Exploring Feature Importance in Predicting Music Popularity

STATS/CSE 780 – Final Project

Jackie Zhou

2024-12-11

Abstract

The increasing availability of music data has opened new opportunities to understand what drives track popularity. However, existing research often relies on metadata, such as artist recognition and marketing, while underutilizing the rich acoustic features inherent in music itself. This study addresses this gap by investigating whether a track's acoustic characteristics, as extracted by Echonest, can accurately predict its popularity, measured by the number of times it has been listened to. Using data from the Free Music Archive (FMA), we applied dimensionality reduction through Principal Component Analysis (PCA) to summarize the eight acoustic features into five principal components (PCs), capturing over 72% of the variance. Two regression models, Random Forest and XGBoost, were then used to predict log-transformed track listens. Our results demonstrate that while the models explain only a small portion of the variance in track popularity (R^2 of 0.054 for Random Forest and 0.075 for XGBoost), they highlight key acoustic patterns associated with popularity. For instance, components emphasizing valence, danceability, tempo, and acousticness emerged as important predictors in both models. XGBoost achieved slightly better accuracy, with a lower RMSE and higher R^2, at the cost of higher computational demand. These findings suggest that while acoustic features alone may not fully explain popularity, they provide valuable insights into listener preferences and can complement metadata-based approaches. This work underscores the importance of incorporating acoustic features into music recommendation systems and highlights the need for future research to explore feature interactions and integrate additional contextual information, such as lyrics and genre metadata, to improve predictive power and interpretability.

Introduction

Music popularity is influenced by a complex interplay of factors, including not only artist recognition and marketing but also the inherent qualities of the music itself. Vast amounts of research has investigated correlates of popularity, including investigating acoustic features, to artist popularity (Lin et al. (2014); Zhang et al. (2024)). By analyzing acoustic features alongside relevant metadata, it is possible to gain insights into the characteristics that make certain songs resonate more widely with audiences.

The Free Music Archive (FMA) dataset (Defferrard et al. (2016)) is a publicly available collection designed to support music information retrieval research and other studies within the field of computational musicology. The FMA dataset includes over 106,000 tracks, each annotated with extensive metadata, such as track titles, artist names, genre classifications, and listening statistics. This makes the FMA dataset a valuable resource for examining trends and patterns in music, allowing researchers to analyze relationships between musical content, metadata, and listener behavior.

One unique feature of the FMA dataset is its inclusion of audio features extracted by Echonest (Ellis et al. (2010)), a music intelligence platform acquired by Spotify. Echonest uses algorithms to analyze tracks, providing a set of eight continuous audio features: acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo, and valence. These audio features have been widely used in MIR tasks, such as genre classification and music recommendation, as they offer insights into the sonic characteristics that differentiate musical styles and appeal to listeners.

In this study, we aim to investigate whether only a song's *musical features* (Echonest features) can accurately predict its popularity. Here, *popularity* is defined by the number of times a track has been listened to, which is encoded in the dataset for every observation. Understanding the predictive power of these musical characteristics in relation to popularity could provide valuable insights for music producers, artists, and industry professionals who aim to create music that resonates with a broad audience. Moreover, insights from this research could have practical applications in music recommendation systems, where algorithms aim to deliver tracks that match listener preferences and have a higher likelihood of popularity. By identifying key audio features that contribute to popularity, recommendation algorithms could prioritize these characteristics when suggesting music, improving listener engagement.

Methods

All data processing, analyses, and visualization was performed in R (R Core Team (2020)). The tidy dataset was saved in an R data format. During data preparation, tracks with missing Echonest features were removed. Our final dataset has a total of 9355 tracks. The 8 Echonest features are continuous numerical values, and categorical variables include artist names, genres, and popularity. The number of track listens increased exponentially, so a log transform was applied on the column, resulting in a normally distributed log number of track listens. This provided a normally distributed target variable.

Principal Components Analysis

The eight Echonest features may contain overlapping information. Using PCA for dimensionality reduction allows for combining these features to a smaller set of principal components, which capture the most variance in the data while minimizing redundancy. These components were then used as inputs for the two regression methods.

Two Approaches to Regression Modelling

Random Forest Regression: Random forest provides feature importance scores, allowing us to assess the contribution of each principal component to the prediction task. It inherently handles non-linear relationships and feature interactions. Random forest was chosen for its balance of accuracy, interpretability, and ability to handle complex interactions among features. The number of trees (`ntree = 500`; how many trees are grown in the forest) and number of features per split (`mtry = 1`; tuned using `tuneRF()`, which identifies the optimal number of features to consider at each split) were tuned.

XGBoost Regression: XGBoost is a gradient boosting algorithm that builds decision trees sequentially, each correcting the errors of the previous tree. It optimizes for the least squares loss function for regression tasks. The depth of each tree (`max.depth = 2`) was tuned iteratively by evaluating performance for different values and 3 was selected because it minimized test RMSE without overfitting. Learning rate (`eta`) was set to 0.1 and controlled how much each tree contributes to the overall model. Lastly, the optimal number of boosting iterations (`nrounds`) was determined to be 137, based on lowest test RMSE for tree depth of 3.

Model Comparison

For comparing the two models, the following criteria were used:

Accuracy

- In both models, Root Mean Squared Error (RMSE) was used on the test set to measure average prediction error magnitude

Feature Importance

- In the RF model, feature importance was determined using out-of-bag (OOB) error rates, which reflect the increase in mean squared error when a particular feature (or PC in our case) is permuted.
- In the boosting model, feature importance was derived using the Gain metric. Gain represents the relative contribution of a feature to reducing loss, similar to RF's OOB rate.

R-squared

- In both models, R^2 was used to evaluate how well each model explains the variance in the target variable (popularity). This complements RMSE by focusing on the proportion of variability accounted for by the models.

Exploratory Data Analysis

Supp. Figure 4 shows the boxplot distributions of each Echonest feature across different genres. This highlights that different musical genres exhibit distinct patterns in their acoustic characteristics.

A correlation matrix (Supp. Figure 5) shows the relationships among the 9 continuous Echonest features, duration of the track, and their associations with popularity. No single feature shows a strong correlation with popularity, though Echonest features show multicollinearity. For example, there is a correlation between energy and acousticness (-0.47) and between valence and danceability (0.44). These relationships suggest that certain musical characteristics may interact in meaningful ways, which may influence a track's potential for popularity.

Looking further at the popularity metric, Figure 1 below shows summary distributions of the number of listens a track has. The top plot illustrates the distribution of track listens, showing a highly positively skewed pattern where most tracks have low listen counts, and a few achieve high popularity. To address this, we applied a log transformation to reduce skewness and compress the data range, improving visualization and modeling by minimizing the influence of extreme values. After the transformation, the data followed a near-normal distribution (as shown on the bottom plot of Fig 1), and we further refined it by removing the most extreme 2.5% of values from both ends.

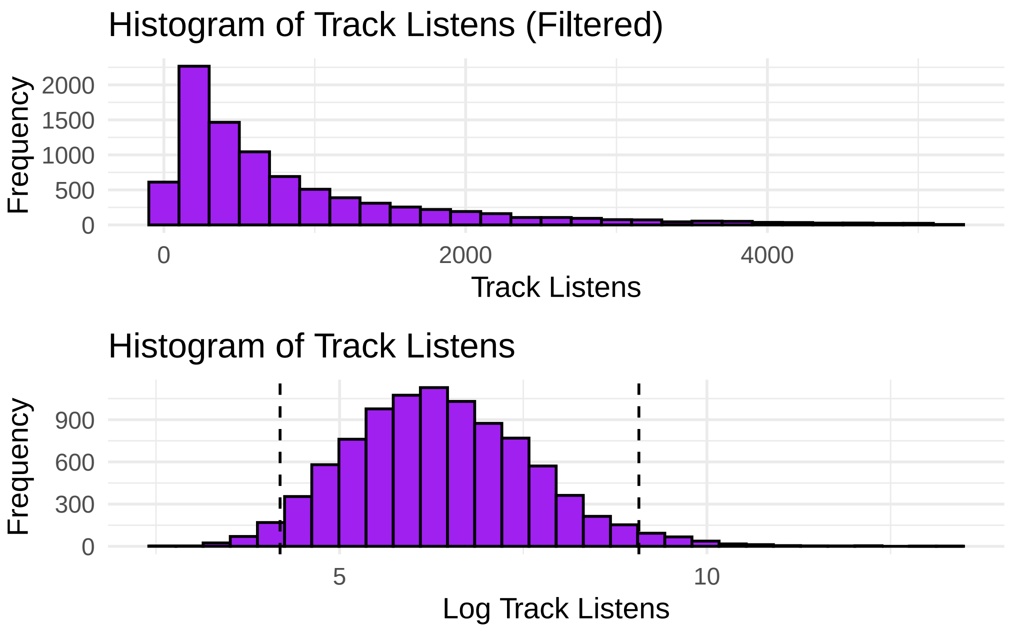


Figure 1: Top: Summary distribution of tracks by the number of times they are listened to. Filtered by the bottom 95% of tracks to visually remove outliers. Bottom: Summary distribution of tracks after log-transformation. Highest and lowest 2.5% of data cut out.

Given the moderate correlations between some features and the diversity in genre-specific attributes, we used PCA to reduce dimensionality by identifying the most significant components among the metadata and audio features. This transformation helps to capture the main variability within the data, allowing us to reduce redundancy and focus on the most informative aspects of each track's acoustic profile. The results of the PCA analysis are shown in Figure 6 (Supplementary). We selected the first five principal components for our regression models because they collectively explained over 72% of the variance in the data. Including additional PCs provided minimal incremental explanatory power, indicating diminishing returns.

Model Comparison Results

We evaluated two regression methods, Random Forest regression and regression with boosting, for predicting the logarithmic transformation of track listen counts.

Random Forest Regression

The RF model achieved an RMSE of 1.043, with an R^2 of .054 ($p<.001$). The most influential principal components were PC1 and PC2, suggesting that these components significantly contributed to the model's predictive accuracy. PC1 displayed the highest importance, contributing over 43% to the model based on the % Increase in MSE (see Figure 7 in Supplementary).

Regression with Boosting

The XGBoost regression model had an RMSE of 1.026 and an R^2 of 0.075($p<.001$). The most important features were PC1 and PC4 with the highest Gain values above all other PCs (see Figure 8 in Supplementary).

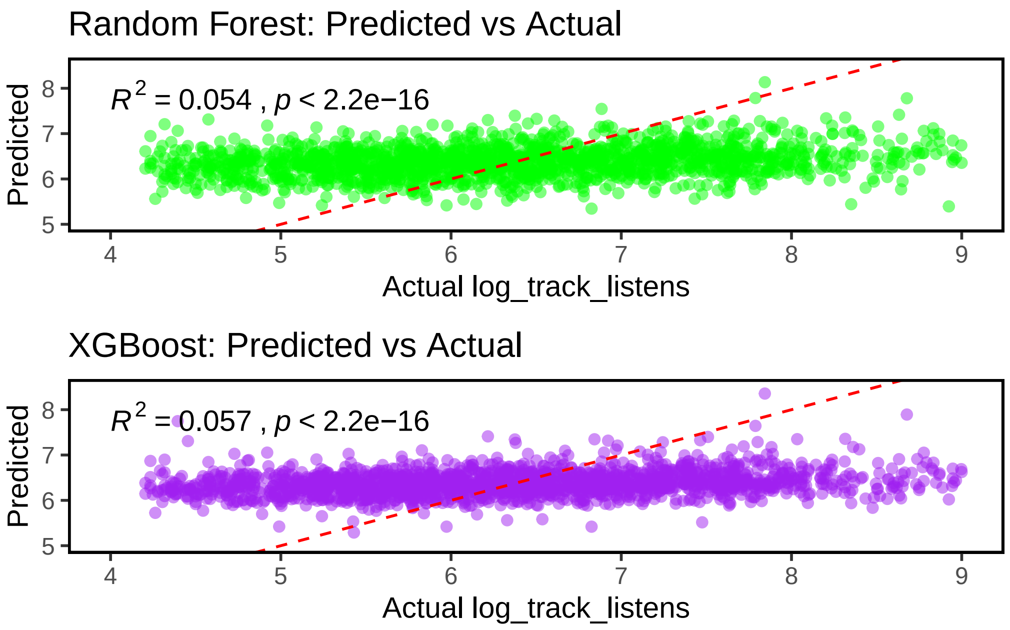


Figure 2: Top: Scatterplot of predicted and actual values of log_track_listens from Random Forest Regression Model. Bottom: Same scatterplot but with values from the Boosting Regression Model

Discussion

XGBoost showed better predictive accuracy compared to RF, as evidenced by a lower RMSE (1.026 vs. 1.043) and a higher R^2 (0.075 vs. 0.054). This suggests that XGBoost was better equipped to handle the complexities and nonlinear relationships within the data. However, both methods agreed on the significant role of PC1 in driving predictions, highlighting its critical role in capturing key information about the tracks. The differences in the ranking of subsequent PCs (e.g., PC2 for RF and PC4 for XGBoost) may reflect the different ways these models handle feature interactions and nonlinearity. These results suggest that XGBoost may be preferable for this dataset due to its superior performance metrics, but Random Forest still provides valuable insights into feature importance and model interpretability.

Insights on Feature Loadings

Looking at the results, the relatively high RMSE and low rsquared for both models suggest that acoustic features alone cannot highly predict music preference. However, since there was enough power to still observe a small significant association between PCs and popularity, we can start to break that down. Analyzing the PC loadings allow us to get a sense of what type of music each PC correspond to. PC loadings are shown in Figure 3 below. The first principal component (PC1) consistently captures features related to low valence, low danceability, and high acousticness. This suggests that listeners' preferences for music with fewer acoustics and slightly positive emotional characteristics strongly influence listen counts. Random Forest emphasized PC2 (high energy, fast tempo, and low acousticness) as the next most significant component, while XGBoost highlighted PC4 (fast tempo, high acousticness, and short duration). These differences illustrate how the models prioritize features differently due to their respective algorithms.

The models confirm that features such as valence, danceability, acousticness, and tempo play a central role in determining track popularity. Random Forest suggests that listeners prefer music with moderate energy and more positive valence, while XGBoost emphasizes tempo and short duration as critical predictors. These findings underscore the importance of not only the acoustic properties of the music but also its emotional and temporal characteristics in influencing listener behavior.

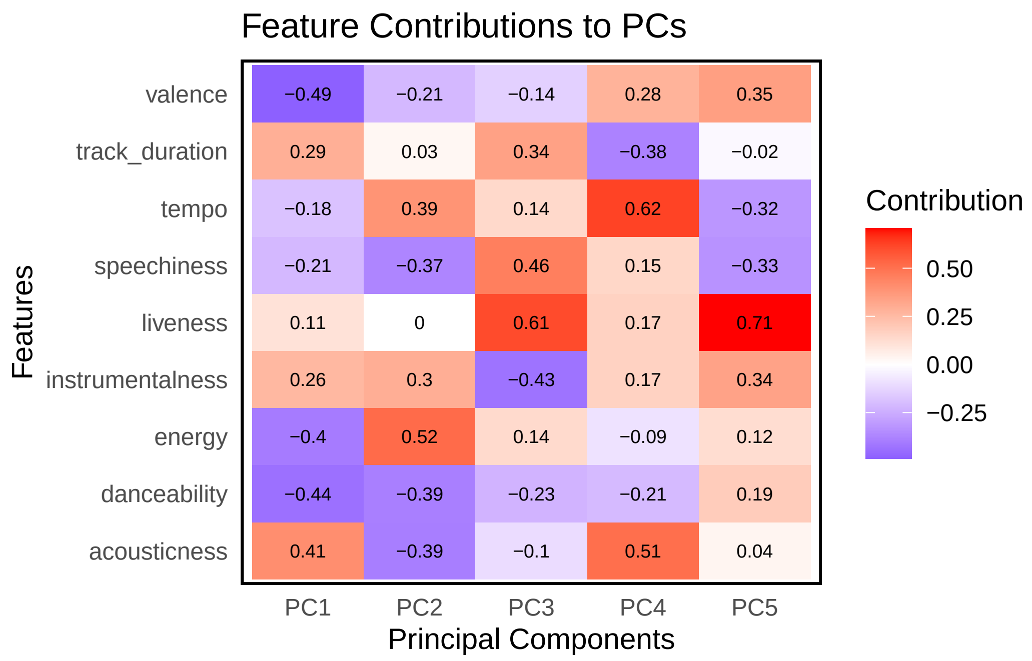


Figure 3: Heatmap of the acoustic feature loadings (predictors) for each Principal Component. Colours represent magnitude of effect.

Challenges

One major analytical challenge was managing the inherent imbalance and skewness in the dataset. The raw `track_listens` distribution was highly positively skewed, making it difficult to model directly. To address this, a log transformation was applied to normalize the data. While this transformation improved the interpretability and stability of the models, it also introduced complexities in interpreting the predictions in their original scale. Another challenge was determining the optimal number of principal components (PCs) to include in the regression models. While the first five PCs explained over 72% of the variance, adding more PCs showed diminishing returns, and their interpretability became more abstract. This tradeoff between variance explained and interpretability was a key consideration in the analysis.

The XGBoost model, while more accurate than the Random Forest, came with a higher computational cost due to the iterative nature of boosting and the hyperparameter tuning process. Each boosting round required re-evaluation of the residuals, increasing runtime, especially when cross-validation was used to identify the best parameters. In contrast, the Random Forest model was faster to train but less precise in capturing the nonlinear interactions in the data.

Both models provided useful insights, but their interpretability differed. The Random Forest regression produced intuitive feature importance scores based on out-of-bag error rates, making it easier to identify key predictors. On the other hand, XGBoost provided gain-based importance scores, which required additional interpretation due to the complexity of the boosting algorithm. Additionally, the PCA transformation used for dimensionality reduction abstracted the features, requiring careful linking of components to original variables for meaningful interpretation. To ensure reproducibility, we did separate data preprocessing and used specific seeds to ensure stable results across runs.

Future work could focus on incorporating additional musical features or metadata, such as genre or lyrics, to improve the models' predictive power. Exploring other machine learning techniques, such as ensemble methods combining boosting and bagging or deep learning approaches, could also yield more accurate and robust models. Moreover, further analysis could investigate interactions between features, such as how acousticness and tempo jointly influence listen counts, to provide deeper insights into listener behavior. Finally, it would also be intriguing to apply the two regression models directly to the raw acoustic features without using dimensionality reduction. While this approach would enhance interpretability, it could potentially result in lower accuracy due to the multicollinearity among the features.

References

Defferrard, Michaël, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2016. "FMA: A Dataset for Music Analysis." *arXiv Preprint arXiv:1612.01840*.

Ellis, Daniel PW, Brian Whitman, Tristan Jehan, and Paul Lamere. 2010. "The Echo Nest Musical Fingerprint."

Lin, Ning, Ping-Chia Tsai, Yu-An Chen, and Homer H Chen. 2014. "Music Recommendation Based on Artist Novelty and Similarity." In *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, 1–6. IEEE.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Zhang, Jurui, Shan Yu, Raymond Liu, Guang-Xin Xie, and Leon Zurawicki. 2024. "Unveiling the Melodic Matrix: Exploring Genre-and-Audio Dynamics in the Digital Music Popularity Using Machine Learning Techniques." *Marketing Intelligence & Planning*.

Supplementary Materials

```r
# FMA Dataset
# https://github.com/mdeff/fma
# Pull Data
temp <- paste(tempfile(), ".zip", sep = "")
options(timeout = 60 * 10)
download.file("https://os.unil.cloud.switch.ch/fma/fma_metadata.zip", temp)

# Feature Data
# Consolidate multiline header
echonest_colnames <- unz(temp, "fma_metadata/echonest.csv") %>%
  read_csv(n_max = 0, skip = 2) %>%
  rename(track_ID = "...1") %>%
  names()
# Read the data
echonest_raw <- unz(temp, "fma_metadata/echonest.csv") %>%
  read_csv(skip = 4, col_names = echonest_colnames) %>%
  # Transform track_ID to integer for tibble merging
  mutate(track_ID = as.integer(track_ID))

# Remove temporal features
echonest <- echonest_raw[, c(1:26)]

# Metadata
# Consolidate multiline header
metadata_colnames_a <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(n_max = 0, skip = 1) %>%
  rename(track_ID = `...1`) %>%
  names() %>%
  # Removing strange encoding
  sub("\\...*", "", .)
```

```r
metadata_colnames_b <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(n_max = 0) %>%
  names() %>%
  # Removing strange encoding
  sub("\\...*", "", .)

metadata_colnames <- paste(metadata_colnames_b, metadata_colnames_a, sep = "_")

# Read the data
metadata_raw <- unz(temp, "fma_metadata/tracks.csv") %>%
  read_csv(skip = 3, col_names = metadata_colnames) %>%
  rename(track_ID = `_track_ID`) %>%
  # Transform track_ID to integer for tibble merging
  mutate(track_ID = as.integer(track_ID))
# Combine the data and metadata
data <- inner_join(metadata_raw, echonest, by = "track_ID")

# Genre data
genres <- unz(temp, "fma_metadata/genres.csv") %>%
  read_csv()

# Clean up downloaded files
unlink(temp)

# Tidy Data
df_tidy <- data %>%
  select(c("track_ID", "artist_id", "artist_name.x",
           "track_duration", "track_genre_top", "track_genres",
           "track_listens", "acousticness", "danceability", "energy",
           "instrumentalness", "liveness", "speechiness", "tempo", "valence")
         )
```

```r
# Adding names to track_genres based on genres dataset
df_tidy <- df_tidy %>%
  # Separate `track_genres` into rows (one genre per row) by removing brackets and splitting
  mutate(track_genres = str_remove_all(track_genres, "\\[|\\]")) %>%
  separate_rows(track_genres, sep = ",") %>%
  mutate(track_genres = as.integer(track_genres)) %>%
  # Join with genres.csv to get genre names
  left_join(genres, by = c("track_genres" = "genre_id")) %>%
  # Group back by track_ID and collapse genre names into a single string
  group_by(track_ID) %>%
  mutate(track_genres_named = str_c(title, collapse = ", ")) %>%
  ungroup() %>%
  # Select relevant columns and drop duplicates
  select(-title, -track_genres, -parent, -top_level, -`#tracks`) %>%
  distinct() %>%
  # Dropping rows with missing genre
  drop_na(., track_genre_top)

# Write .RData
save(df_tidy, file = "df_tidy.RData")
```

```r
# Load data
load("df_tidy.RData")

# Since track_listens is very skewed, we will log-transform it
df_tidy$log_track_listens <- log(df_tidy$track_listens)

# Removing extreme 5%
percentiles <- quantile(df_tidy$log_track_listens, probs = c(0.025, 0.975))

# Filter the data to keep only the middle 95%
```

```r
df_filtered <- df_tidy %>%
  filter(log_track_listens > percentiles[1], log_track_listens < percentiles[2])

# Normalize numeric features of filtered data
df_scaled <- df_filtered %>%
  select(track_duration, acousticness, danceability, energy, instrumentalness,
         liveness, speechiness, tempo, valence) %>%
  scale() %>%
  as.data.frame()
```

```r
# Normalize Tempo
df_tidy <- df_tidy %>%
  mutate(tempo = (tempo - min(tempo)) / (max(tempo) - min(tempo)))

# Summary of features by genre
ggplot(df_tidy %>%
         gather(feature, val, 7:14),
       aes(x = feature, y = val, colour = feature)) +
  geom_boxplot() +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  theme(legend.position = "none") +
  ylab("Value") +
  facet_wrap(~ track_genre_top, nrow = 2)
```

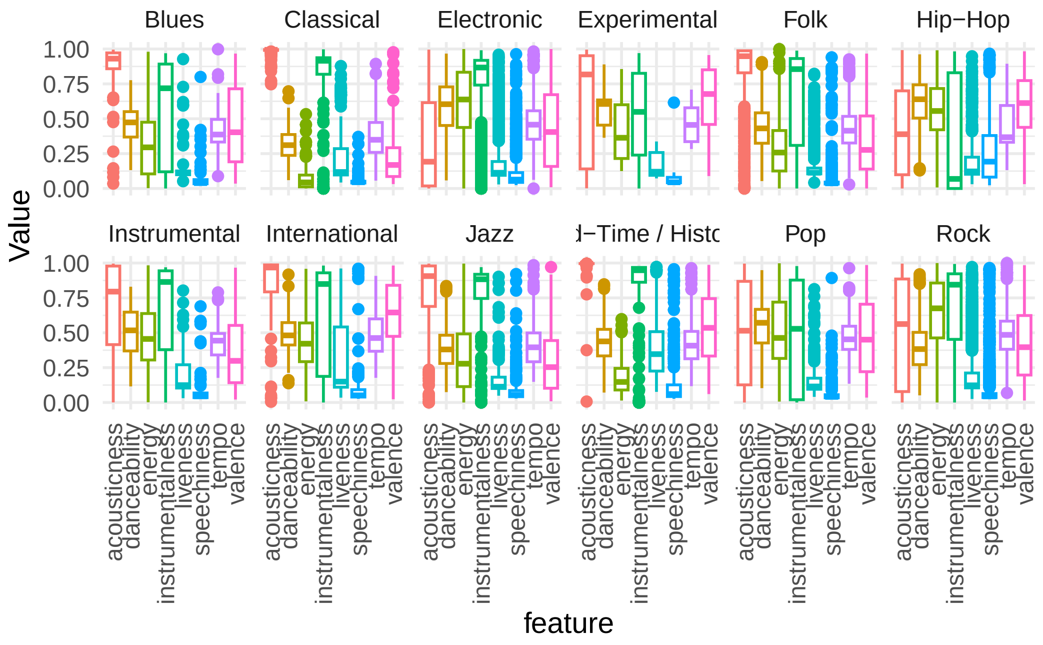
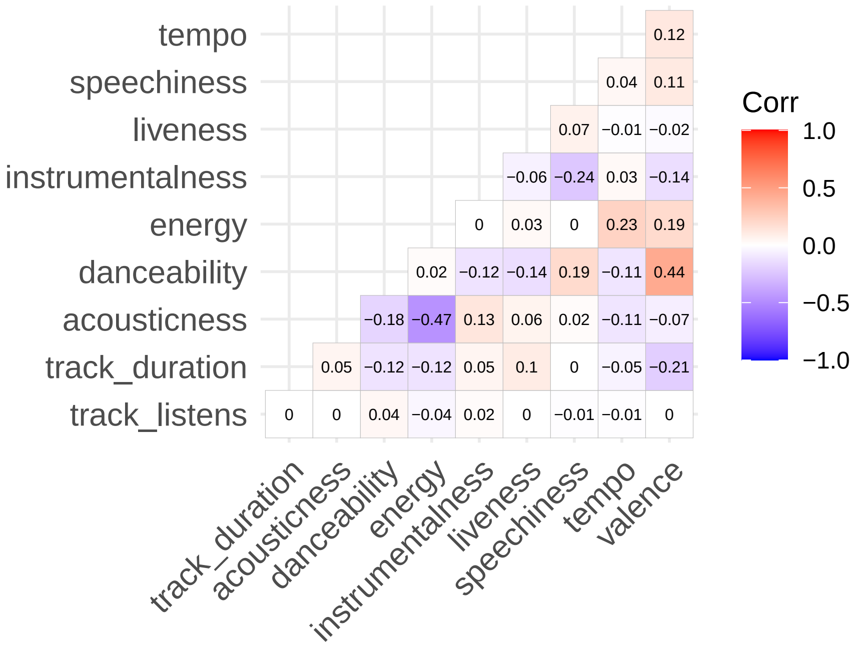


Figure 4: Summary of features by genre. Echonest features are normalized on the x-axis while mean values and boxplot summaries are displayed.

```
# Correlation matrix
numeric_features <- df_tidy %>%
  select_if(is.numeric) %>%
  select(-log_track_listens, -track_ID, -artist_id)
cor_matrix <- cor(numeric_features)

cor_features <- c("track_listens", "track_duration", "acousticness", "danceability",
                  "energy", "instrumentalness", "liveness", "speechiness", "tempo", "valence
cor_matrix <- cor_matrix[cor_features, cor_features]

ggcorrplot(cor_matrix, lab = TRUE, lab_size = 2, type = "lower")
```



	track_duration	acousticness	danceability	energy	instrumentalness	liveness	speechiness	tempo	valence
tempo									0.12
speechiness								0.04	0.11
liveness							0.07	−0.01	−0.02
instrumentalness						−0.06	−0.24	0.03	−0.14
energy					0	0.03	0	0.23	0.19
danceability				0.02	−0.12	−0.14	0.19	−0.11	0.44
acousticness			−0.18	−0.47	0.13	0.06	0.02	−0.11	−0.07
track_duration		0.05	−0.12	−0.12	0.05	0.1	0	−0.05	−0.21
track_listens	0	0	0.04	−0.04	0.02	0	−0.01	−0.01	0

Figure 5: Correlation matrix between all 8 Echonest features, duration of track, and popularity metric.

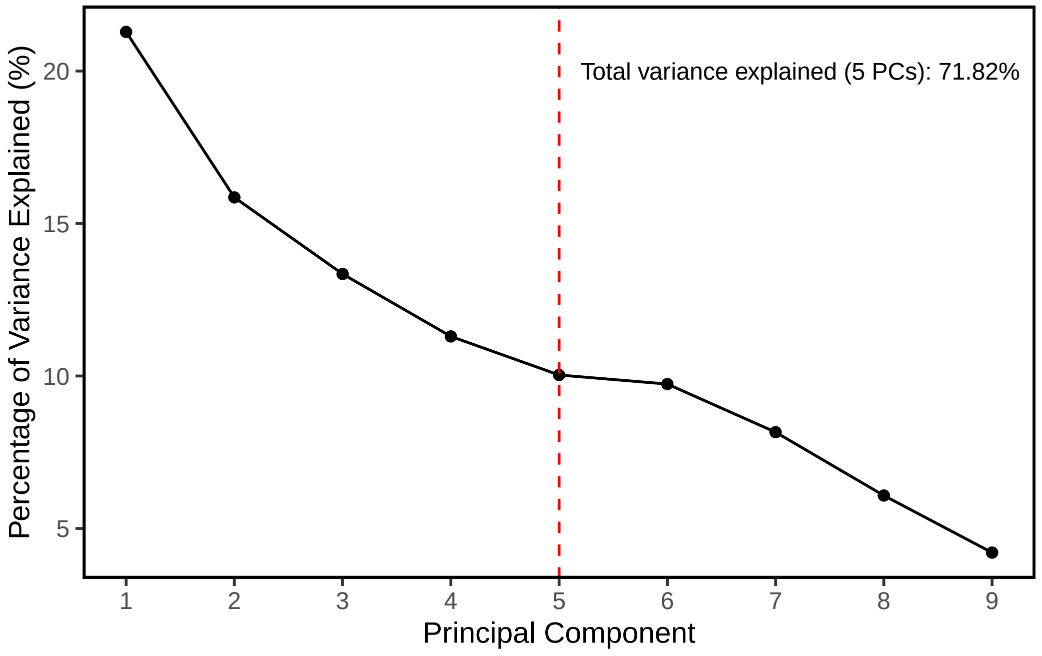


Figure 6: PCA Scree Plot displaying the cumulative percentage of variance explained by each PC

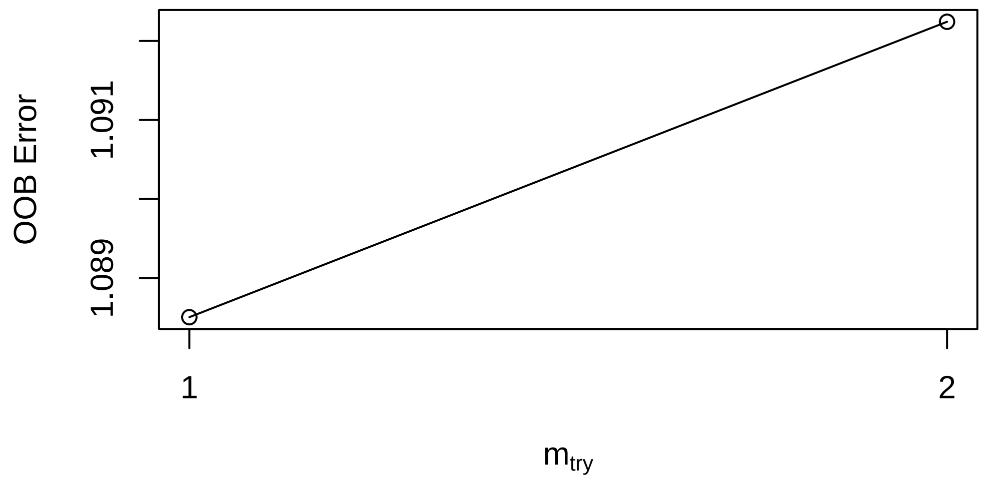
```r
# Extract the first 5 principal components
df_pca <- as.data.frame(pca_result$x[, 1:5])
```

```r
# Add the target variable to the reduced dataset
df_pca$track_listens <- df_filtered$track_listens
df_pca$log_track_listens <- df_filtered$log_track_listens

# Split train and test set -----------------------------------------------------
set.seed(123)
train_indices <- sample(1:nrow(df_pca), size = 0.8 * nrow(df_pca))
train_data <- df_pca[train_indices, ]
test_data <- df_pca[-train_indices, ]

# Random Forest Regression -----------------------------------------------------
# Use tuneRF() to refine the mtry parameter
tuned_rf <- tuneRF(
  train_data[, c("PC1", "PC2", "PC3", "PC4", "PC5")], # Predictors
  train_data$log_track_listens,                      # Target variable
  stepFactor = 2,                                    # Increment to try larger `mtry`
  ntreeTry = 500,                                    # Number of trees to try
  improve = 0.001,                                   # Minimum improvement to keep tuning
  trace = TRUE,                                      # Print progress
  plot = TRUE
)
```

```
mtry = 1  OOB error = 1.088505
Searching left ...
Searching right ...
mtry = 2    OOB error = 1.092243
-0.003434089 0.001
```



```
rf_model_1 <- randomForest(
  log_track_listens ~ PC1 + PC2 + PC3 + PC4 + PC5,
  data = train_data,
  ntree = 500,      # Number of trees
  mtry = 1,         # Number of variables to try at each split
  importance = TRUE # Track feature importance
)

# rf_model_2 <- randomForest(
#   log_track_listens ~ PC1 + PC2 + PC3 + PC4 + PC5,
#   data = train_data,
#   ntree = 500,      # Number of trees
#   mtry = 2,         # Number of variables to try at each split
#   importance = TRUE # Track feature importance
# )

test_data$rf_model_1_pred <- predict(rf_model_1, test_data)
rf_model_1_mse <- mean((test_data$rf_model_1_pred - test_data$log_track_listens)^2)
```

```r
sqrt(rf_model_1_mse)
```

```
[1] 1.052677
```

```r
rf_rmse <- sqrt(rf_model_1_mse)

# test_data$rf_model_2_pred <- predict(rf_model_2, test_data)
# rf_model_2_mse <- mean((test_data$rf_model_2_pred - test_data$log_track_listens)^2)
# sqrt(rf_model_2_mse)

# rf_model_1 (with mtry = 1) has lower RMSE

# Importance of PCA components
# Higher value for %IncMSE indicates higher importance for prediction accuracy
importance(rf_model_1) # Importance scores
```

```
     %IncMSE IncNodePurity
PC1 43.19651      1539.731
PC2 41.31020      1443.674
PC3 35.79585      1373.566
PC4 38.95853      1469.646
PC5 33.78692      1450.436
```

```r
varImpPlot(rf_model_1)  # Visualization
```

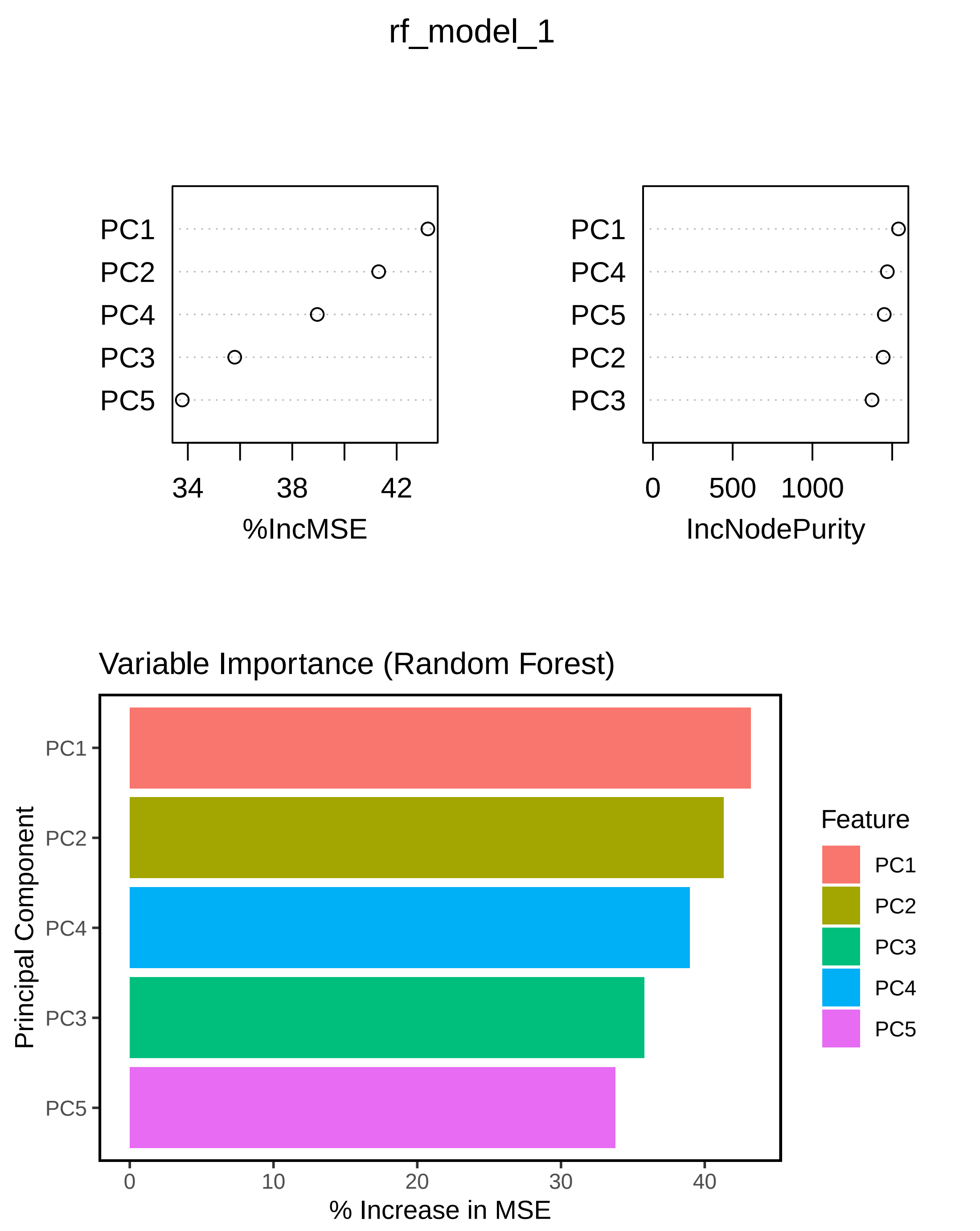
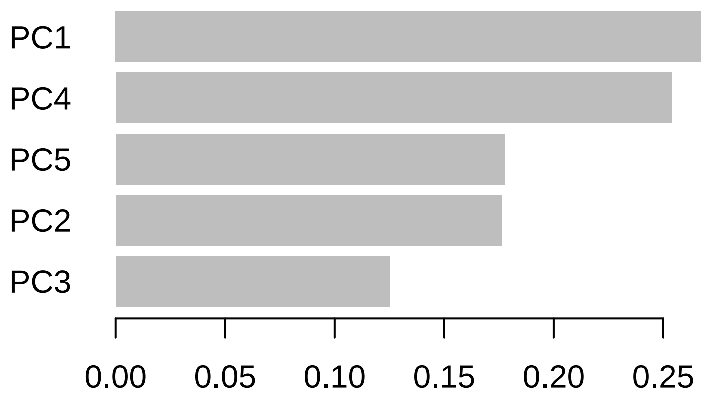


Figure 7: Random Forest Variable Importance Plot displaying the order of PC importance to the model.

```
# Train the model and monitor RMSE
set.seed(123)
model <- xgb.train(
  data = xgb_train,
  max.depth = 2,                  # Maximum tree depth
  eta = 0.1,                      # Learning rate
  nrounds = 200,                  # Total boosting rounds
  objective = "reg:squarederror", # Regression objective
  watchlist = watchlist,          # Track training and testing performance
  verbose = 1                     # Print RMSE for each round
)

min(model$evaluation_log$test_rmse)
model$evaluation_log$iter[which.min(model$evaluation_log$test_rmse)]

# Lowest Test RMSE with max_depth = 5: 1.047334 (iteration 53)
# Lowest Test RMSE with max_depth = 4: 1.04653  (iteration 91)
# Lowest Test RMSE with max_depth = 3: 1.046522 (iteration 137)
# Lowest Test RMSE with max_depth = 2: 1.048087 (iteration 186)
# Use max_depth = 3 for lower test RMSE but without overfitting
```



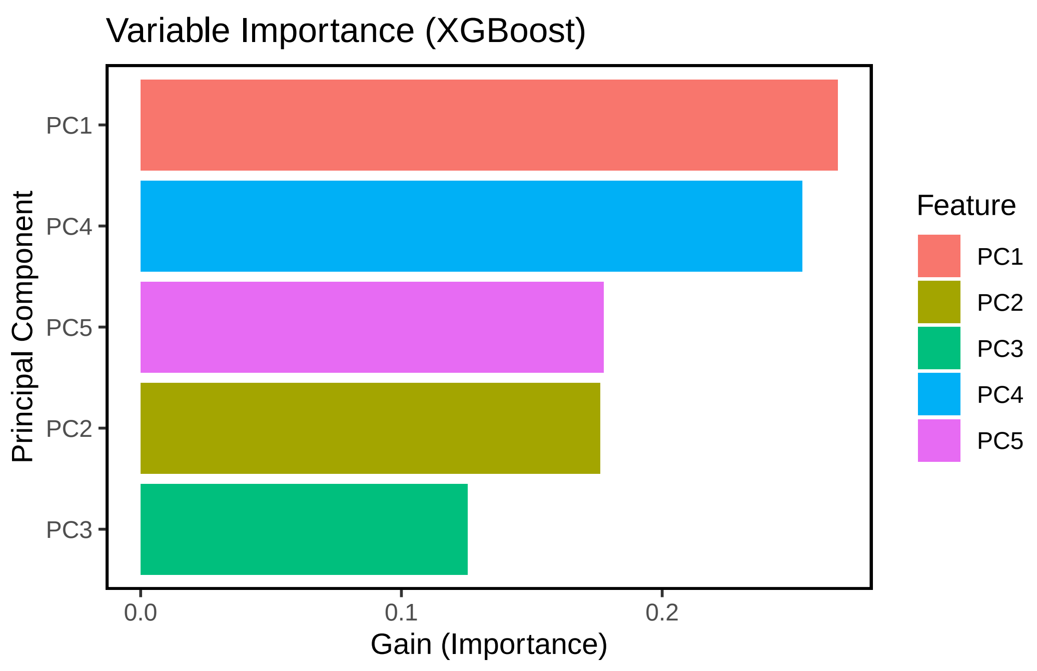


Figure 8: Boosting Regression Model Variable Importance Plot displaying the order of PC importance to the model.