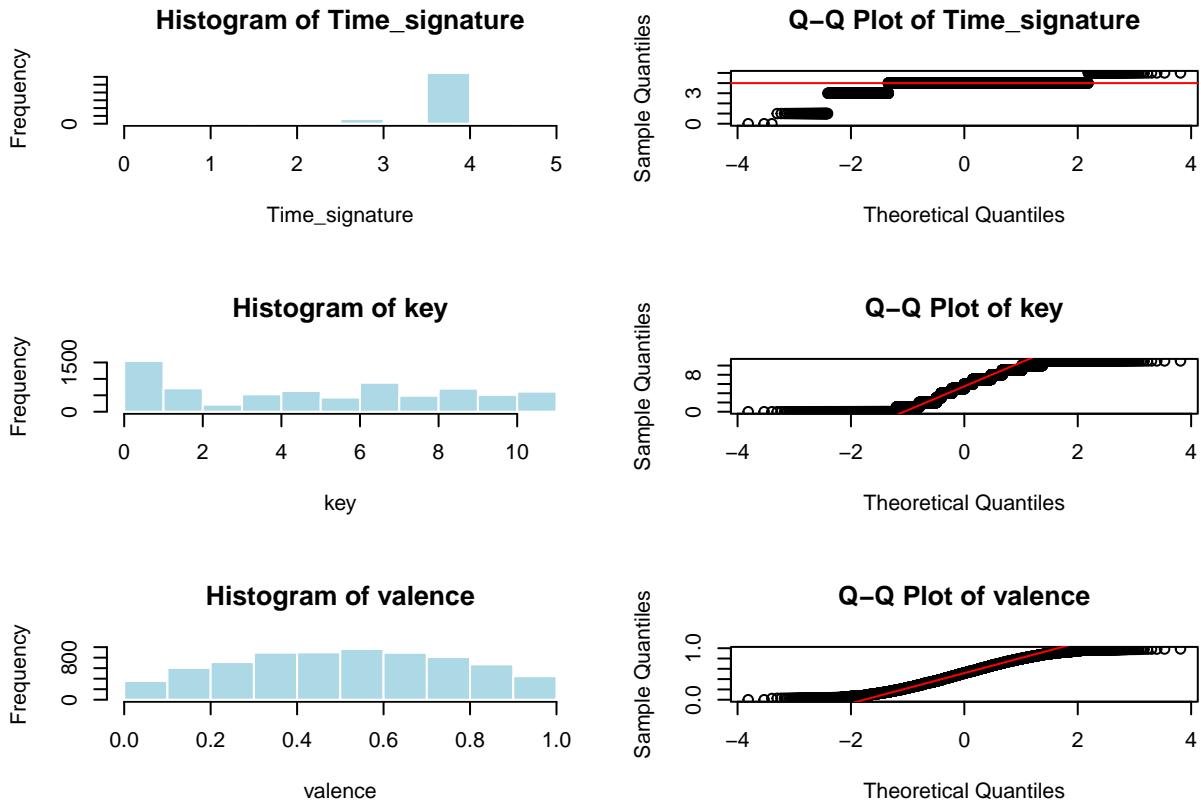
# Loop through the numeric columns and create plots
for (col in cols) {
  # Histogram plot
  hist(df_features[[col]], main = paste("Histogram of", col), xlab = col, col = "lightblue", border = "red")

  # Q-Q plot
  qqnorm(df_features[[col]], main = paste("Q-Q Plot of", col))
  qqline(df_features[[col]], col = "red")
}
```



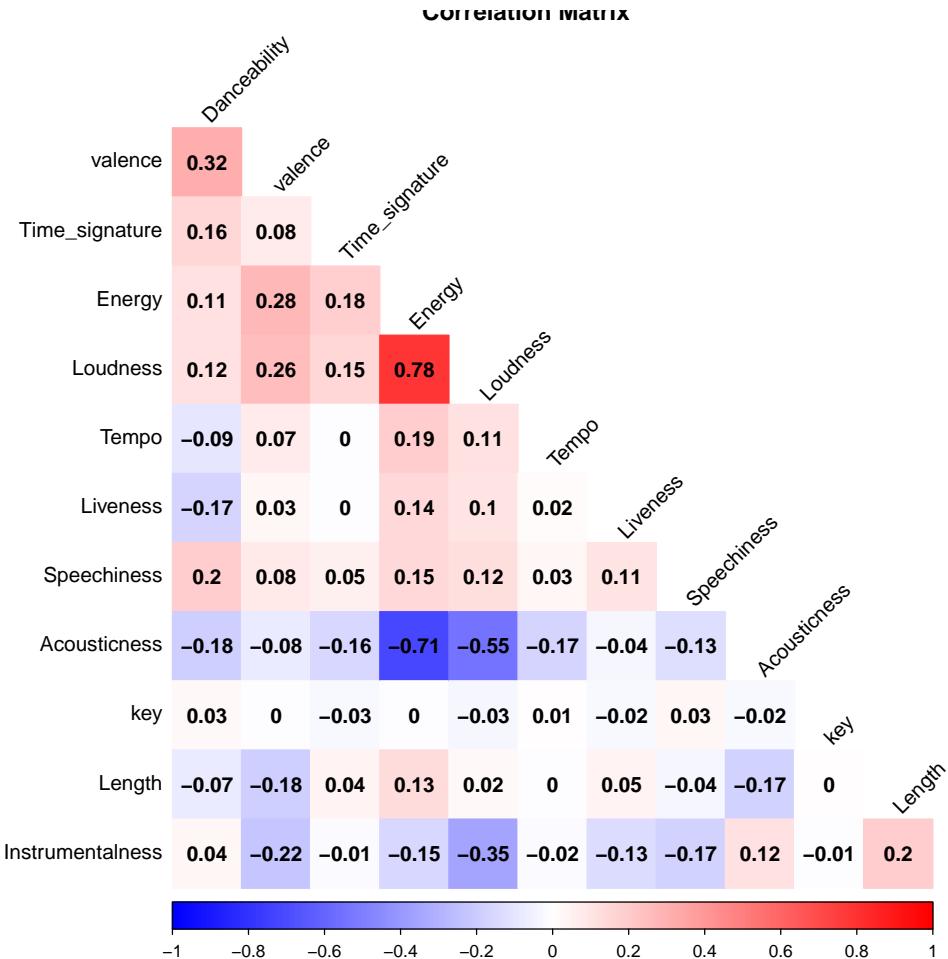






### 6.1.3. Check Correlation

```
library(corrplot)
numeric_df <- df_features[sapply(df_features, is.numeric)]
correlation_mat <- cor(numeric_df, use = "complete.obs")
corrplot(correlation_mat, method = "color", type = "lower", # Focusing on the lower triangle
          order = "hclust",
          addCoef.col = "black",
          tl.col = "black", tl.srt = 45,
          diag = FALSE,
          na.label = "",
          col = colorRampPalette(c("blue", "white", "red"))(200),
          title = "Correlation Matrix"
        )
```



According to the correlation diagram, we found that the energy column has the high correlation with other variables, and we decide to drop it.

```
#drop energy
numeric_df <- numeric_df[, !names(numeric_df) %in% c("Energy")]
```

## 6.2 Regression: Feature Selection

```
library(caret)

set.seed(123)

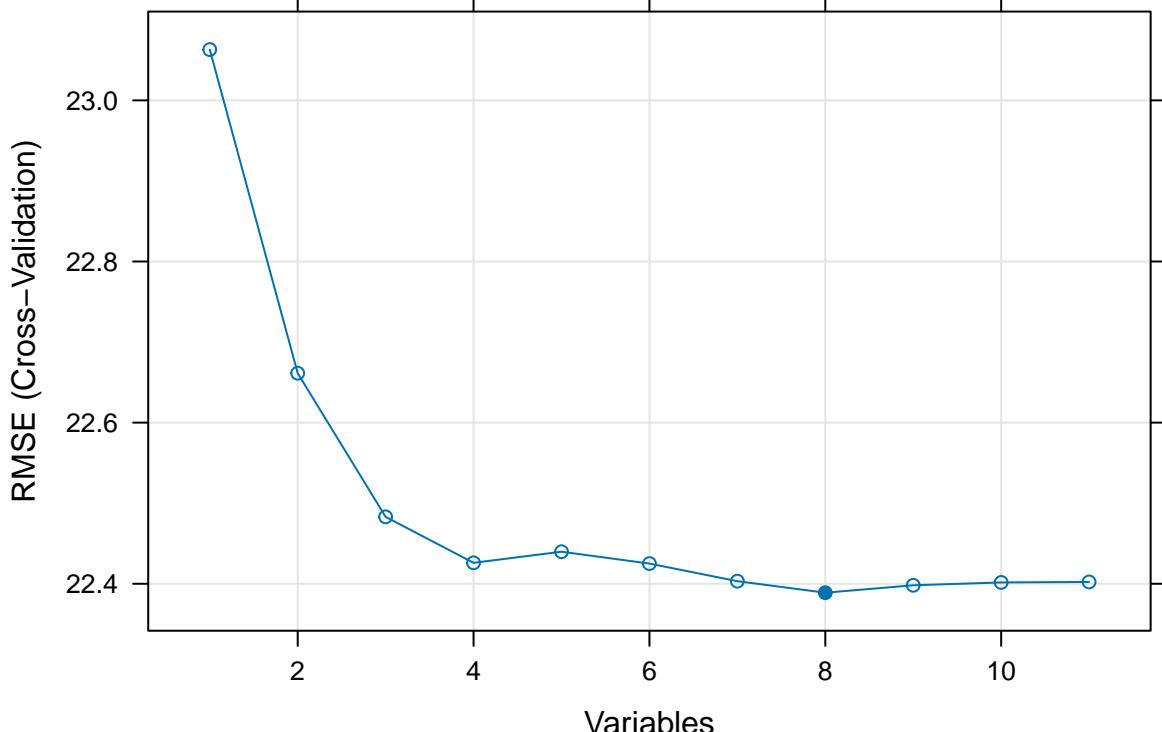
X_scaled <- as.data.frame(scale(numeric_df))
y <- df_y

scale <- data.frame(X_scaled, Popularity = y)

control <- rfeControl(functions=lmFuncs, method="cv", number=10)

results <- rfe(x=scale[, -ncol(scale)], y=scale$Popularity, sizes=1:ncol(scale[, -ncol(scale)]), rfeCon
```

```
plot(results, type=c("o", "g"))
```



Select the best Variables

```
# The best subset of variables is in
bestVars <- results$optVariables
print(bestVars)

## [1] "Acousticness"      "Instrumentalness" "Loudness"          "Danceability"
## [5] "Liveness"          "valence"           "Length"            "Time_signature"
```

### 6.3 Regression: Build Model

```
finalModel <- lm(Popularity ~ ., data=data[, c(bestVars, "Popularity")])
# Summary of the final model
summary(finalModel)
```

```
##
## Call:
## lm(formula = Popularity ~ ., data = data[, c(bestVars, "Popularity")])
##
## Residuals:
```

```

##      Min     1Q   Median     3Q    Max
## -30.267 -16.036 -7.043  7.825 74.447
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.261e+01 2.971e+00 7.610 3.08e-14 ***
## Acousticness -1.397e+01 9.756e-01 -14.323 < 2e-16 ***
## Instrumentalness -1.039e+01 8.415e-01 -12.350 < 2e-16 ***
## Loudness 7.268e-01 7.095e-02 10.244 < 2e-16 ***
## Danceability 1.004e+01 1.826e+00 5.501 3.91e-08 ***
## Liveness -5.334e+00 1.592e+00 -3.350 0.000813 ***
## valence -3.409e+00 1.180e+00 -2.889 0.003874 **
## Length -7.492e-06 2.603e-06 -2.878 0.004009 **
## Time_signature 1.556e+00 6.679e-01 2.330 0.019822 *
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.39 on 7253 degrees of freedom
## Multiple R-squared: 0.1423, Adjusted R-squared: 0.1413
## F-statistic: 150.4 on 8 and 7253 DF, p-value: < 2.2e-16

```

Our first linear regression model. The minimum residual is -30.267, and the maximum is 74.447. The Adjusted R-squared is really low. This model does not have a high predictive power, an alternative models is need for research.

## 6.4 Classification: Feature Selection

Our Classification model mutates popularity base on its average. If a popularity value is greater than 16(the average of the variable), it returns a 1. If it is smaller than 16, it returns a 0.

**Add if\_popularity, should return 1 and 0s**

```

df_y2 <- as.data.frame(as.factor(ifelse(data$Popularity > 16, 1, 0)))
names(df_y2)[names(df_y2) == "as.factor(ifelse(data$Popularity > 16, 1, 0))"] <- "popularity_01"
df_features2=subset(df_features, select = -c(genre))
df_model <- cbind(df_y2, df_features2)

```

Before building models, we identify the most relevant features for predicting our target variable. This will take roughly 10min to run.

```

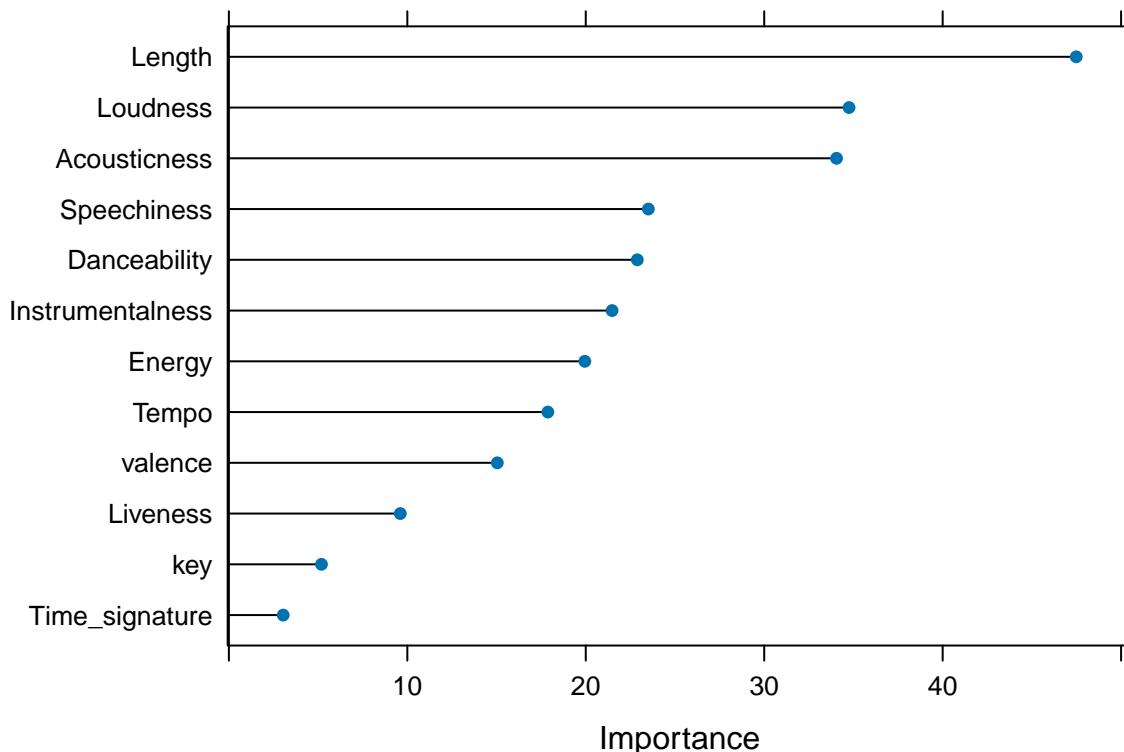
library(caret)
set.seed(123)

# Train a Random Forest model to assess feature importance
control = trainControl(method = "cv", number = 5)
model <- train(popularity_01~., data=df_model, method="rf", importance=TRUE, trControl = control)

# Plot the variable importance
# Extracting variable importance
importance <- varImp(model, scale=FALSE)

# Plotting variable importance
plot(importance)

```



This chunk runs on High CPU occupation

```

library(randomForest)
library(Boruta)
# Run Boruta
set.seed(123) # For reproducibility
boruta_output <- Boruta(popularity_01 ~ ., data = df_model, doTrace = 0)

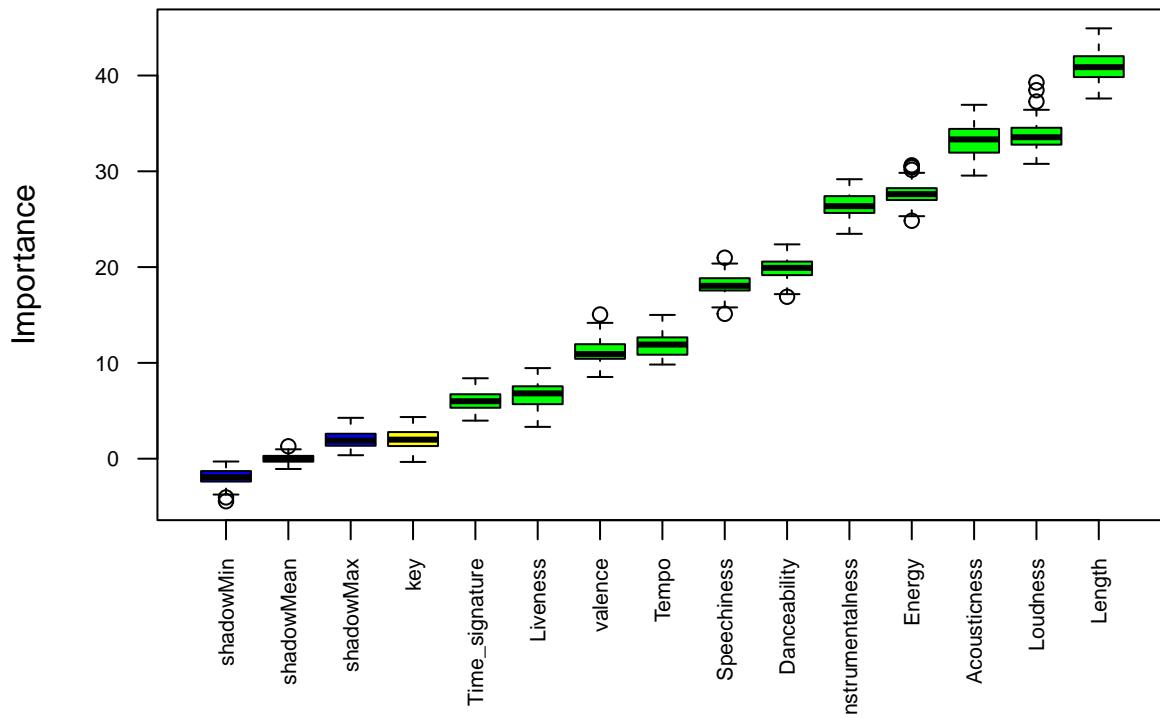
# Print the results
print(boruta_output)

## Boruta performed 99 iterations in 1.51503 mins.
## 11 attributes confirmed important: Acousticness, Danceability, Energy,
## Instrumentalness, Length and 6 more;
## No attributes deemed unimportant.
## 1 tentative attributes left: key;

# Plot the results for visualization
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")

```

## Variable Importance



```
# Get the final decision of which variables were confirmed as important
final_vars <- getSelectedAttributes(boruta_output, withTentative = FALSE)
```

List all the important variables

```
final_vars_with_target <- c("popularity_01", final_vars)
selected_features_df <- df_model[, final_vars_with_target]
print(final_vars)
```

```
## [1] "Length"           "Acousticness"      "Danceability"      "Energy"
## [5] "Instrumentalness" "Liveness"          "Loudness"          "Speechiness"
## [9] "Tempo"            "Time_signature"    "valence"
```

### 6.5 Classification: Build Model

Describe the models you have built, including any assumptions or choices made in the process.

#### 6.5.1. Model 1: Random Forest

```
# Split the data into training and testing sets
index <- createDataPartition(selected_features_df$popularity_01, p=0.8, list=FALSE)
trainData <- selected_features_df[index, ]
```

```

 testData <- selected_features_df[-index, ]

# Model 1: Random Forest
rfModel <- randomForest(popularity_01 ~ ., data=trainData)
rfPredictions <- predict(rfModel, newdata=testData)
rfConfMatrix <- confusionMatrix(rfPredictions, testData$popularity_01)
print("Random Forest Model Results:")

## [1] "Random Forest Model Results:"

print(rfConfMatrix)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction 0 1
##          0 913 294
##          1  92 152
##
##          Accuracy : 0.734
##             95% CI : (0.7104, 0.7566)
##    No Information Rate : 0.6926
##    P-Value [Acc > NIR] : 0.0003051
##
##          Kappa : 0.2852
##
## McNemar's Test P-Value : < 2.2e-16
##
##          Sensitivity : 0.9085
##          Specificity : 0.3408
##    Pos Pred Value : 0.7564
##    Neg Pred Value : 0.6230
##          Prevalence : 0.6926
##          Detection Rate : 0.6292
##    Detection Prevalence : 0.8318
##          Balanced Accuracy : 0.6246
##
##          'Positive' Class : 0
##

```

Overall, the model demonstrates reasonably good performance in terms of overall accuracy, but there are notable differences in performance between sensitivity and specificity, indicating potential future direction of analysis. This also result the balanced accuracy to be relatively low compare to the accuracy.

### 6.5.2. Model 2: Generalized Linear Model

```

index <- createDataPartition(df_model$popularity_01, p=0.8, list=FALSE)
trainData <- df_model[index, ]
testData <- df_model[-index, ]

```

```

glmModel <- glm(popularity_01 ~ ., data=trainData, family="binomial")
glmPredictions <- predict(glmModel, newdata=testData, type="response")
glmPredictionsClass <- ifelse(glmPredictions > 0.5, 1, 0)
glmConfMatrix <- confusionMatrix(as.factor(glmPredictionsClass), testData$popularity_01)
print("Generalized Linear Model Results:")

## [1] "Generalized Linear Model Results:

print(glmConfMatrix)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0     1
##           0 916 351
##           1   89  95
##
##                  Accuracy : 0.6968
##                  95% CI : (0.6724, 0.7203)
##      No Information Rate : 0.6926
##      P-Value [Acc > NIR] : 0.3784
##
##                  Kappa : 0.1487
##
##      Mcnemar's Test P-Value : <2e-16
##
##                  Sensitivity : 0.9114
##                  Specificity : 0.2130
##      Pos Pred Value : 0.7230
##      Neg Pred Value : 0.5163
##                  Prevalence : 0.6926
##      Detection Rate : 0.6313
##      Detection Prevalence : 0.8732
##      Balanced Accuracy : 0.5622
##
##      'Positive' Class : 0
##

```

the GLM model exhibits similar patterns to the Random Forest model, with relatively good sensitivity but lower specificity which indicates a tendency to correctly identify positive instances but struggles with accurately identifying negatives.

## 6.6 Model Selection

```

# Random Forest Metrics
rfAccuracy <- rfConfMatrix$overall['Accuracy']
rfSensitivity <- rfConfMatrix$byClass['Sensitivity']
rfSpecificity <- rfConfMatrix$byClass['Specificity']
rfBalanced = rfConfMatrix$byClass['Balanced Accuracy']
# GLM Metrics

```

```

glmAccuracy <- glmConfMatrix$overall['Accuracy']
glmSensitivity <- glmConfMatrix$byClass['Sensitivity']
glmSpecificity <- glmConfMatrix$byClass['Specificity']
glmBalanced = glmConfMatrix$byClass['Balanced Accuracy']
# Print the metrics
print(paste("Random Forest Accuracy:", rfAccuracy))

## [1] "Random Forest Accuracy: 0.733976567884218"

print(paste("Random Forest Sensitivity:", rfSensitivity))

## [1] "Random Forest Sensitivity: 0.908457711442786"

print(paste("Random Forest Specificity:", rfSpecificity))

## [1] "Random Forest Specificity: 0.340807174887892"

print(paste("Random Forest Balanced Accuracy:", rfBalanced))

## [1] "Random Forest Balanced Accuracy: 0.624632443165339"

print(paste("GLM Accuracy:", glmAccuracy))

## [1] "GLM Accuracy: 0.696760854583046"

print(paste("GLM Sensitivity:", glmSensitivity))

## [1] "GLM Sensitivity: 0.911442786069652"

print(paste("GLM Specificity:", glmSpecificity))

## [1] "GLM Specificity: 0.213004484304933"

print(paste("GLM Balanced Accuracy:", glmBalanced))

## [1] "GLM Balanced Accuracy: 0.562223635187292"

library(pROC)

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

##
## Attaching package: 'pROC'

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

```

```

# Random Forest AUC
rfProbabilities <- predict(rfModel, newdata=testData, type="prob")
rfROC <- roc(response=testData$popularity_01, predictor=rfProbabilities[,2])

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

rfAUC <- auc(rfROC)

# GLM AUC
glmProbabilities <- predict(glmModel, newdata=testData, type="response")
glmROC <- roc(response=testData$popularity_01, predictor=glmProbabilities)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

glmAUC <- auc(glmROC)

# Print AUC values
print(paste("Random Forest AUC:", rfAUC))

## [1] "Random Forest AUC: 0.986049572763983"

print(paste("GLM AUC:", glmAUC))

## [1] "GLM AUC: 0.699915222095799"

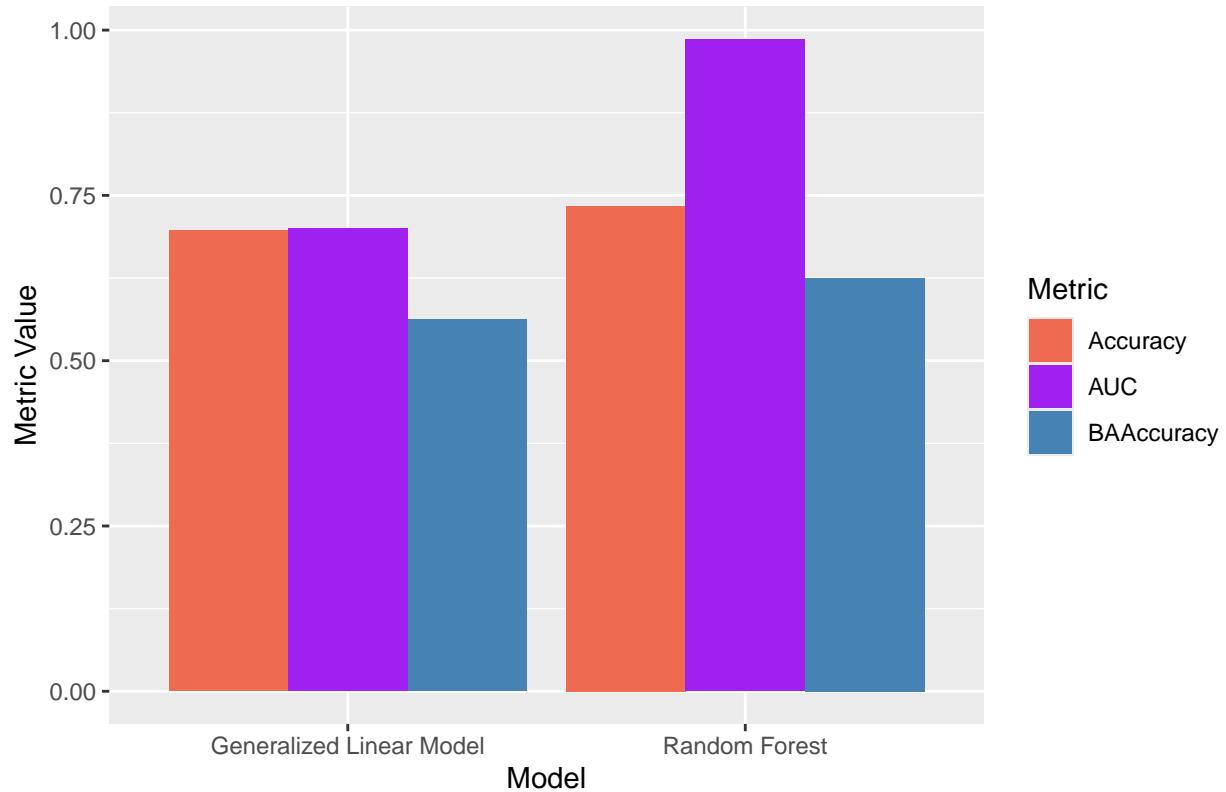
print(glmROC)

## 
## Call:
## roc.default(response = testData$popularity_01, predictor = glmProbabilities)
## 
## Data: glmProbabilities in 1005 controls (testData$popularity_01 0) < 446 cases (testData$popularity_01 1)
## Area under the curve: 0.6999

data_for_plot = data.frame(Model = c("Random Forest", "Generalized Linear Model"),
                           Accuracy = c(rfAccuracy, glmAccuracy), BAAccuracy = c(rfBalanced, glmBalanced), AUC = c(auc(rfROC), auc(glmROC)),
                           df_long <- pivot_longer(data_for_plot, cols = c(Accuracy, BAAccuracy, AUC), names_to = "Metric", values_to = "Value"))
ggplot(df_long, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Model", y = "Metric Value", title = "Model Performance Metrics") +
  scale_fill_manual(values = c("Accuracy" = "coral2", "BAAccuracy" = "steelblue", "AUC" = "purple"))

```

## Model Performance Metrics



```
# Summary for GLM model
summary(glmModel)
```

```
##
## Call:
## glm(formula = popularity_01 ~ ., family = "binomial", data = trainData)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.186e+00  5.049e-01 -2.348  0.018852 *
## Length                -7.157e-07  3.465e-07 -2.065  0.038906 *
## Acousticness        -1.209e+00  1.450e-01 -8.338 < 2e-16 ***
## Danceability         1.011e+00  2.241e-01  4.511 6.45e-06 ***
## Energy               2.404e-01  2.604e-01  0.923 0.355922
## Instrumentalness   -9.251e-01  1.173e-01 -7.887 3.11e-15 ***
## Liveness              -5.969e-01  1.940e-01 -3.077 0.002089 **
## Loudness             9.478e-02  1.340e-02  7.071 1.54e-12 ***
## Speechiness          -2.926e-01  2.959e-01 -0.989 0.322824
## Tempo                -7.006e-04  1.145e-03 -0.612 0.540767
## Time_signature       3.864e-01  1.009e-01  3.828 0.000129 ***
## key                  5.710e-03  8.470e-03  0.674 0.500180
## valence              -4.476e-01  1.428e-01 -3.135 0.001721 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```

## Null deviance: 7173.6 on 5810 degrees of freedom
## Residual deviance: 6415.5 on 5798 degrees of freedom
## AIC: 6441.5
##
## Number of Fisher Scoring iterations: 5

```

Length, Acousticness, Danceability, Instrumentalness, Liveness, Loudness, Time\_Signature and valence are important predictors of the outcome variable.

## 6.7 Other features

```

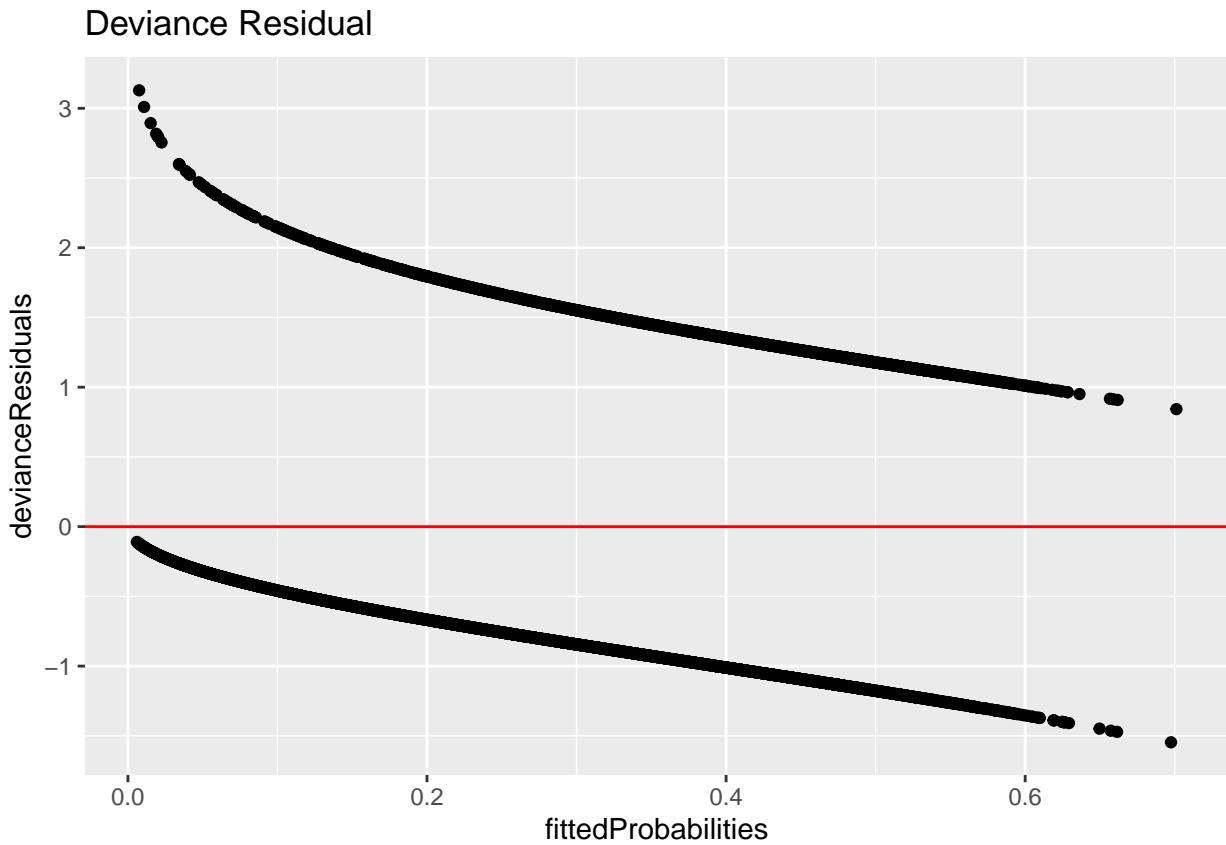
# Assuming logistic regression model

# Calculating fitted probabilities
fittedProbabilities <- predict(glmModel, type = "response")
# Extracting deviance residuals
devianceResiduals <- residuals(glmModel, type = "deviance")

# Plotting deviance residuals against fitted probabilities
ggplot() +geom_point(aes( x = fittedProbabilities, y = devianceResiduals)) + geom_abline(h= 0,slope = 0)

## Warning in geom_abline(h = 0, slope = 0, col = "red"): Ignoring unknown
## parameters: 'h'

```

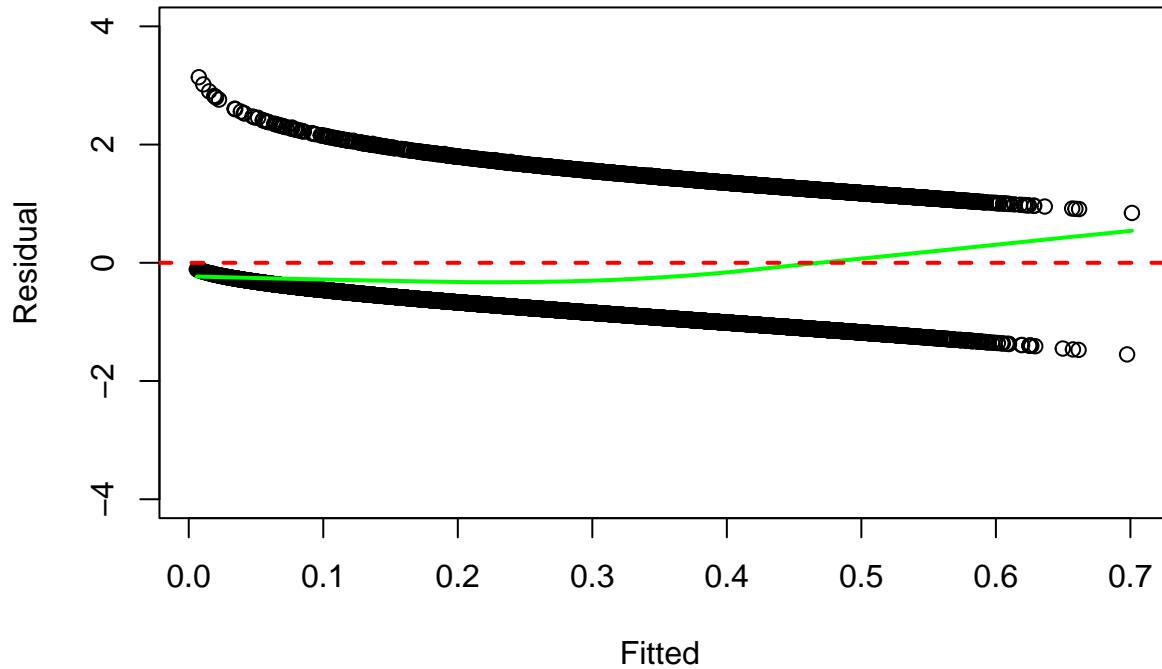


This plot can reveal patterns in the residuals, indicating potential issues with the model. We can see that as the fitted probability grows, the model has a lower positive residual, which means the model is better on the assignment on predicting positive class (1). But when as the fitted probability grows, the negative residual also grows. Indicating the model is not doing well on predicting negative class (0).

### Other residuals

```
rjack <- rstudent(glmModel)
# Jackknife Residuals
plot(fittedProbabilities, rjack,
      xlab = "Fitted ", ylab = "Residual",
      main = "Jackknife Residual",
      ylim = c(-4,4))
lines(lowess(fittedProbabilities, rjack), col = "green", lwd = 2)
abline(h = 0, col = "red", lty = "dashed", lwd = 2)
```

## Jackknife Residual

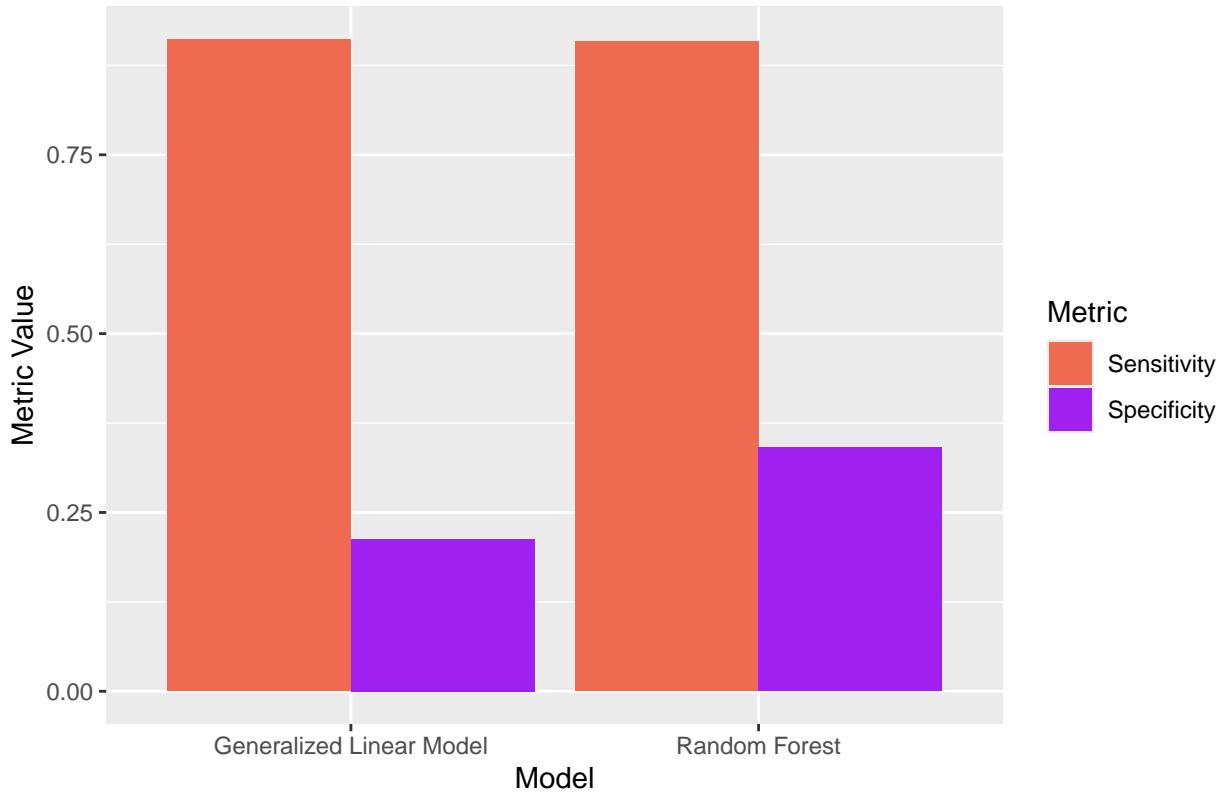


## 6.8 Results Analysis

Discuss the results of your models, including any important metrics and how they compare to your objectives.

```
model_metrics <- data.frame(
  Model = c(rep("Random Forest", 2), rep("Generalized Linear Model", 2)),
  Metric = c(rep(c("Sensitivity", "Specificity"), 2)),
  Value = c(rfSensitivity, rfSpecificity, glmSensitivity, glmSpecificity))
ggplot(model_metrics, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = " Sensitivity vs. Specificity", y = "Metric Value")+
  scale_fill_manual(values = c("Sensitivity" = "coral2", "Specificity" = "purple"))
```

## Sensitivity vs. Specificity



Our model has a high Sensitivity but a low specificity. It indicates that the model is good at correctly identifying positive cases but tends to incorrectly classify negative cases as positive. Apply to our case, the model is good at predicting popular songs but it's falsely identifying less popular songs with high popularity. The model has a decent accuracy and AUC which indicates certain strengths of it. We need to further refine the model to achieve a better balance between sensitivity and specificity. This could involve adjusting the model's threshold, feature selection, or gathering more balanced datasets. This also makes sense because less popular song are indeed more prevalent than hit songs. For important variables: Length, Acousticness, Danceability, Instrumentalness, Liveness, Loudness, Time\_Signature, Some of them are very intuitive; Song with more danceability and liveness indicate a certain feature of Pop and electronic songs, and popularity is literally the definition of pop songs. Loudness and liveness are the key feature of Rock and hip-hop songs. Most importantly, Length is a key feature of all songs in the streaming age. A song with longer duration tends to have less streams (fast paced world).

A reasonable doubt: we have many 0s and low popularity values in the dataset. In fact, the dataset is not balanced, and the number of tracks with popularity values smaller than the average is much higher than the number of tracks with higher popularity. Class imbalance occurs when one class (in this case, tracks with lower popularity) is significantly more prevalent than another class.

## 7. Objective 2: Predict song genres by audio features from Spotify data

```
# for the year column, mostly of raw data is in the format of m/d/yyyy, however, some tracks have only
# to make it consistent, we only extract the release year
data$Release_year <- sub(".*(\\d{4})$", "\\\\1", data$Release_date)
```

```

data$Release_year <- as.numeric(data$Release_year)
#make genre a factor
genre_data = data[, 7:21]
genre_data$genre <- as.factor(genre_data$genre)
genre_data = select(genre_data, -Popularity)
str(genre_data)

## 'data.frame':    7262 obs. of  14 variables:
##   $ Length      : num  208306 197411 253906 131999 179123 ...
##   $ Acousticness: num  0.113 0.0965 0.417 0.0642 0.383 0.153 0.0305 0.151 0.0106 0.521 ...
##   $ Danceability: num  0.762 0.458 0.637 0.892 0.505 0.377 0.676 0.621 0.783 0.846 ...
##   $ Energy       : num  0.692 0.561 0.69 0.503 0.694 0.584 0.926 0.792 0.636 0.652 ...
##   $ Instrumentalness: num  0.00 0.00 0.00 0.00 0.00 0.00 6.58e-04 1.83e-06 0.00 2.33e-04 ...
##   $ Liveness     : num  0.094 0.156 0.211 0.102 0.16 0.0662 0.0771 0.077 0.283 0.0814 ...
##   $ Loudness     : num  -3.97 -4.82 -3.21 -7.09 -5.42 ...
##   $ Speechiness  : num  0.0438 0.0269 0.0693 0.178 0.0295 0.0495 0.0425 0.037 0.0614 0.0333 ...
##   $ Tempo        : num  115 150 140 140 154 ...
##   $ Time_signature: int  4 3 4 4 4 4 4 4 4 ...
##   $ key          : int  5 10 9 6 2 2 0 0 7 7 ...
##   $ valence      : num  0.397 0.206 0.103 0.615 0.406 0.444 0.818 0.375 0.745 0.888 ...
##   $ genre         : Factor w/ 6 levels "country","electronic",...: 5 5 5 5 5 5 5 5 5 5 ...
##   $ Release_year  : num  2008 2018 2009 2023 2023 ...

```

## 7.1 EDA

Firstly, I try to visualize the distribution of audio features across different music genres using **box plots**.

```

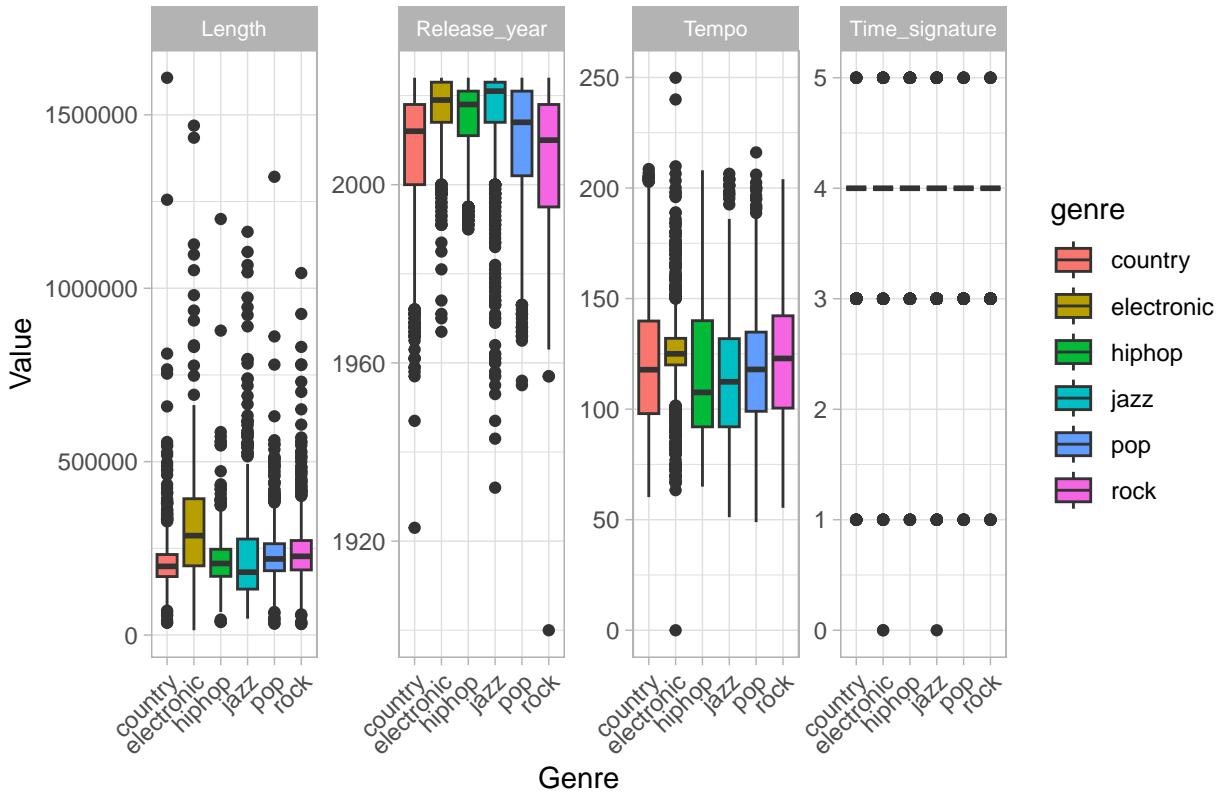
music_data_long <- pivot_longer(genre_data,
                                 cols = c(Length, Release_year, Tempo, Time_signature),
                                 names_to = "Feature",
                                 values_to = "Value")

ggplot(music_data_long, aes(x = genre, y = Value, fill = genre)) +
  geom_boxplot() +
  facet_wrap(~ Feature, scales = "free", ncol = 4) +
  theme_light() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        strip.text.x = element_text(size = 8)) +
  labs(title = "Distribution of Audio Features by Genre",
       x = "Genre", y = "Value")

## Warning: Removed 660 rows containing non-finite outside the scale range
## ('stat_boxplot()').

```

## Distribution of Audio Features by Genre



**Length:** Electronic music (yellow) has the highest median song length, indicating that it typically has longer tracks. Country has the most outliers above the upper whisker, highlighting the presence of significantly longer tracks in this genre.

**Release\_year:** The IQR for electronic (yellow) is the most compact, indicating less variation in the release years, while rock (pink) has a broader IQR, indicating more diversity in the release years of its tracks. Rock (pink) reaches back to the earliest years, indicating a history of older releases in this genre.

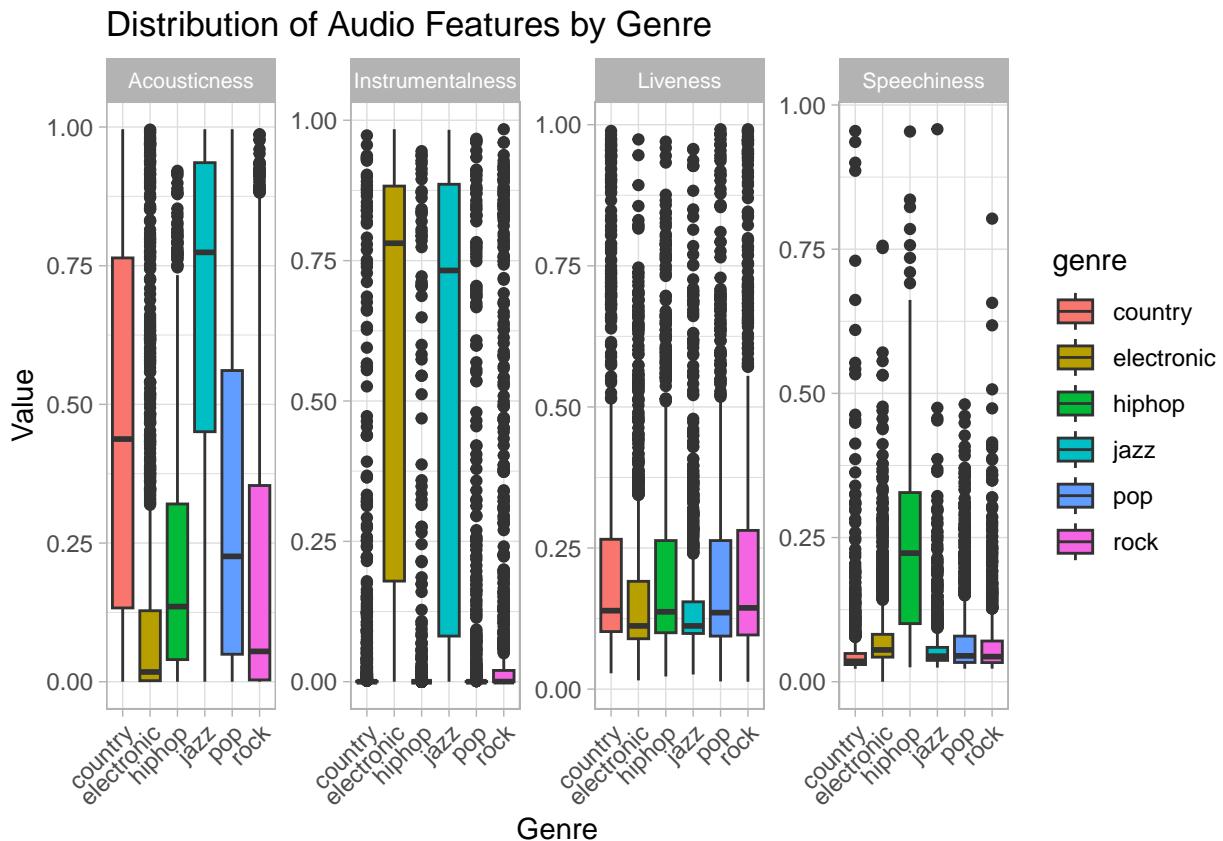
**Tempo:** The median tempo is similar across genres, but electronic (yellow) and rock (pink) have slightly higher median tempos, which could indicate a tendency for these genres to have faster beats. And electronic (yellow) has a narrower IQR, indicating more consistency in tempo.

**Time\_signature:** The median for all genres appears to be 4, indicating a standard beat pattern across genres. The IQR for all genres is very narrow or non-existent. The absence of outliers indicates that it is rare for songs in these genres to deviate from the common 4/4 time signature.

```
music_data_long <- pivot_longer(genre_data,
                                cols = c(Acousticness, Instrumentalness,
                                         Liveness, Speechiness),
                                names_to = "Feature",
                                values_to = "Value")

ggplot(music_data_long, aes(x = genre, y = Value, fill = genre)) +
  geom_boxplot() +
  facet_wrap(~ Feature, scales = "free", ncol = 4) +
  theme_light() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        strip.text.x = element_text(size = 8)) +
```

```
labs(title = "Distribution of Audio Features by Genre",
x = "Genre", y = "Value")
```



**Acousticness:** Jazz (light blue) has the highest median, indicating that on average, jazz songs have a higher degree of acousticness. Electronic (yellow) has the lowest median, suggesting that electronic music typically has less acoustic sound. Electronic music have a narrower IQR, suggesting more consistency in their levels of acousticness.

**Instrumentalness:** The jazz genre has a large IQR, showing a broad range of instrumentalness, suggesting varying levels of instrumental elements in jazz music. Rock and country exhibit a significant number of outliers, emphasizing tracks that are predominantly instrumental.

**Liveness:** The IQR is narrow for all genres, with rock (pink) being slightly wider, indicating a modest variation in the live aspect of their tracks. There are a few outliers, particularly in jazz and country, showing some tracks with higher liveness.

**Speechiness:** Hip-hop (green) has the highest median, which is expected due to the spoken word element of the genre. Hip-hop has a wide IQR, indicating a range of speechiness, from songs with more instrumental parts to those with spoken words. Also, Hip-hop shows the longest whiskers, reflecting a broad spread of speechiness in the genre.

```
music_data_long <- pivot_longer(genre_data,
                                cols = c(Danceability, Energy, Loudness, valence),
                                names_to = "Feature",
                                values_to = "Value")

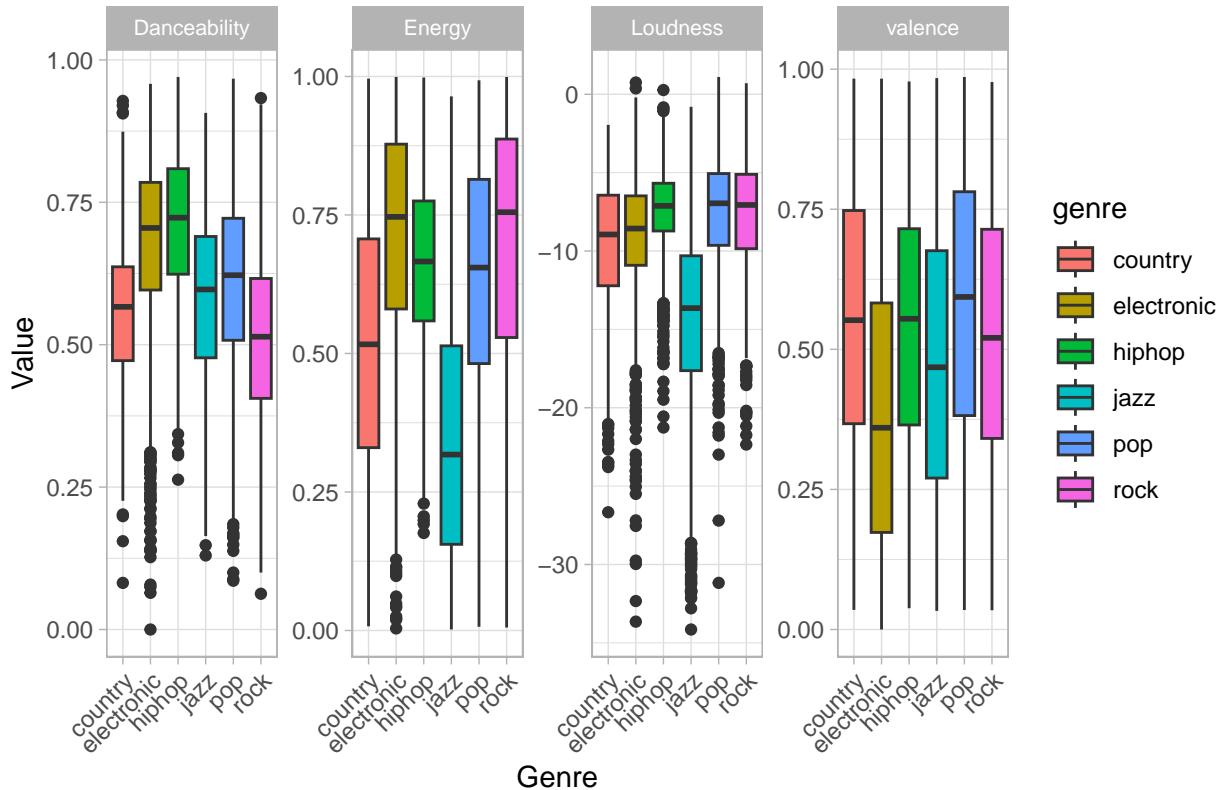
ggplot(music_data_long, aes(x = genre, y = Value, fill = genre)) +
```

```

geom_boxplot() +
facet_wrap(~ Feature, scales = "free", ncol = 4) +
theme_light() +
theme(axis.text.x = element_text(angle = 45, hjust = 1),
      strip.text.x = element_text(size = 8)) +
labs(title = "Distribution of Audio Features by Genre",
     x = "Genre", y = "Value")

```

## Distribution of Audio Features by Genre



**Danceability:** Hip-hop (green) has the highest median, indicating its songs are generally the most danceable. Rock (pink) has the lowest median, suggesting it is less danceable on average than other genres. The whiskers indicate the overall range of danceability, with hip-hop showing a broad spread, indicating diversity in this aspect.

**Energy:** Jazz have lower median value, which suggests this genre's songs are typically much less energetic. There are outliers for several genres, with electronic (yellow) showing songs with particularly low energy levels.

**Loudness:** The median loudness levels are fairly central across all genres, with jazz (light blue) slightly higher, implying it tends to be louder. Jazz again shows a wide IQR, suggesting a varied loudness range within the genre.

**valence:** Pop (purple) has one of the highest median valence scores, suggesting its songs often have a more positive mood. In contrast, electronic (yellow) has a lower median, indicating a generally less positive mood.

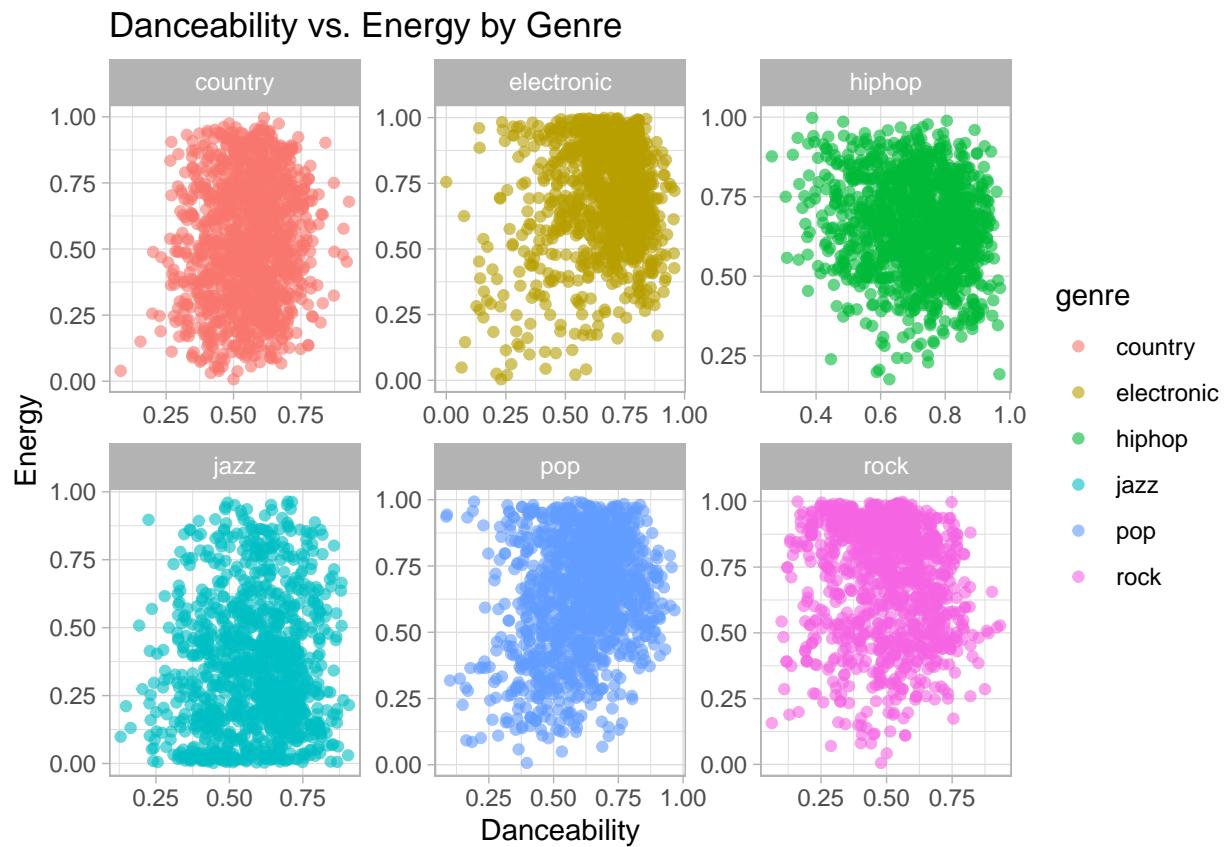
After exploring the relationship between single feature and genre, now we aim to investigate future the relationships between pairs of audio features across different music genres by **scatter plot**.

The plots are designed to uncover potential interaction effects that could be important for data engineering in the following section.

```

ggplot(genre_data, aes(x = Danceability, y = Energy, color = genre)) +
  geom_point(alpha = 0.6) +
  facet_wrap(~ genre, scales = "free") +
  theme_light() +
  labs(title = "Danceability vs. Energy by Genre",
       x = "Danceability", y = "Energy")

```

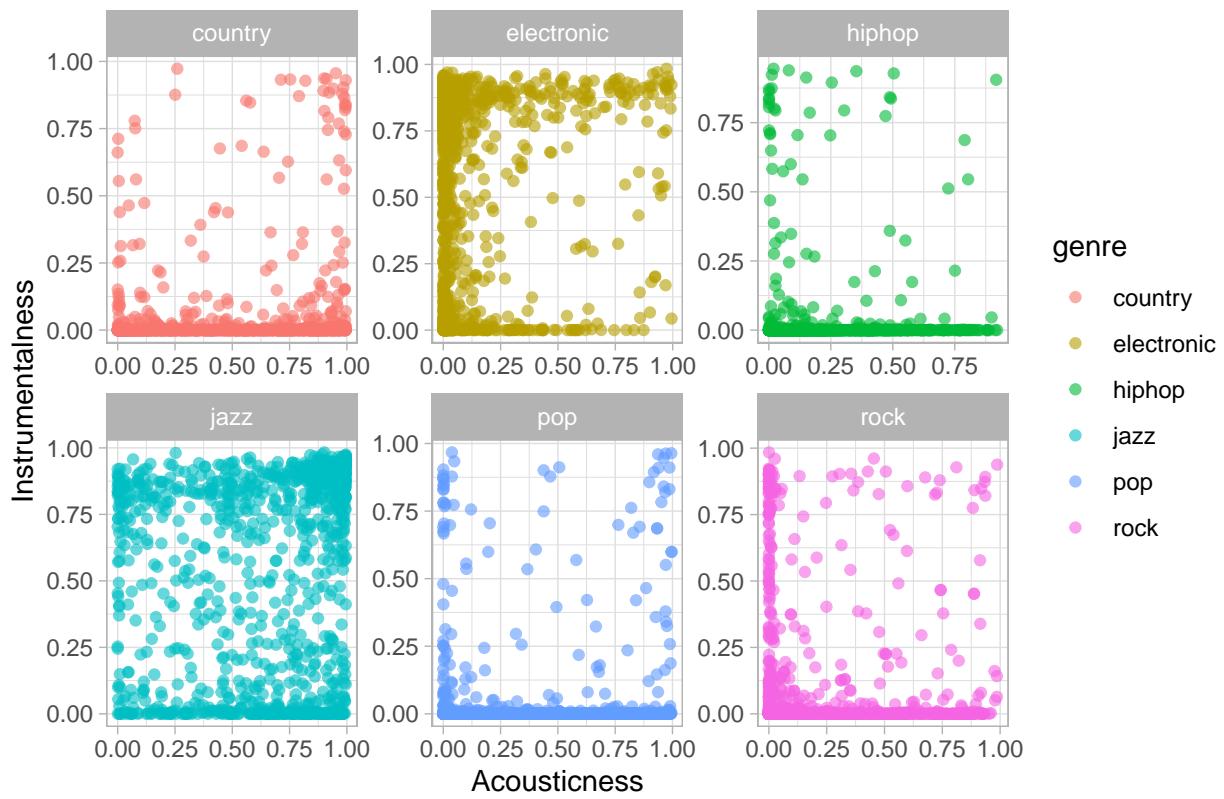


```

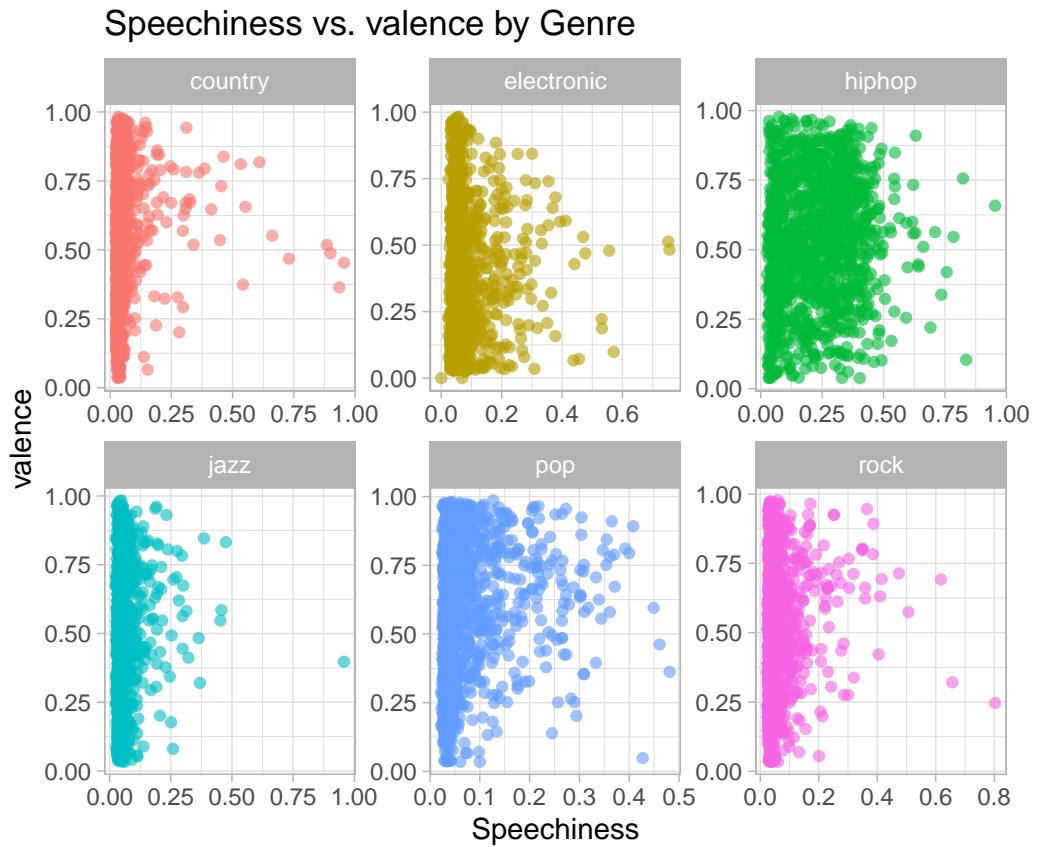
ggplot(genre_data, aes(x = Acousticness, y = Instrumentalness, color = genre)) +
  geom_point(alpha = 0.6) +
  facet_wrap(~ genre, scales = "free") +
  theme_light() +
  labs(title = "Acousticness vs. Instrumentalness by Genre",
       x = "Acousticness", y = "Instrumentalness")

```

## Acousticness vs. Instrumentalness by Genre



```
ggplot(genre_data, aes(x = Speechiness, y = valence, color = genre)) +
  geom_point(alpha = 0.6) +
  facet_wrap(~ genre, scales = "free") +
  theme_light() +
  labs(title = "Speechiness vs. valence by Genre",
       x = "Speechiness", y = "valence")
```



- **Danceability vs. Energy by Genre**

High danceability and energy levels are often associated with more upbeat and fast-paced music.

Electronic and Hip-hop music are often designed for dancing and tend to have higher danceability and energy levels.

- **Acousticness vs. Instrumentalness**

These features can help differentiate genres known for live instrumentation from those that are more electronic or vocal-heavy.

Jazz exhibits higher levels of acousticness and instrumentalness, reflecting the genre's reliance on acoustic instruments and instrumental solos.

- **Speechiness vs. Valence**

We think it will be interesting to explore genres that feature more lyrical content with varying emotional tones.

Hip-hop is distinguished by its use of spoken words, resulting in higher speechiness levels. The valence in hip-hop can vary widely, reflecting the diverse emotional tones explored in the lyrics. In contrast, Jazz and Electronic music might show lower speechiness.

## 7.2 Data Pre-processing

### 7.2.1 Data Engineering

Based on the scatter plots above, we create three new features.

```

#Energy and Danceability:
genre_data$Energy_Danceability_Interaction <-
  genre_data$Energy * genre_data$Danceability

#Acousticness and Instrumentalness
genre_data$Acousticness_Instrumentalness_Interaction <-
  genre_data$Acousticness * genre_data$Instrumentalness

#Speechiness and Valence:
genre_data$Speechiness_Valence_Interaction <-
  genre_data$Speechiness * genre_data$valence

```

### 7.2.2 Normalization

Because of potential classification methods in the following parts (e.g. SVM), we scale numerical features to a standard range.

The normalization formula used is  $(x - \min(x)) / (\max(x) - \min(x))$ , which scales the data to a [0, 1] range.

Part of audio features have already in the format of [0,1], therefore, we only need to normalize the remaining numeric features.

```

# Normalizing selected features
features_to_normalize <- c("Loudness", "Tempo", "Time_signature",
                           "key", "Release_year", "Length",
                           "Energy_Danceability_Interaction",
                           "Acousticness_Instrumentalness_Interaction",
                           "Speechiness_Valence_Interaction")

genre_data[features_to_normalize] <-
  as.data.frame(lapply(
    genre_data[features_to_normalize],
    function(x) (x - min(x, na.rm = TRUE)) /
      (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))))

head(genre_data)

##          Length Acousticness Danceability Energy Instrumentalness Liveness
## 1 0.12206093     0.1130      0.762  0.692                  0  0.0940
## 2 0.11522191     0.0965      0.458  0.561                  0  0.1560
## 3 0.15068500     0.4170      0.637  0.690                  0  0.2110
## 4 0.07416144     0.0642      0.892  0.503                  0  0.1020
## 5 0.10374216     0.3830      0.505  0.694                  0  0.1600
## 6 0.10504593     0.1530      0.377  0.584                  0  0.0662
##          Loudness Speechiness      Tempo Time_signature        key valence genre
## 1 0.8560082       0.0438 0.4598999      0.8 0.4545455  0.397   pop
## 2 0.8320095       0.0269 0.5998599      0.6 0.9090909  0.206   pop
## 3 0.8775389       0.0693 0.5616530      0.8 0.8181818  0.103   pop
## 4 0.7677011       0.1780 0.5600480      0.8 0.5454545  0.615   pop
## 5 0.8150460       0.0295 0.6161977      0.8 0.1818182  0.406   pop
## 6 0.8549302       0.0495 0.3669041      0.8 0.1818182  0.444   pop

```

```

##   Release_year Energy_Danceability_Interaction
## 1      0.8709677          0.6164703
## 2      0.9516129          0.3003858
## 3      0.8790323          0.5138538
## 4      0.9919355          0.5245464
## 5      0.9919355          0.4097339
## 6      0.9677419          0.2573981
##   Acousticness_Instrumentalness_Interaction Speechiness_Valence_Interaction
## 1                               0           0.027700675
## 2                               0           0.008827653
## 3                               0           0.011370935
## 4                               0           0.174389708
## 5                               0           0.019079798
## 6                               0           0.035011757

```

### 7.3 Feature Selection

We would like to conduct model-based selection.

**Decision tree** usually works as a starting point for classification model.

```

#split into training and testing sets to evaluate model performance.
set.seed(19603)

index <- createDataPartition(genre_data$genre, p = 0.8, list = FALSE)
traindata <- genre_data[index, ]
testdata <- genre_data[-index, ]

```

```

#Decision tree
dt_model <- rpart(genre ~ ., data = traindata, method = "class")
dt_predictions <- predict(dt_model, newdata = testdata, type = "class")

# Evaluating model performance
confusionMatrix(dt_predictions, testdata$genre)

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   country electronic hiphop jazz pop rock
##   country       118          9    15   28   77   42
##   electronic     9         167     9   31     3   30
##   hiphop        10          8   161     6   31   17
##   jazz          25         27     8  145   12   15
##   pop           69         18   58    20  112   44
##   rock          23          6     2     0   18   76
##
## Overall Statistics
##
##               Accuracy : 0.5376
##                 95% CI : (0.5115, 0.5635)
##   No Information Rate : 0.1753
##   P-Value [Acc > NIR] : < 2.2e-16
##

```

```

##                               Kappa : 0.444
##
##  Mcnemar's Test P-Value : 2.501e-11
##
##  Statistics by Class:
##
##                                Class: country Class: electronic Class: hiphop Class: jazz
## Sensitivity                  0.46457      0.7106      0.6364      0.6304
## Specificity                 0.85690      0.9325      0.9398      0.9286
## Pos Pred Value               0.40830      0.6707      0.6910      0.6250
## Neg Pred Value               0.88276      0.9433      0.9243      0.9302
## Prevalence                   0.17529      0.1622      0.1746      0.1587
## Detection Rate                0.08144      0.1153      0.1111      0.1001
## Detection Prevalence         0.19945      0.1718      0.1608      0.1601
## Balanced Accuracy            0.66074      0.8215      0.7881      0.7795
##
##                                Class: pop   Class: rock
## Sensitivity                  0.44269      0.33929
## Specificity                 0.82525      0.96000
## Pos Pred Value               0.34891      0.60800
## Neg Pred Value               0.87500      0.88822
## Prevalence                   0.17460      0.15459
## Detection Rate                0.07729      0.05245
## Detection Prevalence         0.22153      0.08627
## Balanced Accuracy            0.63397      0.64964

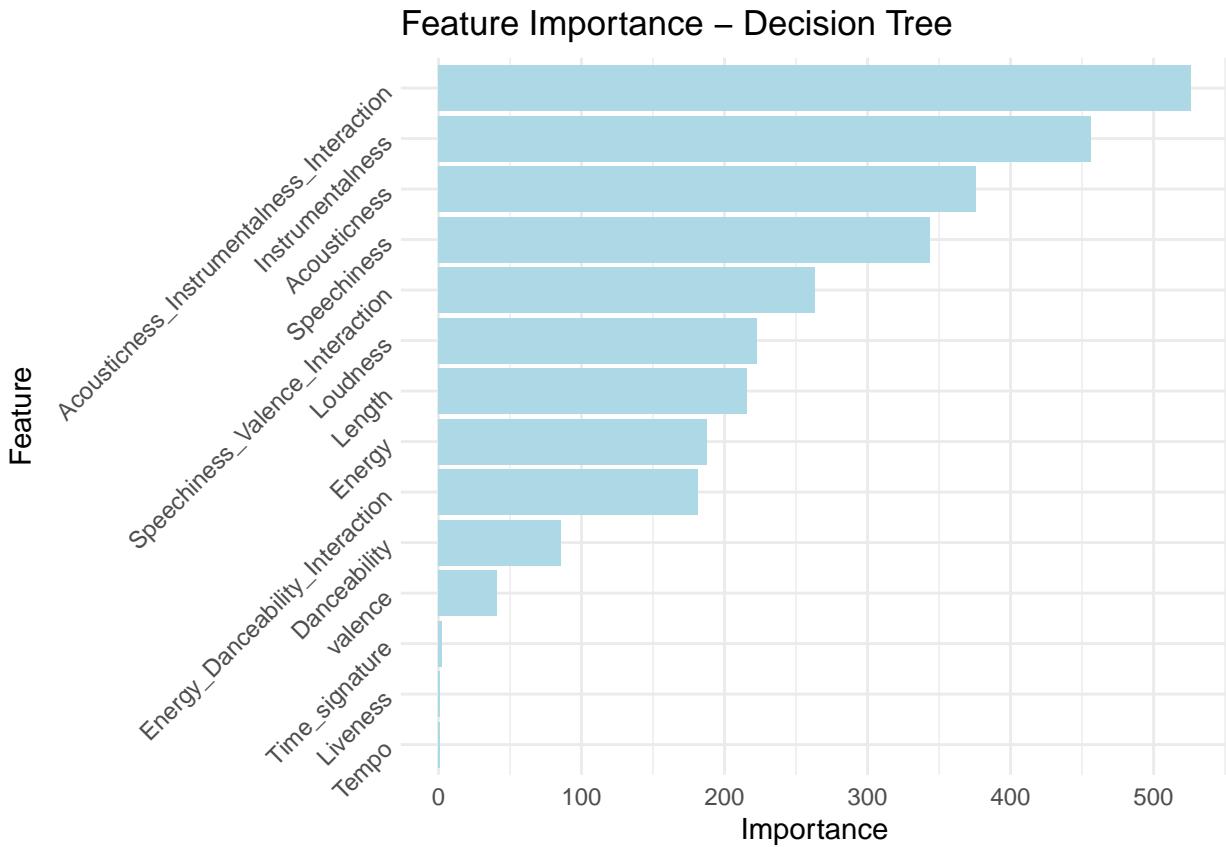
```

```

# Extracting feature importance
importance <- dt_model$variable.importance
importance_df <- data.frame(Feature = names(importance), Importance = importance)
importance_df <- importance_df[order(-importance_df$Importance),]

# Visualizing the feature importance
ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_col(fill = "lightblue") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Feature Importance - Decision Tree", x = "Feature", y = "Importance") +
  theme(axis.text.y = element_text(angle = 45))

```



This horizontal bar chart shows the relative importance of different features as determined by a decision tree model used for genre classification. From the evaluation matrix, we see the decision tree model shows some ability to predict genres beyond random guessing or always choosing the most common genre, and there is definitely room for improvement.

Based on above feature importance table, it seems like **Time\_signature**, **Liveness**, **Tempo** and **Release\_year** are the least important features.

Dropping less important features can help simplify our model.

```
train_data_updated <-
  traindata[,
    !(names(traindata) %in% c("Time_signature", "Liveness", "Tempo", "Release_year"))]

test_data_updated <-
  testdata[,
    !(names(testdata) %in% c("Time_signature", "Liveness", "Tempo", "Release_year"))]
```

## 7.4 Build Model

**Potential Model Choices:** Random Forest; Support Vector Machine; Neural Network

### 7.4.1. Model 1: Random Forest

```

rf_model <- randomForest(genre ~ ., data=train_data_updated, ntree=500)
rf_predictions <- predict(rf_model, newdata=test_data_updated)
cm_rf <- confusionMatrix(rf_predictions, test_data_updated$genre)

```

#### 7.4.2. Model 2: SVM

```

svm_model <- svm(genre ~ ., data = train_data_updated, kernel = "radial")
svm_predictions <- predict(svm_model, newdata = test_data_updated)
cm_svm <- confusionMatrix(svm_predictions, testdata$genre)

```

#### 7.4.3. Model 3: Neural Network

```

size <- 10 # Number of units in the hidden layer
decay <- 0.01 # Weight decay for regularization

nn_model <-
  nnet(genre ~ ., data = train_data_updated,
       size = size, decay = decay, maxit = 500, linout = FALSE, trace=FALSE)

nn_pred_probs <-
  predict(nn_model, newdata=test_data_updated, type="raw")

nn_pred_class <-
  factor(max.col(nn_pred_probs),
        levels=1:length(levels(train_data_updated$genre)),
        labels=levels(train_data_updated$genre))

cm_nn <- confusionMatrix(nn_pred_class, test_data_updated$genre)

```

### 7.5 Model Selection

```

overall_accuracy_rf <- cm_rf$overall['Accuracy']
overall_accuracy_svm <- cm_svm$overall['Accuracy']
overall_accuracy_nn <- cm_nn$overall['Accuracy']

accuracy_comparison <- data.frame(
  Model = c("Random Forest", "SVM", "Neural Network"),
  Accuracy = c(overall_accuracy_rf, overall_accuracy_svm, overall_accuracy_nn) )

print(accuracy_comparison)

##          Model Accuracy
## 1 Random Forest 0.6135266
## 2           SVM 0.6038647
## 3 Neural Network 0.6031746

```

This simple table presents the overall accuracy of the three models. Although the accuracy rates are somewhat close, the slight advantage of the Random Forest model could generally provide a more robust prediction by reducing the likelihood of overfitting.

```
# Extracting Balanced Accuracy for each class
balanced_accuracy_rf <- cm_rf$byClass[, "Balanced Accuracy"]
balanced_accuracy_svm <- cm_svm$byClass[, "Balanced Accuracy"]
balanced_accuracy_nn <- cm_nn$byClass[, "Balanced Accuracy"]

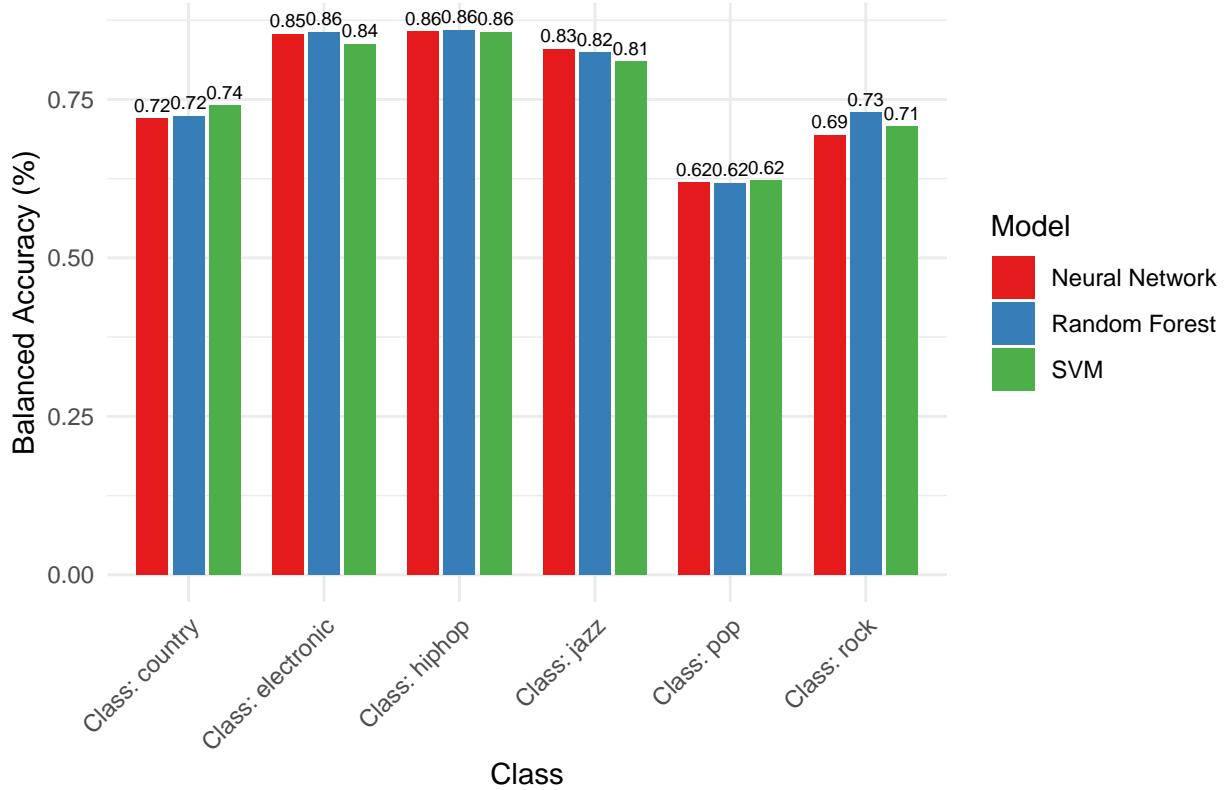
df_balanced_accuracy_rf <-
  data.frame(Class = rownames(cm_rf$byClass), Balanced_Accuracy = balanced_accuracy_rf, Model = "Random Forest")
df_balanced_accuracy_svm <-
  data.frame(Class = rownames(cm_svm$byClass), Balanced_Accuracy = balanced_accuracy_svm, Model = "SVM")
df_balanced_accuracy_nn <-
  data.frame(Class = rownames(cm_nn$byClass), Balanced_Accuracy = balanced_accuracy_nn, Model = "Neural Network")

combined_balanced_accuracy <-
  rbind(df_balanced_accuracy_rf, df_balanced_accuracy_svm, df_balanced_accuracy_nn)

library(ggplot2)

ggplot(combined_balanced_accuracy, aes(x = Class, y = Balanced_Accuracy, fill = Model)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) + # Adjust bar width
  geom_text(aes(label = sprintf("%.2f", Balanced_Accuracy)),
            position = position_dodge(width = 0.8), vjust = -0.5, size = 2.5) +
  theme_minimal() +
  labs(title = "Balanced Accuracy by Class and Model", y = "Balanced Accuracy (%)", x = "Class") +
  scale_fill_brewer(palette = "Set1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

## Balanced Accuracy by Class and Model



**Balanced Accuracy by Class and Model:** This bar chart compares the balanced accuracy of three different machine learning models - Neural Network, Random Forest, and SVM (Support Vector Machine) - for predicting different music genres. The chart indicates that genres like electronic and hip-hop have distinguishing features that are captured well by the models, while other genres might be more challenging to differentiate due to overlapping characteristics or less distinct features.

#8 Conclusion Our project undertook an innovative task where we utilized Spotify's audio features to predict song popularity and genre, demonstrating the intersection of musicology and data science. Through extensive data preprocessing, exploratory data analysis (EDA), and model building, we identified certain audio features that indicate song popularity and genre classification. Exploratory data analysis (EDA) reveals the distribution of features such as danceability, energy, and acousticity across genres, while regression and classification models attempt to quantify these relationships. In terms of predicting song popularity, our work highlights the challenges of modeling with diverse datasets, as evidenced by the range of performance of the models and their predictive accuracy. The models achieved varying degrees of success, with Random Forests and Generalized Linear Models (GLMs) showing potential in capturing the complex dynamics that lead to song popularity on Spotify. Further work needs to be conducted regarding the specificity and sensitivity of these two models. Improving false positive will be a potential topic of our next stage. In terms of genre classification, our innovative approach to feature selection and engineering (e.g., creating interaction terms and normalizing selected features) paved the way for building nuanced models that are able to distinguish between genres with reasonable accuracy. The application of models such as Random Forests, SVMs, and Neural Networks in predicting genres illustrates a comprehensive attempt to leverage Spotify's rich dataset to understand music trends.

#9 Future Work Enhanced Feature Engineering: We might try to explore deeper into creating more complex interaction terms and investigating unsupervised learning techniques for feature extraction. Maybe we could uncover hidden patterns within the audio features which significantly impact both genre classification and song popularity.

Model Exploration and Optimization: We will also try to experiment with advanced modeling techniques, including some ensemble methods for sequential analysis of songs.

Handling Class Imbalance: We will consider resampling Techniques like Oversampling or undersampling. Aim tp mitigate the impact of imbalance and improve the performance of our models.

User-Centric Applications: Last but not least, we might explore the development of user-centric applications such as personalized song recommendations and dynamic genre classification tools. In this way, we could leverage the insights and models developed in this project to enhance music discovery and curation on streaming platforms.