Business Analytics for Managers and Entrepreneurs: A Comprehensive Report

Contents

Introduction

Through this R script researchers perform an extensive investigation of popular songs from Spotify 2023 utilizing multiple data science methods. The analysis consists of three main sections including Music Analysis and Artist Impact and Predictive Analytics. The code in the Music Analysis section investigates the distribution of audio features together with their inter-feature correlations along with BPM patterns while analyzing key/mode preferences of most streamed songs. The Artist Impact section examines active artist trends by focusing on their preference for solo work or collaborative albums as well as ranking top artists by both song and stream counts and the timing of releases and their audio feature choices and preferences. The Predictive Analytics segment implements three machine learning methodologies including Linear Regression combined with Random Forest and Decision Tree for determining the most effective prediction variables of streaming success. Visual representations as PNG files emerge from the analysis to present clear evidence about successful song factors on Spotify.

Data Preparation and Preprocessing

Data preprocessing operations start by cleaning all data in the Spotify dataset before performing accurate studies. The script transforms the streams column from textual format to numerical values by stripping away commas before renaming selected columns which deletes the percentage symbols required for data calculations. Standardization methods normalize feature names from "danceability_%" to "danceability" along with "energy" for better data handling and graphical display functions.

The script implements standardization of column formats before proceeding to detect any missing values in the essential "streams" column. The script uses stream data median values as a reliable solution to replace detected missing values because median values help avoid result distortions from outlier effects. All future music features analysis and predictive modeling depend on the reliable spotify_clean dataset which results from the initial data preparation steps.

Descriptive Analysis

The descriptive evaluation establishes crucial information about musical features together with artist performance trends in Spotify's list of top songs for 2023. The Music Analysis section evaluates various elements of audio features while studying correlations of fundamental elements and BPM patterns alongside key/mode preferences to define characteristics of popular music.

The Artist Impact analysis concentrates on music creators by analyzing both their collaboration methods and their dominant styles together with their success rate. The section investigates timing methods for album releases and measures single and collaborative track performance to show how artists impact Spotify streaming results through their choices and teaming up strategies.

Music Analysis

Multiple insightful visual presentations contained within the provided figures exhibit detailed audio attribute assessments of popular music songs. The features included danceability alongside energy and valence along with acousticness and more display important relationships and patterns throughout the popular Spotify tracks data set.

Correlation between Audio Features

A correlation matrix displays the relationships which exist between vital sound characteristics. Energy and danceability show very high positive association (0.8) because tracks that make people want to dance usually maintain energetic qualities. According to data (0.7), music tracks that are easy to dance to tend to share positive emotional tones. Speechiness and acousticness produce a negative relationship (-0.4) which demonstrates higher levels of spoken content within songs typically correspond to lower sonic acoustics.

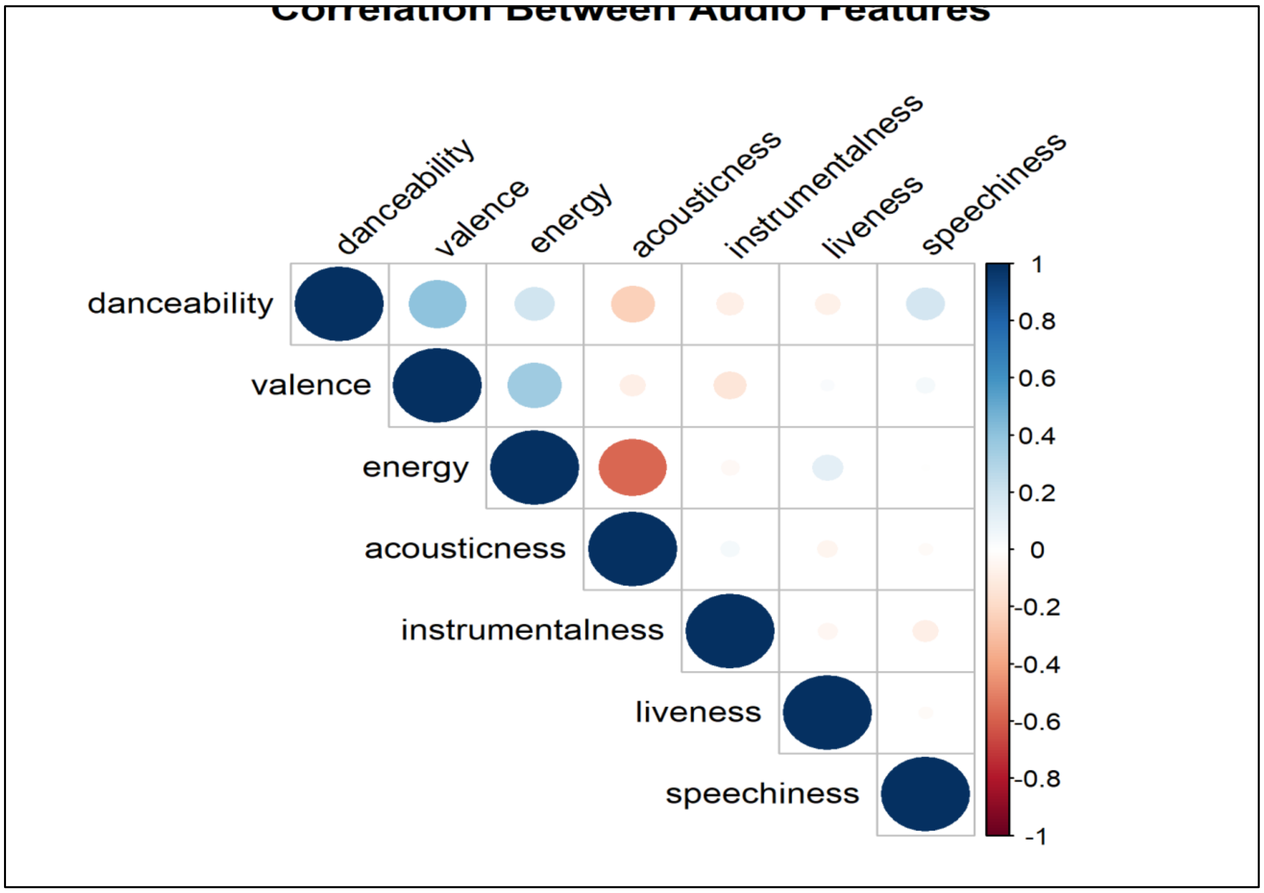

Figure 1: Correlation distribution of audio features represnting several features

Distribution of Audio Features

The graphical representation of box plots shows the written distribution of various properties. The energy scale extends across multiple values until high energy values appear which align with EDM music. Most popular songs exhibit moderate levels of danceability because the feature distribution peaks at the mid-point of the spectrum. Acousticness reveals a strong bias against low values thus demonstrating that electronic production is the main musical force in contemporary popular music standards.

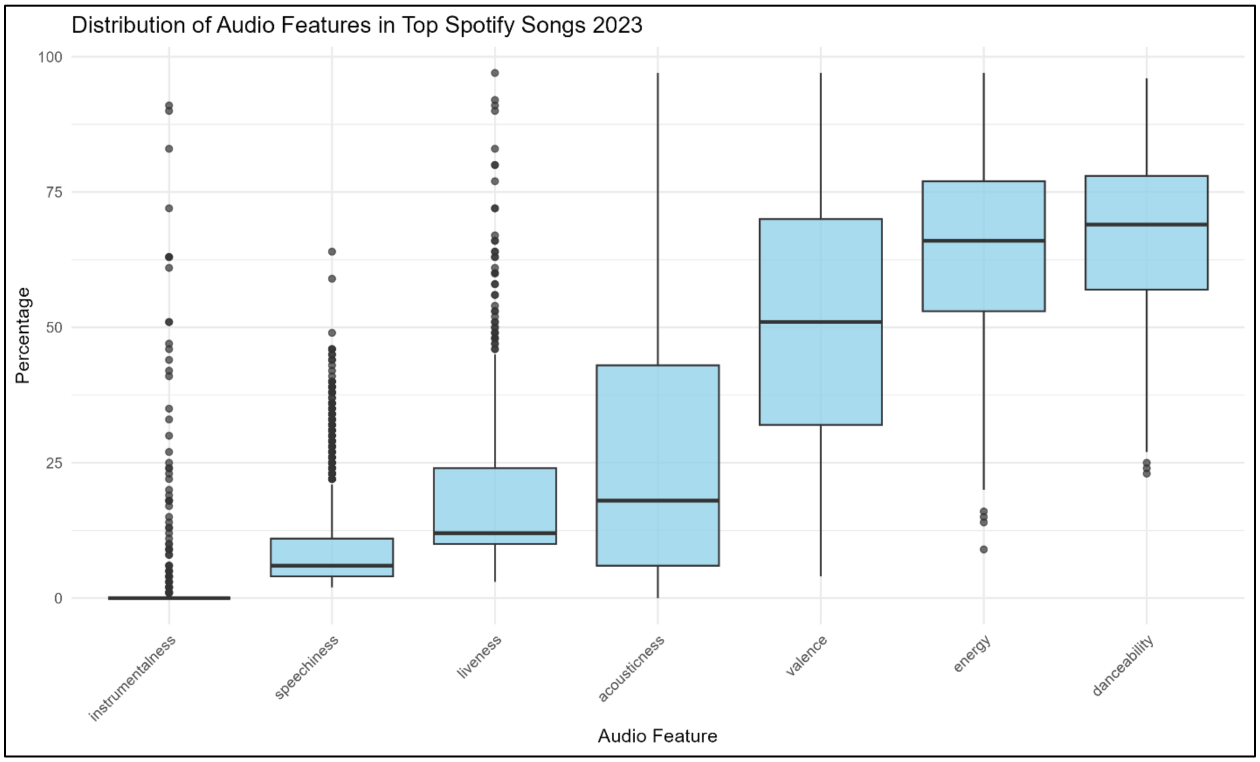

Figure 2: Boxplot showing the distribution of audio features

BPM Distribution

Beats per minute (BPM) frequency data shows two main peaks that appear within the lower range and the higher range. Both slow and quick-paced musical tracks find popularity in Spotify while the frequent occurrence of mid-tempo songs demonstrates the broad range of listener preferences.

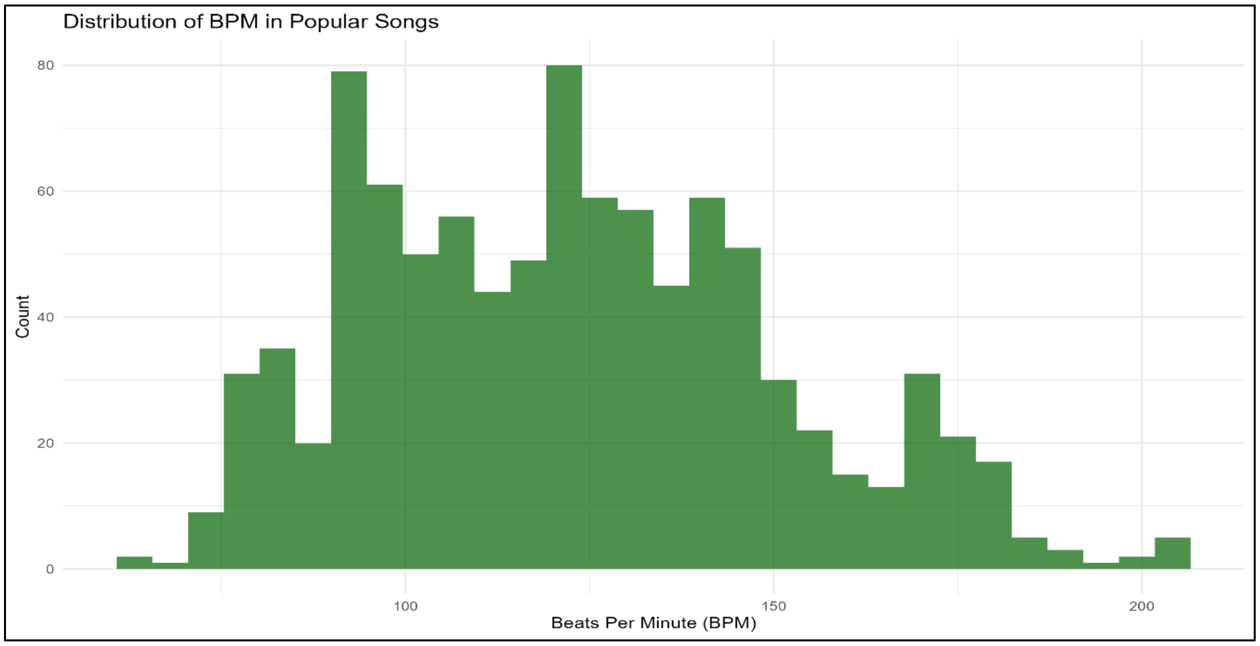

Figure 3: Beats per minute frquency plot

Key and Mode Distribution

Most musical songs exist within major keys while minor keys show less frequent usage according to key and mode distribution patterns. The mainstream music industry primarily utilizes major keys because these positive emotion-based keys become more common than minor keys.

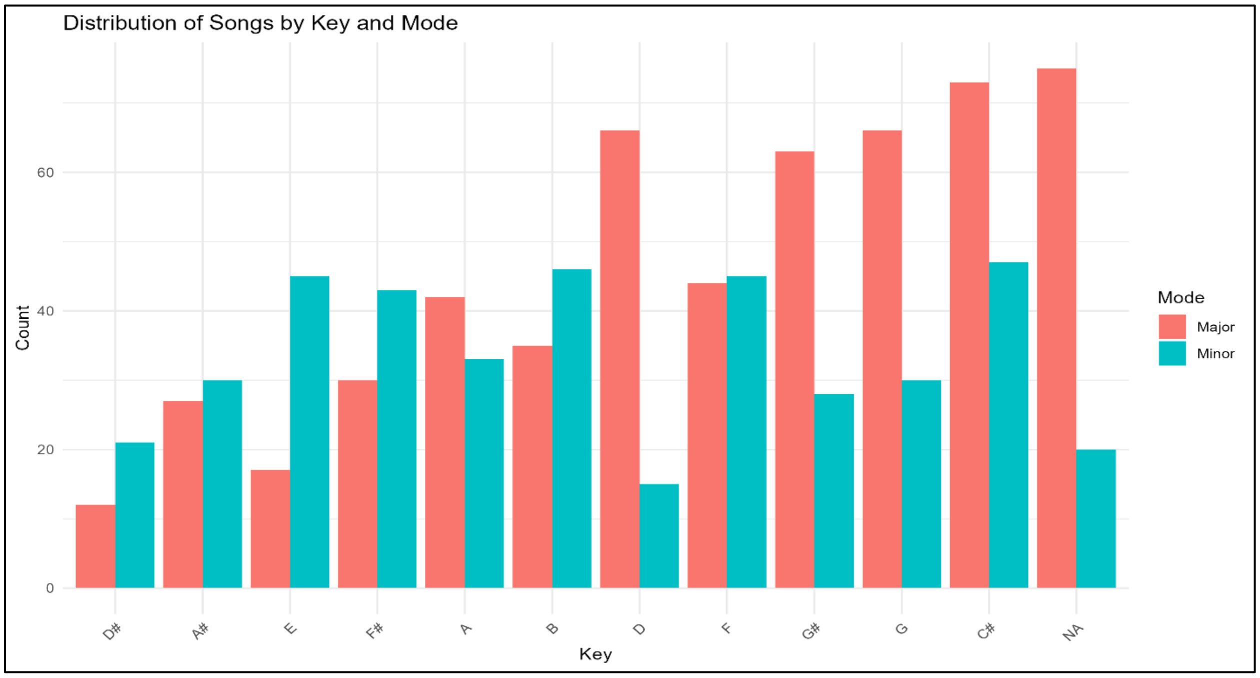

Figure 4: Distribution of Key and Mode features of spotify data

Comparison of Audio Features in Top Songs

Danceability and energy along with valence demonstrate higher values within the most successful music tracks according to audio metrics analysis. Hit songs obtain their joyful and memorable characteristics from these sound attributes. The level of energy distinguishes itself since popular tracks show continuous high energy levels. Sound quality features acousticness together with instrumentalness play a minor role in these tracks because music lovers choose recordings that use vocals and electronic production methods.

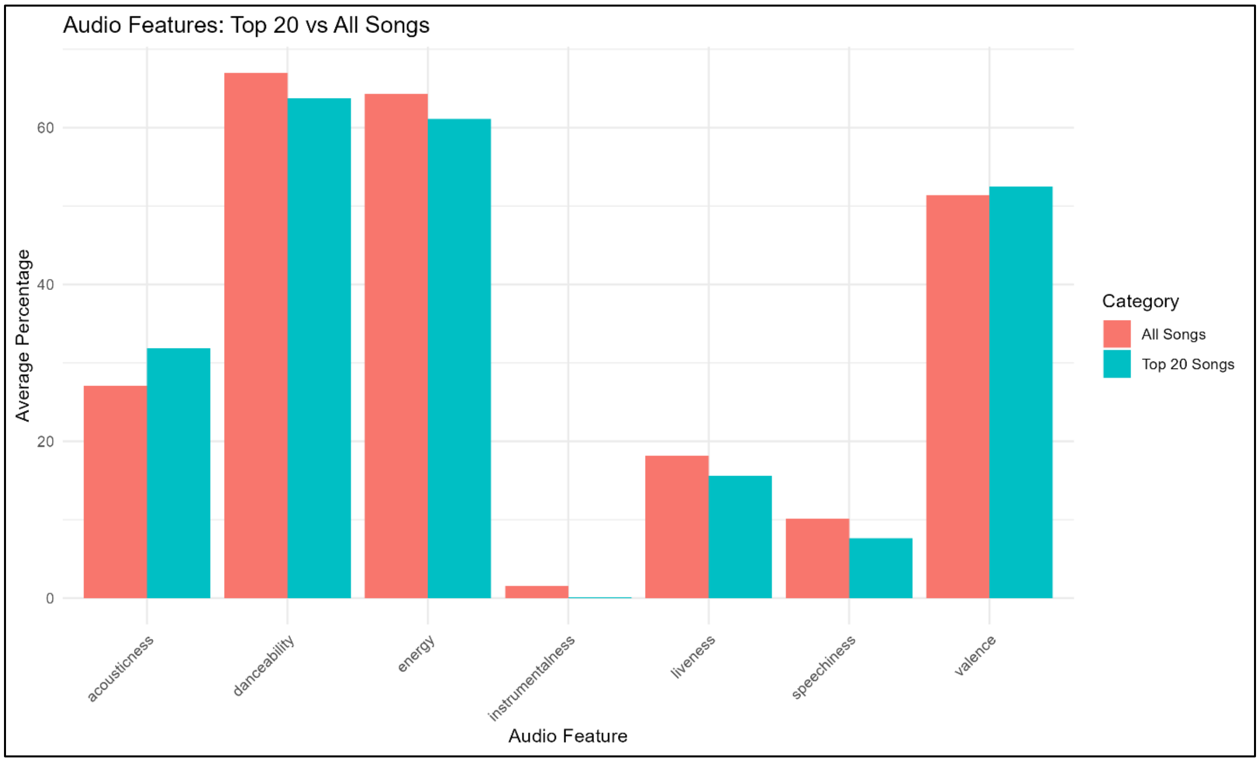

Figure 5: Distribution of Danceability and energy of audio

Artistic Impact

The influence of an artist on the success of their music is a key factor in determining its popularity and streaming performance on platforms like Spotify. The data provided in the figures offers valuable insights into how various elements such as an artist's output, collaborations, release strategies, and musical choices shape the success of their songs.

Top Artists and Song Counts

Figure 1 shows how many songs were released by top artists (see Bad Bunny and Taylor Swift whose production was considerably high). Their continual production secures the market and keeps them on the tip of listeners to always keep up with what they are doing in the market. Every artist releasing songs on a regular basis interacts more effectively with their fan base, which naturally results in higher streaming numbers.

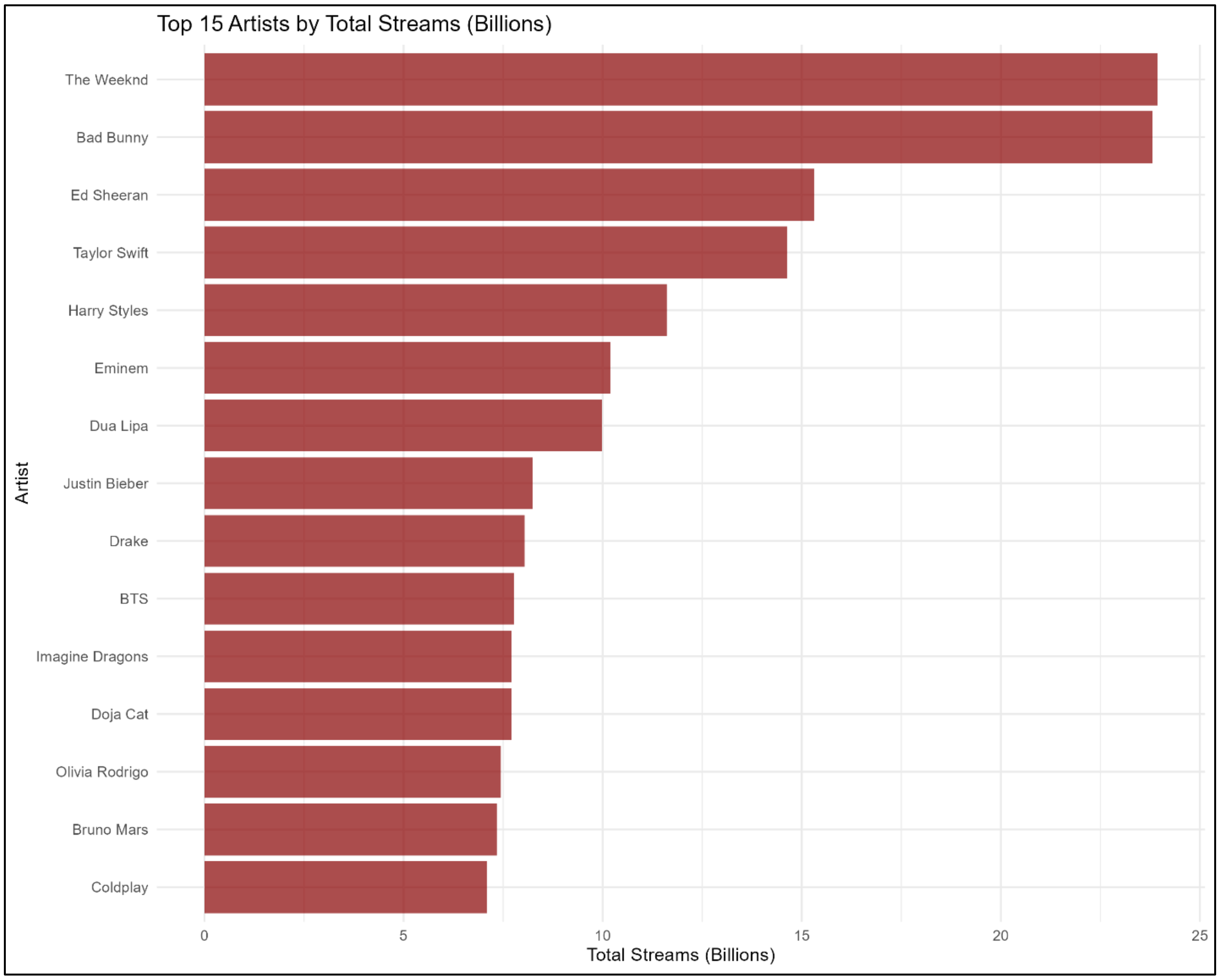

Figure 6: Visualizatin of top 15 artists based on total streams

Artist Streaming Performance

Figuratively, the chart is presented in Figure 2 and as you can see Bad Bunny, Taylor Swift, and The Weeknd are situated on top. Their high streaming numbers are well attributed to the fact that these artists have very large and global followings. As a result of their music being featured in popular playlists, visibility and engagement increase.

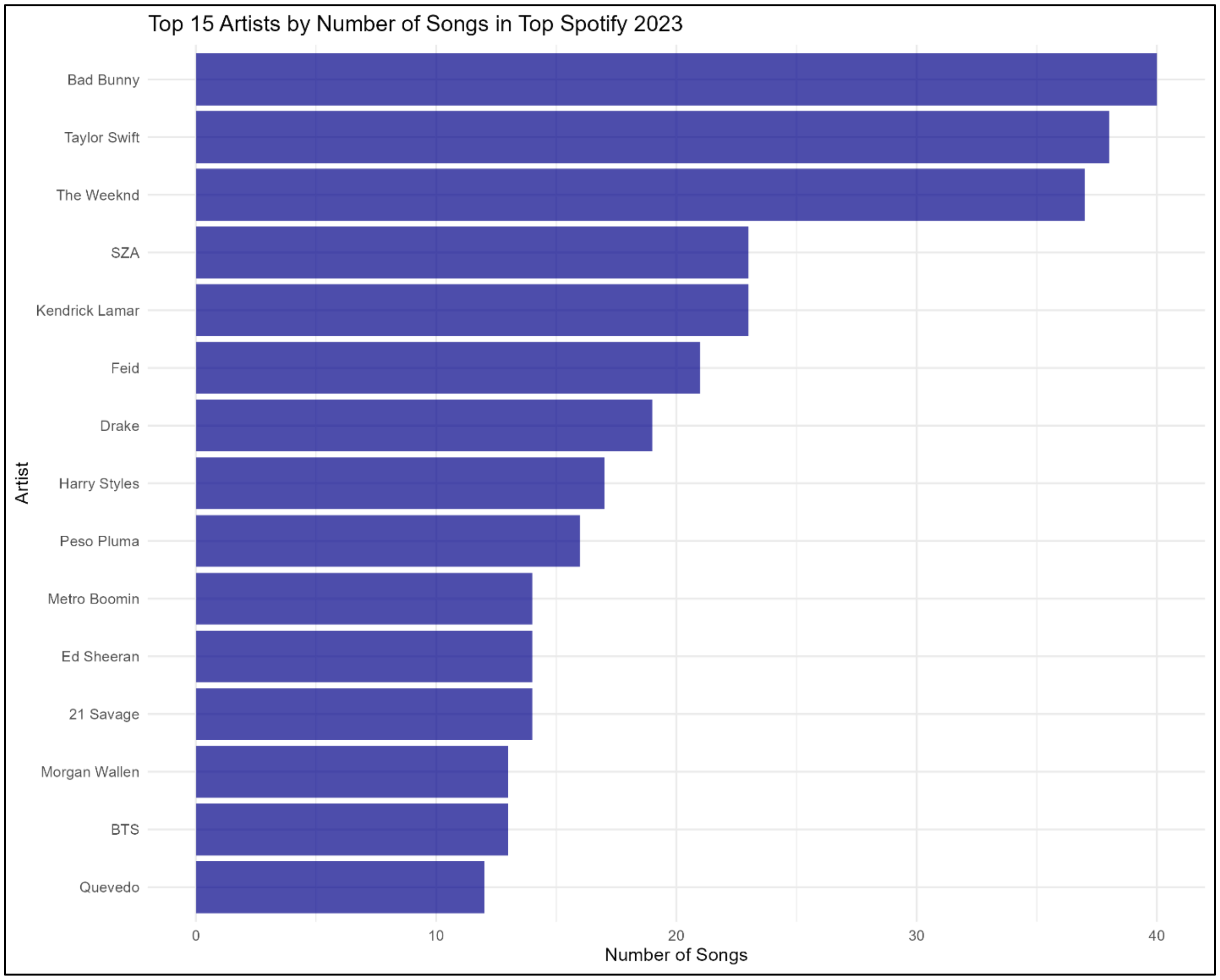

Figure 7: Distribution of top 15 artists based on Number of songs

Collaboration vs. Solo Performance

The performance of songs released as collaborations is compared to that of songs released alone. They also simply do better by streaming numbers in collaboratives since they already have this super combined fanbase between everyone involved. When artists work together, they expand the dimension of their audience from their own fanbase to someone new and raise their streaming numbers. This development demonstrates how strategic partnerships in the music industry are now playing a vital role in an artist's success and increasing exposure due to collaboration with other artists.

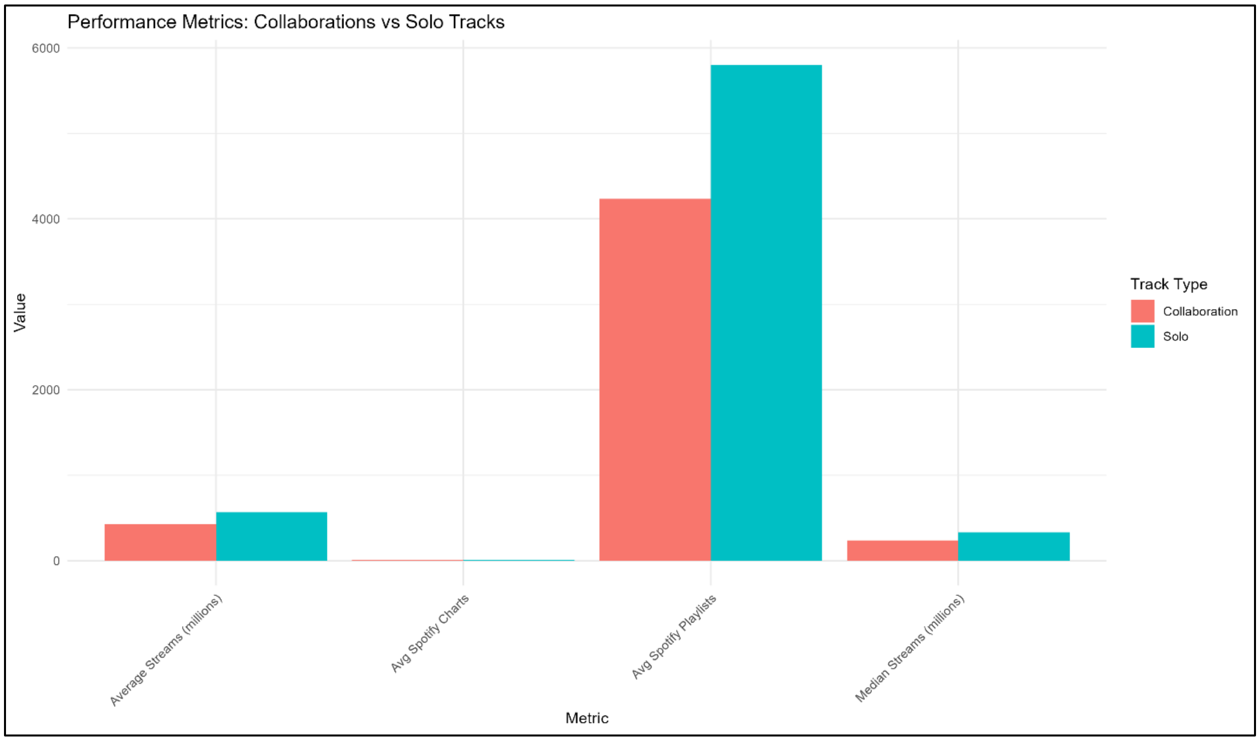

Figure 8: Arstists performance based on Collaborations vs Solo Perfermance

Release Timing Strategy

There is also release timing in a song as a very critical factor. Figure 4 shows that songs released when the key moments, e.g., holiday seasons, contribute to better perform. With a view to gaining some of that attention and streaming numbers, the artist can get a boost by riding on these optimal release windows. The figure shows that artists that plan their releases with industry events, or release in terms of high traffic periods tend to get better visibility and greater success on streaming platforms.

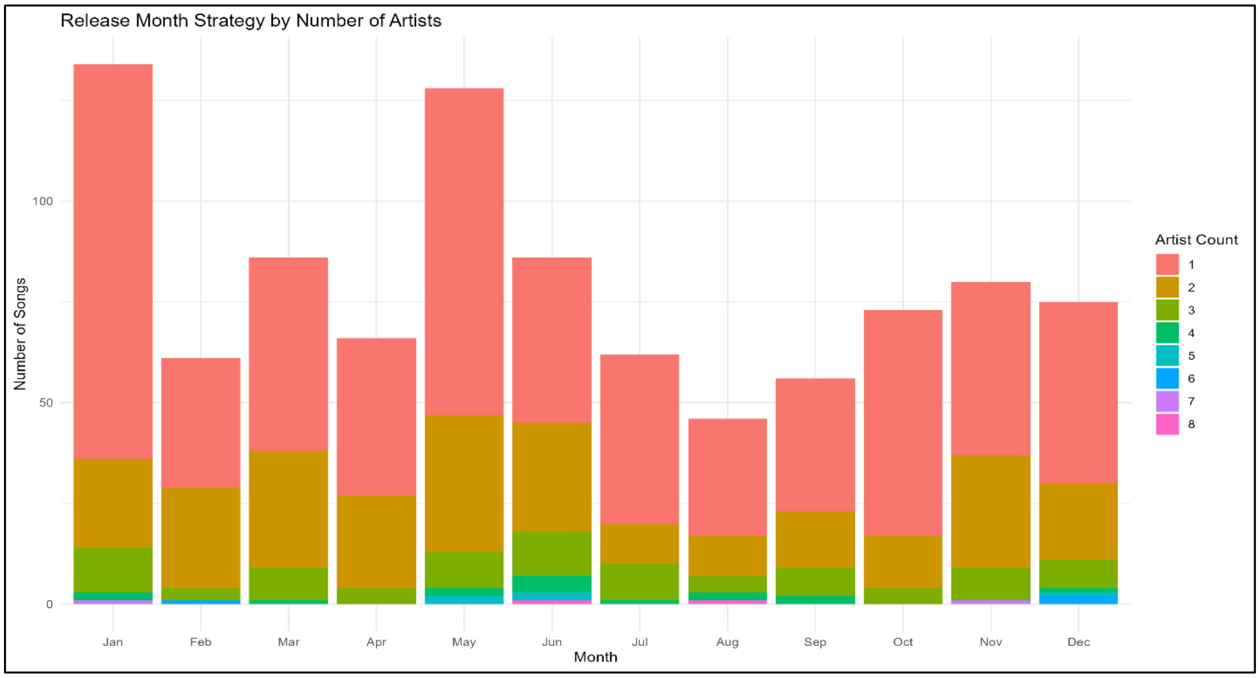

Figure 9: Distributions of Release dates of music

Artist's Key and Mode Influence

Another aspect that also determine the success of a song in an artist's music is the choice of key and mode in the song. Figure 5 also illustrates that on top artists' behalf, major keys are preferred, which are usually characterized by more upbeat and positive emotions. This echoes the popular music tendency to have major keys tracks leading the charge through the streaming charts because they are more energetic and pleasing to the ear. Such data reveals that little but strong differences in how audiences perceive an artist's songs depend on their choices of key and mode.

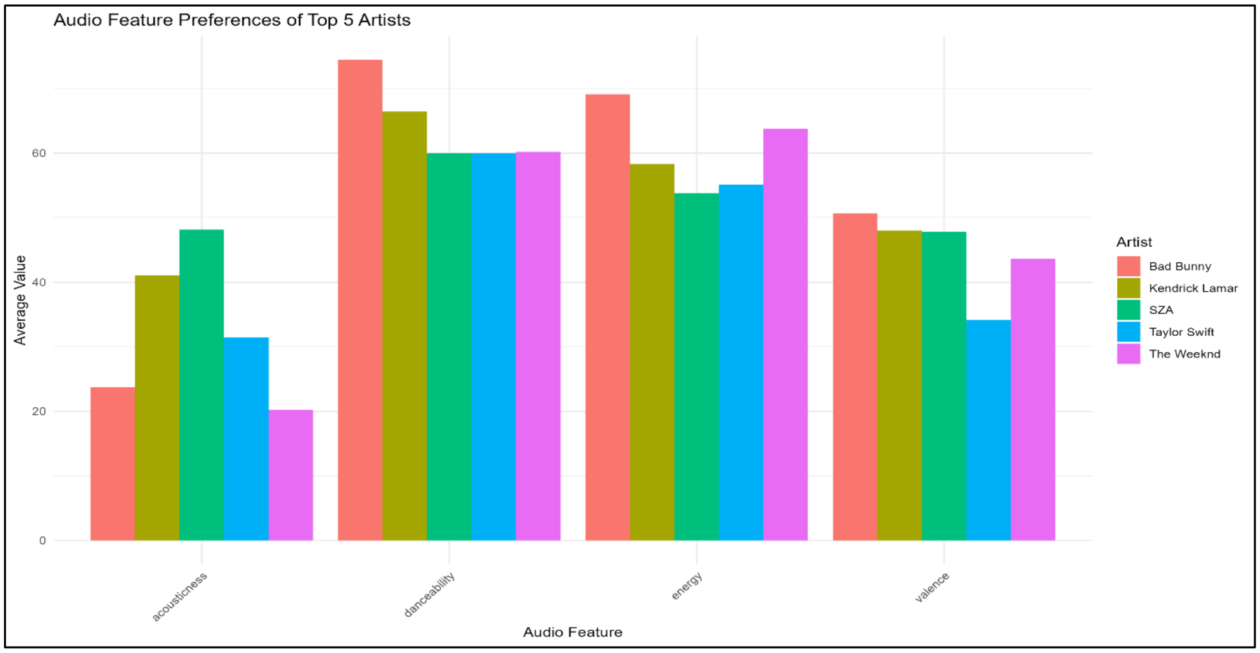

Figure 10: Visualization of top 5 artists' audio features preferences

Predictive Analysis

The knowledge of key success elements for music tracks in streaming services depends on predictive analysis. The audio performance success of songs on Spotify Apple Music and Deezer depends heavily on identification features through Decision Trees, Linear Regression, Random Forests machine learning models. Predictive model results appear through these figures to demonstrate feature significance and performance evaluation and algorithmic predictive strength.

Linear Regression

Figure 1 presents the results of Linear Regression that analyze feature significance. Artist count emerges as the key determinant for song success based on the results which indicate it stands above streaming metrics including in_deezer_charts and in_spotify_playlists. Artist engagement across different platforms together with playlist inclusion play essential roles in forecasting streaming success metrics. The factors of danceability and energy demonstrate significant impacts on streaming results because music tracks with higher levels of these characteristics yield superior streaming performance.

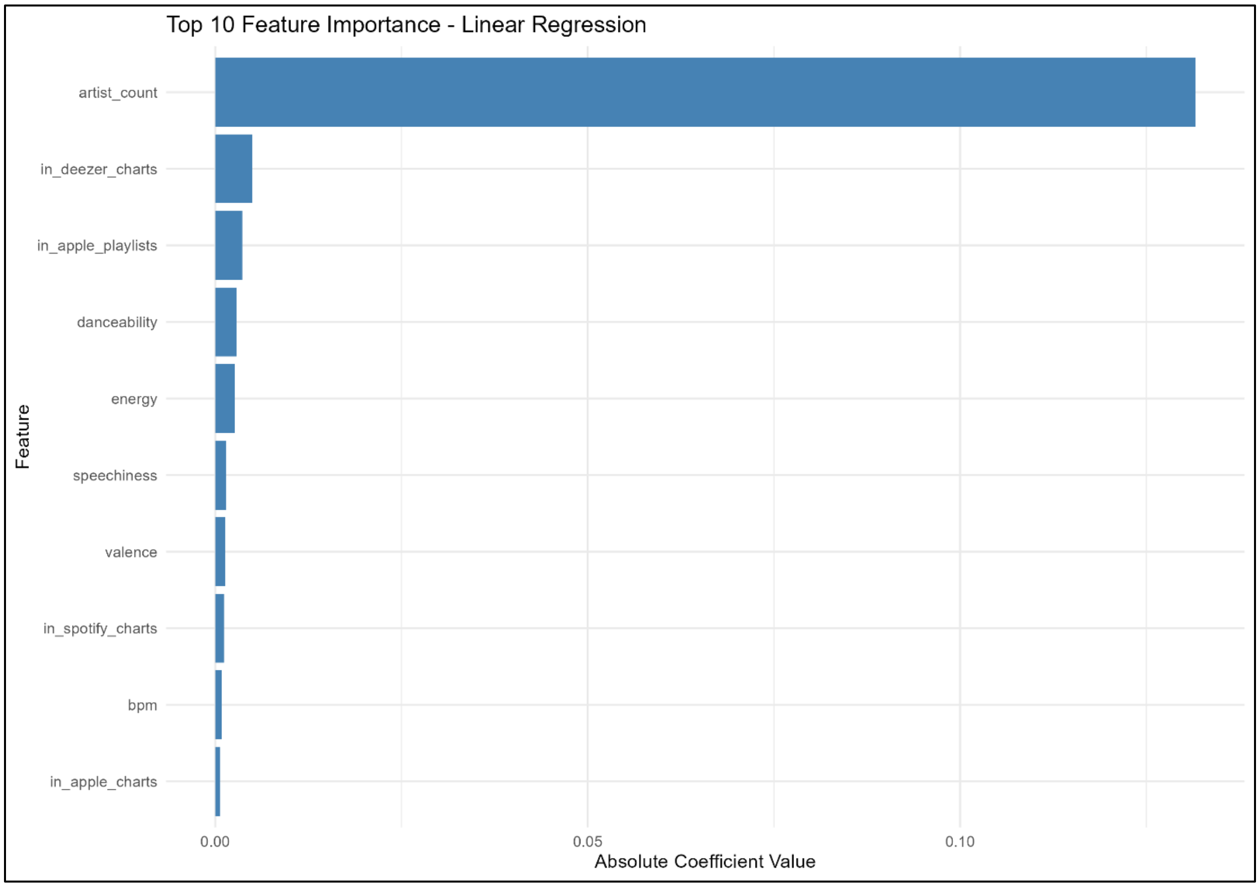

Figure 11: Feature importance for Linear regression model

Random Forest

This figure depicts a wider perspective on song performance drivers according to the Random Forest model. The Random Forest analysis displays in_spotify_playlists and in_deezer_playlists as the prime influence factors that shape the results similarly to Linear Regression. By implementing Random Forest analysis it becomes clear that while danceability and energy are important secondary factors for success prediction. Song success is significantly influenced by playlist placement together with natural attributes of individual tracks such as their danceable aspect.

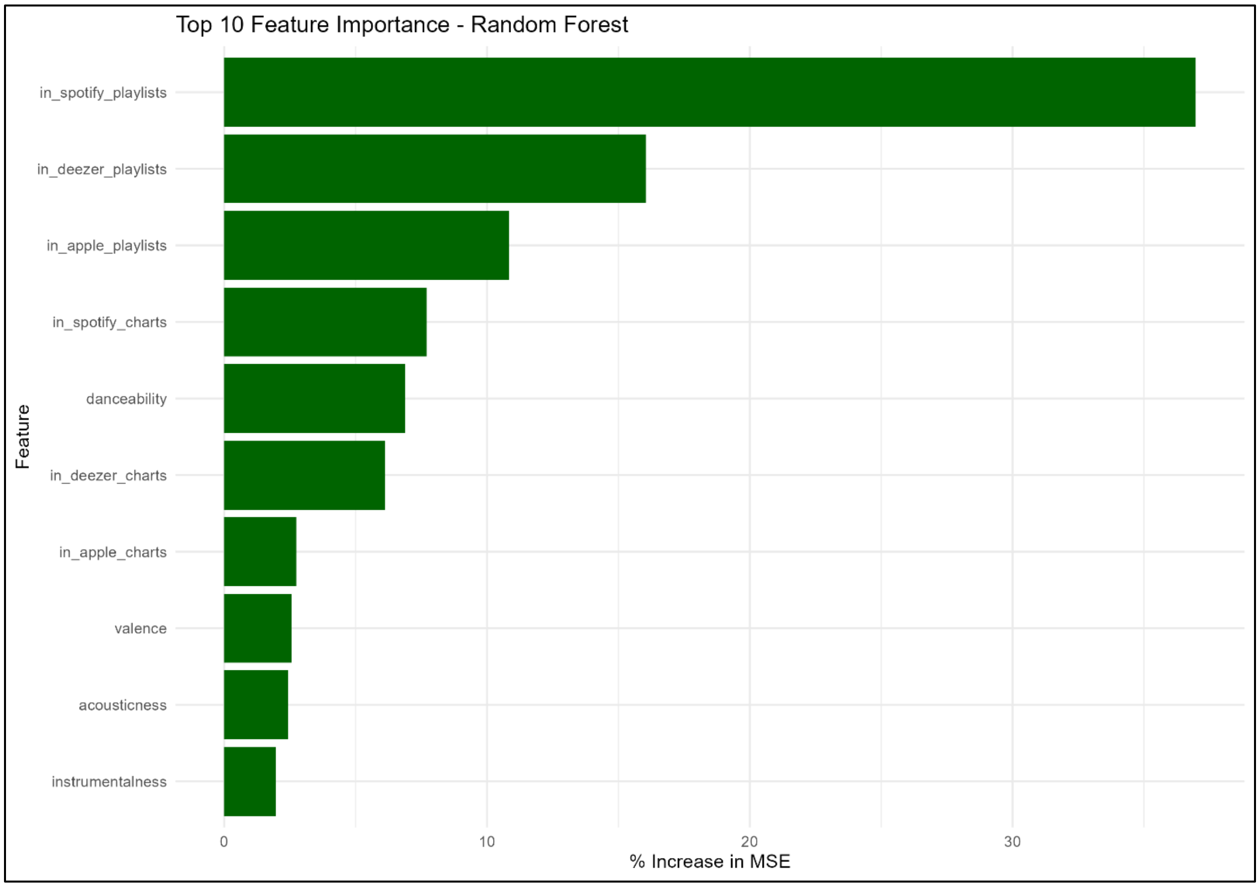

Figure 12: Feature importance for random forest model

Decision Tree

The visual representation of song success prediction via specific features appears in Figure 3 through the Decision Tree model structure. The in_spotify_playlists feature stands as the initial decision point of the tree because it defines the core factor influencing song performance. Song streaming performance improves with higher danceability ratings as well as more positive valence scores as indicated by the model splits. Users can interpret the tree-based model because it demonstrates exact defining feature values that generate high or low predicted success outcomes.

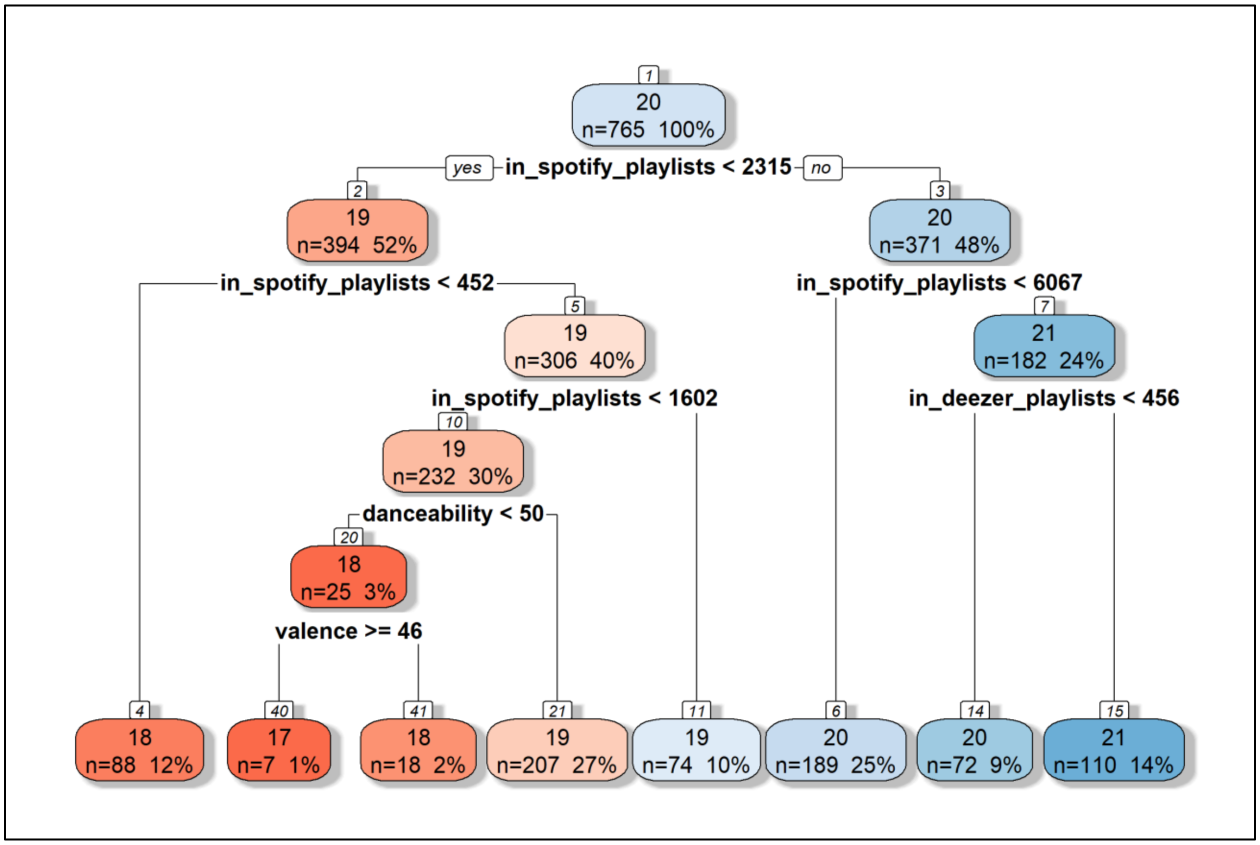

Figure 13: Visualization of Decision tree for interpretability

Model Comparison

As shown in figure 4, the metrics used for the comparison between Decision Tree, Linear Regression and Random Forest models are R-Squared and Root Mean Square Error (RMSE). R-Squared shows that the Random Forest model outperforms the Decision Tree and Linear Regression models and proves that is the most accurate model. Though simpler, the Linear Regression model provides a less complex explanation of how features are related yet is poor at estimating the complexity of interactions between the features that Random Forest presents. The Decision Tree model has both good performance and interpretability, a little bit less than Random Forest on performance.

Model Performance Visualization

The model performance is visualised in Figure 5, where Random Forest has the best performance as per R-Squared and RMSE metrics, closely followed by Linear Regression. This is then further proved that Random Forest is the best model to predict the success of a song using the features given. Whilst shown acceptable Decision Tree performance, it is not as accurate as other two models. Overall, that implies that ensemble models such as Random Forests are intended to be able to capture all these complex patterns in the data and less simple are models like Linear Regression that give up a bit less of an final analysis.

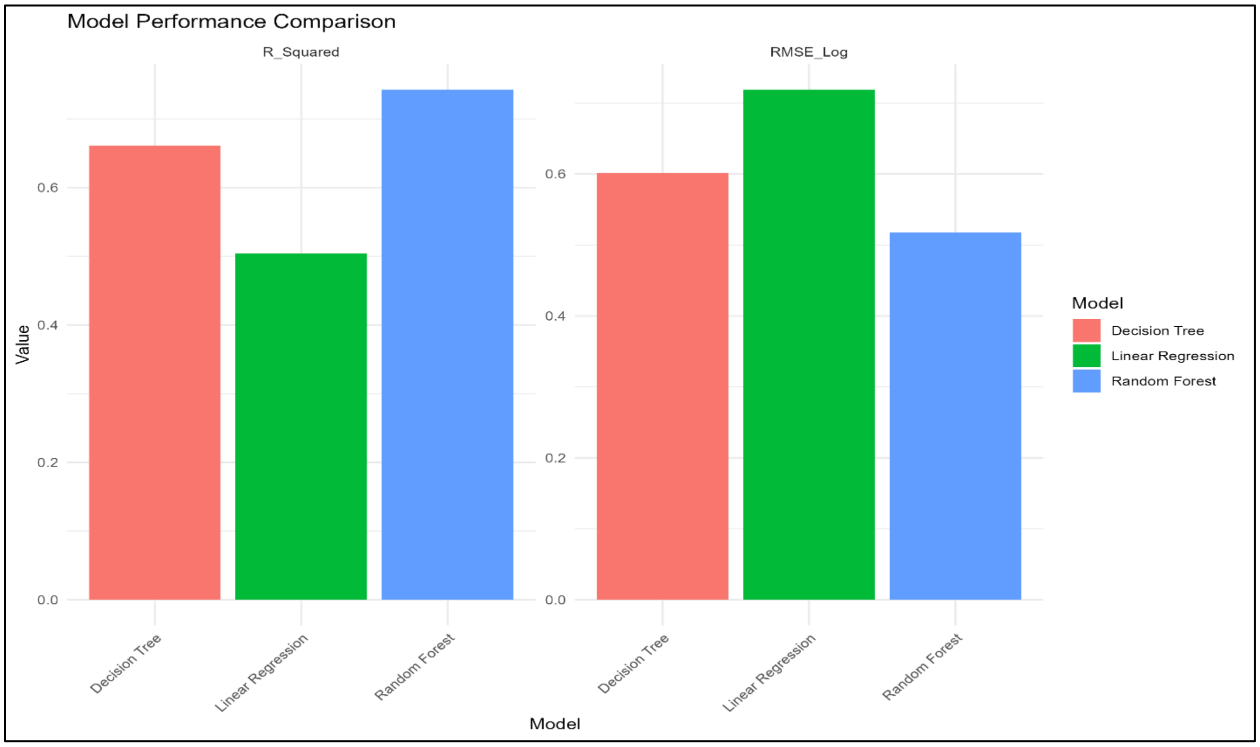

Figure 14: Models performance visualization

Conclusion

The study on Spotify's 2023 top tracks is a comprehensive analysis of the type of songs that people listen to choose from in 2023. These findings indicate that successful songs generally have high energy levels, high danceability metrics, and positive emotional tones, are made today with modern electronic techniques, and not with acoustic instruments. A consistent output, strategic collaborations and timed releases are artist impact analysis confirmed factors that affect the streaming success. As far as we know, the predictive models, especially Random Forest, did a great job in identifying playlist inclusion, artist presence in the different platforms, and audio characteristics such as danceability and energy as the strongest predictors of streaming performance. This actionable intelligence for artists, producers and music executive on how to optimize their streaming strategies in today's overcrowded digital market. Using these data-driven insights, industry professionals can apply them to scale up success by making more informed composition, production, release timing, and collaboration decisions for streaming success.

Appendix

~~~~
```{r}
# Let's import the required libraries
library(readxl)
library(dplyr)
library(ggplot2)
library(corrplot)
library(tidyr)
library(scales)
library(forcats)
```


```{r}
# Reading the dataset
spotify_data <- read_excel("C:/Users/ /Desktop
/freelancing/spotify/Spotify_2023.xlsx")


# Extract the directory path for saving plots
save_directory <- dirname("C:/Users /Desktop /freelancing/spotify")
```


# ==================
# DATA PREPARATION
# ==================
```{r}
# Clean teh data and check for missing vlaues
spotify_clean <- spotify_data %>%
  # Converting stream cloumn to numeric form
  mutate(streams = as.numeric(gsub(",", "", streams))) %>%
  # Remving the % sign from the column names to ensure safe and acuurate analysis
~~~~

```
  rename(
    danceability = `danceability_%`,
    valence = `valence_%`,
    energy = `energy_%`,
    acousticness = `acousticness_%`,
    instrumentalness = `instrumentalness_%`,
    liveness = `liveness_%`,
    speechiness = `speechiness_%`
  )

# Now check for missing values
if(any(is.na(spotify_clean$streams))) {
  # Fill the missing values using median method
  median_streams <- median(spotify_clean$streams, na.rm = TRUE)
  spotify_clean <- spotify_clean %>%
    mutate(streams = ifelse(is.na(streams), median_streams, streams))
}
```

# ---------------------------------

# PART 1: MUSIC ANALYSIS

# ---------------------------------

```{r}
# 1. Checking and visualizing Audio features distribution
audio_features <- spotify_clean %>%
  select(danceability, valence, energy, acousticness,
         instrumentalness, liveness, speechiness)

audio_long <- audio_features %>%
  pivot_longer(cols = everything(),
```

~~~
               names_to = "feature",

               values_to = "percentage")


# Plotting boxplot for audion features

p1 <- ggplot(audio_long, aes(x = reorder(feature, percentage, FUN = median), y =
percentage)) +

  geom_boxplot(fill = "skyblue", alpha = 0.7) +

  theme_minimal() +

  labs(title = "Distribution of Audio Features in Top Spotify Songs 2023",

       x = "Audio Feature",

       y = "Percentage") +

  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p1)


# Save the plot

ggsave(file.path(save_directory, "audio_features_distribution.png"), p1, width =
10, height = 6, dpi = 300)

```



```{r}

# 2. Correlating audio features

cor_matrix <- cor(audio_features)

png(file.path(save_directory, "audio_features_correlation.png"), width = 1800,
height = 1600, res = 300)

corrplot(cor_matrix, method = "circle", type = "upper",

         tl.col = "black", tl.srt = 45,

         title = "Correlation Between Audio Features")

dev.off()

ggsave(file.path(save_directory, "audio_features_distribution.png"), p1, width =
10, height = 6, dpi = 300)

```
~~~

~~~~
```{r}
# 3. Checking BPM distribution
p2 <- ggplot(spotify_clean, aes(x = bpm)) +
  geom_histogram(bins = 30, fill = "darkgreen", alpha = 0.7) +
  theme_minimal() +
  labs(title = "Distribution of BPM in Popular Songs",
       x = "Beats Per Minute (BPM)",
       y = "Count")
print(p2)

# Save the plot
ggsave(file.path(save_directory, "bpm_distribution.png"), p2, width = 10, height
= 6, dpi = 300)
```


```{r}
# 4. Analyzing mode and key distribution
key_mode_count <- spotify_clean %>%
  group_by(key, mode) %>%
  summarise(count = n(), .groups = "drop") %>%
  arrange(desc(count))

p3 <- ggplot(key_mode_count, aes(x = reorder(key, count), y = count, fill =
mode)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Distribution of Songs by Key and Mode",
       x = "Key",
       y = "Count",
       fill = "Mode") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
~~~~

```
print(p3)


# Save the plot

ggsave(file.path(save_directory, "key_mode_distribution.png"), p3, width = 10, 
height = 6, dpi = 300)

```

```{r}

# 5. Visualizing the most streamed songs

# First filter 20 most streamed songs

top_songs <- spotify_clean %>%

  arrange(desc(streams)) %>%

  head(20)


# Take the average of audion features for top songs

top_features <- top_songs %>%

  summarise(across(c(danceability, valence, energy, acousticness, 

                     instrumentalness, liveness, speechiness), mean))


all_features <- spotify_clean %>%

  summarise(across(c(danceability, valence, energy, acousticness, 

                     instrumentalness, liveness, speechiness), mean))


# Combining for comparison

feature_comparison <- bind_rows(

  pivot_longer(top_features, everything(), names_to = "feature", values_to = 
"value") %>%

    mutate(category = "Top 20 Songs"),

  pivot_longer(all_features, everything(), names_to = "feature", values_to = 
"value") %>%

    mutate(category = "All Songs")

)
```

```
p4 <- ggplot(feature_comparison, aes(x = feature, y = value, fill = category)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Audio Features: Top 20 vs All Songs",
       x = "Audio Feature",
       y = "Average Percentage",
       fill = "Category") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p4)

# Save the plot
ggsave(file.path(save_directory, "top_songs_features_comparison.png"), p4, width
= 10, height = 6, dpi = 300)
```

```
```

# ---------------------------------

# PART 2: ARTIST IMPACT

# ---------------------------------

~~~
```{r}
# 1. Analyze Artisits per song
artist_count_analysis <- spotify_clean %>%
  group_by(artist_count) %>%
  summarise(
    avg_streams = mean(streams, na.rm = TRUE),
    count = n(),
    .groups = "drop"
  ) %>%
  arrange(artist_count)
~~~

```
p5 <- ggplot(artist_count_analysis, aes(x = factor(artist_count), y =
avg_streams)) +

  geom_bar(stat = "identity", fill = "purple", alpha = 0.7) +

  theme_minimal() +

  labs(title = "Average Streams by Number of Artists on a Track",

       x = "Number of Artists",

       y = "Average Streams")

print(p5)


# Save the plot

ggsave(file.path(save_directory, "artist_count_analysis.png"), p5, width = 10,
height = 6, dpi = 300)
```
~~~
```{r}

# 2. Getting top artists by number of songs

top_artists <- spotify_clean %>%

  separate_rows(`artist(s)_name`, sep = ", ") %>%

  group_by(`artist(s)_name`) %>%

  summarise(

    song_count = n(),

    total_streams = sum(streams, na.rm = TRUE),

    avg_streams_per_song = mean(streams, na.rm = TRUE),

    .groups = "drop"

  ) %>%

  arrange(desc(song_count)) %>%

  head(15)


p6 <- ggplot(top_artists, aes(x = reorder(`artist(s)_name`, song_count), y =
song_count)) +

  geom_bar(stat = "identity", fill = "darkblue", alpha = 0.7) +

  theme_minimal() +
~~~

~~~~
  coord_flip() +
  labs(title = "Top 15 Artists by Number of Songs in Top Spotify 2023",
       x = "Artist",
       y = "Number of Songs")
print(p6)

# Save the plot
ggsave(file.path(save_directory, "top_artists_song_count.png"), p6, width = 10,
height = 8, dpi = 300)
```

```{r}
# 3. Total stream and top artists analysis
top_artists_streams <- spotify_clean %>%
  separate_rows(`artist(s)_name`, sep = ", ") %>%
  group_by(`artist(s)_name`) %>%
  summarise(
    song_count = n(),
    total_streams = sum(streams, na.rm = TRUE),
    avg_streams_per_song = mean(streams, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(total_streams)) %>%
  head(15)

p7 <- ggplot(top_artists_streams, aes(x = reorder(`artist(s)_name`,
total_streams), y = total_streams/1e9)) +
  geom_bar(stat = "identity", fill = "darkred", alpha = 0.7) +
  theme_minimal() +
  coord_flip() +
  labs(title = "Top 15 Artists by Total Streams (Billions)",
~~~~

~~~
       x = "Artist",
       y = "Total Streams (Billions)")
print(p7)

# Save the plot
ggsave(file.path(save_directory, "top_artists_streams.png"), p7, width = 10,
height = 8, dpi = 300)
```

```{r}
# 4. Audio features preferences analysis
top5_artists <- top_artists$`artist(s)_name`[1:5]

# Create a dataset for the top 5 artists
top_artists_features <- spotify_clean %>%
  filter(grepl(paste(top5_artists, collapse = "|"), `artist(s)_name`)) %>%
  mutate(main_artist = case_when(
    grepl(top5_artists[1], `artist(s)_name`) ~ top5_artists[1],
    grepl(top5_artists[2], `artist(s)_name`) ~ top5_artists[2],
    grepl(top5_artists[3], `artist(s)_name`) ~ top5_artists[3],
    grepl(top5_artists[4], `artist(s)_name`) ~ top5_artists[4],
    grepl(top5_artists[5], `artist(s)_name`) ~ top5_artists[5],
    TRUE ~ "Other"
  )) %>%
  group_by(main_artist) %>%
  summarise(
    avg_danceability = mean(danceability),
    avg_energy = mean(energy),
    avg_valence = mean(valence),
    avg_acousticness = mean(acousticness),
    .groups = "drop"
~~~

~~~~
  )


top_artists_features_long <- top_artists_features %>%

  pivot_longer(cols = starts_with("avg_"),

               names_to = "feature",

               values_to = "value") %>%

  mutate(feature = gsub("avg_", "", feature))


p8 <- ggplot(top_artists_features_long, aes(x = feature, y = value, fill =
main_artist)) +

  geom_bar(stat = "identity", position = "dodge") +

  theme_minimal() +

  labs(title = "Audio Feature Preferences of Top 5 Artists",

       x = "Audio Feature",

       y = "Average Value",

       fill = "Artist") +

  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p8)


# Save the plot

ggsave(file.path(save_directory, "top_artists_feature_preferences.png"), p8,
width = 12, height = 7, dpi = 300)

```


```{r}

# 5. Release Timing Strategy by Artist Count

release_timing <- spotify_clean %>%

  group_by(artist_count, released_month) %>%

  summarise(

    song_count = n(),

    avg_streams = mean(streams, na.rm = TRUE),
~~~~

```
    .groups = "drop"
  )


p9 <- ggplot(release_timing, aes(x = factor(released_month), y = song_count, fill
= factor(artist_count))) +
  geom_bar(stat = "identity", position = "stack") +
  scale_x_discrete(labels = month.abb) +
  theme_minimal() +
  labs(title = "Release Month Strategy by Number of Artists",
       x = "Month",
       y = "Number of Songs",
       fill = "Artist Count")
print(p9)


# Save the plot
ggsave(file.path(save_directory, "release_timing_strategy.png"), p9, width = 12,
height = 7, dpi = 300)
```

```{r}
# 6.  Anlayzing success by collaborations vs solo work
collab_vs_solo <- spotify_clean %>%
  mutate(is_collab = ifelse(artist_count > 1, "Collaboration", "Solo")) %>%
  group_by(is_collab) %>%
  summarise(
    avg_streams = mean(streams, na.rm = TRUE),
    median_streams = median(streams, na.rm = TRUE),
    song_count = n(),
    avg_spotify_playlists = mean(in_spotify_playlists),
    avg_spotify_charts = mean(in_spotify_charts),
    avg_apple_playlists = mean(in_apple_playlists),
    avg_apple_charts = mean(in_apple_charts),
    .groups = "drop"
  )

collab_metrics <- collab_vs_solo %>%
  select(is_collab, avg_streams, median_streams, avg_spotify_playlists,
avg_spotify_charts) %>%
  pivot_longer(cols = -is_collab, names_to = "metric", values_to = "value") %>%
  mutate(
    metric = case_when(
      metric == "avg_streams" ~ "Average Streams (millions)",
      metric == "median_streams" ~ "Median Streams (millions)",
      metric == "avg_spotify_playlists" ~ "Avg Spotify Playlists",
      metric == "avg_spotify_charts" ~ "Avg Spotify Charts"
    ),
    value = ifelse(grepl("Streams", metric), value / 1e6, value)
  )

p10 <- ggplot(collab_metrics, aes(x = metric, y = value, fill = is_collab)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Performance Metrics: Collaborations vs Solo Tracks",
       x = "Metric",
       y = "Value",
       fill = "Track Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p10)

# Save the plot
ggsave(file.path(save_directory, "collab_vs_solo_performance.png"), p10, width =
12, height = 7, dpi = 300)
```

~~~
```

# --------------------------------
# PART 3: PREDICTIVE ANALYTICS
# --------------------------------
```{r}
# Preparing data for modelling
spotify_model_data <- spotify_clean %>%
  # Selecting features for prediction
  select(
    streams, artist_count, in_spotify_playlists, in_spotify_charts,
    in_apple_playlists, in_apple_charts, in_deezer_playlists, in_deezer_charts,
    bpm, danceability, valence, energy, acousticness,
    instrumentalness, liveness, speechiness
  ) %>%
  # Removing null values
  na.omit()

# Transforming steams to log scale
spotify_model_data <- spotify_model_data %>%
  mutate(log_streams = log(streams + 1))

# Splitting dataset for training and testing
set.seed(123) # For reproducibility
training_indices <- createDataPartition(spotify_model_data$log_streams, p = 0.8,
list = FALSE)
train_data <- spotify_model_data[training_indices, ]
test_data <- spotify_model_data[-training_indices, ]
```

```{r}
~~~

```
# 1. Linear Regression Model

# Train the model

lm_model <- lm(log_streams ~ ., data = train_data %>% select(-streams))


# Summarize the model

lm_summary <- summary(lm_model)

print(lm_summary)


# Extract variable importance

lm_importance <- data.frame(

  Feature = names(lm_model$coefficients)[-1],

  Importance = abs(lm_model$coefficients[-1])

) %>%

  arrange(desc(Importance)) %>%

  head(10)


# Visualize feature importance for Linear Regression

p11 <- ggplot(lm_importance, aes(x = reorder(Feature, Importance), y =
Importance)) +

  geom_bar(stat = "identity", fill = "steelblue") +

  coord_flip() +

  theme_minimal() +

  labs(title = "Top 10 Feature Importance - Linear Regression",

       x = "Feature",

       y = "Absolute Coefficient Value")

print(p11)


# Save the plot

ggsave(file.path(save_directory, "lm_feature_importance.png"), p11, width = 10,
height = 7, dpi = 300)
```

~~~~
# Make predictions
lm_predictions <- predict(lm_model, newdata = test_data)

# Calculate RMSE
lm_rmse <- sqrt(mean((lm_predictions - test_data$log_streams)^2))
cat("Linear Regression RMSE (log scale):", lm_rmse, "\n")
cat("Linear Regression RMSE (original scale):", exp(lm_rmse), "\n")

# Calculate R-squared for test data
lm_r2 <- cor(lm_predictions, test_data$log_streams)^2
cat("Linear Regression R-squared:", lm_r2, "\n")
```

```{r}
# 2. Random Forest Model
# Train the model
rf_model <- randomForest(
  log_streams ~ .,
  data = train_data %>% select(-streams),
  ntree = 500,
  importance = TRUE
)

# Print model summary
print(rf_model)

# variable importance
rf_importance <- importance(rf_model) %>%
  as.data.frame() %>%
  mutate(Feature = rownames(.)) %>%
~~~~

```
  arrange(desc(`%IncMSE`)) %>%
  head(10)

# Visualize feature importance
p12 <- ggplot(rf_importance, aes(x = reorder(Feature, `%IncMSE`), y = `%IncMSE`))
+
  geom_bar(stat = "identity", fill = "darkgreen") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Top 10 Feature Importance - Random Forest",
       x = "Feature",
       y = "% Increase in MSE")
print(p12)

# Save the plot
ggsave(file.path(save_directory, "rf_feature_importance.png"), p12, width = 10,
height = 7, dpi = 300)

# Make predictions on test data
rf_predictions <- predict(rf_model, newdata = test_data)

# Calculate RMSE
rf_rmse <- sqrt(mean((rf_predictions - test_data$log_streams)^2))
cat("Random Forest RMSE (log scale):", rf_rmse, "\n")
cat("Random Forest RMSE (original scale):", exp(rf_rmse), "\n")

# Calculate R-squared for test data
rf_r2 <- cor(rf_predictions, test_data$log_streams)^2
cat("Random Forest R-squared:", rf_r2, "\n")
```

~~~~
```{r}
# 3. Decision Tree Model (for interpretability)
# Train the model
dt_model <- rpart(
  log_streams ~ .,
  data = train_data %>% select(-streams),
  method = "anova",
  control = rpart.control(maxdepth = 6)
)

# Plot the decision tree
png(file.path(save_directory, "decision_tree_plot.png"), width = 2400, height =
1600, res = 300)
rpart.plot(dt_model, extra = 101, box.palette = "RdBu", shadow.col = "gray", nn =
TRUE)
dev.off()

# Make predictions on test data
dt_predictions <- predict(dt_model, newdata = test_data)

# Calculate RMSE
dt_rmse <- sqrt(mean((dt_predictions - test_data$log_streams)^2))
cat("Decision Tree RMSE (log scale):", dt_rmse, "\n")
cat("Decision Tree RMSE (original scale):", exp(dt_rmse), "\n")

# Calculate R-squared for test data
dt_r2 <- cor(dt_predictions, test_data$log_streams)^2
cat("Decision Tree R-squared:", dt_r2, "\n")
```

```{r}
~~~~

```
# 4. Model Comparison
model_comparison <- data.frame(
  Model = c("Linear Regression", "Random Forest", "Decision Tree"),
  RMSE_Log = c(lm_rmse, rf_rmse, dt_rmse),
  RMSE_Original = c(exp(lm_rmse), exp(rf_rmse), exp(dt_rmse)),
  R_Squared = c(lm_r2, rf_r2, dt_r2)
)

print(model_comparison)

# Visualize model comparison
model_comparison_long <- model_comparison %>%
  select(Model, RMSE_Log, R_Squared) %>%
  pivot_longer(cols = c(RMSE_Log, R_Squared),
               names_to = "Metric",
               values_to = "Value")

p13 <- ggplot(model_comparison_long, aes(x = Model, y = Value, fill = Model)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Metric, scales = "free_y") +
  theme_minimal() +
  labs(title = "Model Performance Comparison",
       x = "Model",
       y = "Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p13)

# Save the plot
ggsave(file.path(save_directory, "model_comparison.png"), p13, width = 10, height = 7, dpi = 300)
```