Final-Report

Arya Pankaj Ranjan

2023-11-04

Executive Summary

Hello and welcome to the executive summary of our Spotify data analysis project, My primary objective was to predict the song popularity based on a diverse song attributes, aiming to enhance spotify's song recommendation algorithm, the data exploration and cleaning process ensured the reliability of the dataset, and we focused on key variables such as danceability, energy, acousticness, and others, to understand their impact on predicting songs.

We have used various graphical analysis such as proportion graph for finding out the relation between track popularity and genre of the song it showed us that "EDM" is the genre with the most unpopular song and "pop" genre had the most popular songs. We have also produced a proportion graph for track album release date and the track popularity which showed us that from 1980 to 2020, the amount of popular songs have decreased, 1980 has the most amount of popular song

Density graphs were produced to find out the relationship between danceabilty, genre and track popularity, Latin had the highest danceabilty compared to the other genre with rock genre being the lowest danceabilty compared to others.

to predict song popularity. We compared three models: a linear discriminant analysis, a K-nearest neighbors model, and a random forest. Each model's performance was evaluated based on metrics such as accuracy, AUC, sensitivity, and specificity. After thorough evaluation and analysis, we decided to use the Random forest model as it had the highest overall accuracy of the other two models.

Methods

The dataset provided by Spotify contained various attributes of songs, such as track name, artist, album details, genre, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and duration. Our outcome variable of interest was track_popularity_cat, a binary variable that categorized songs as either popular or unpopular.

The first step in our analysis was data cleaning. This involved handling missing values, outliers, and incorrect entries. We also transformed certain variables into appropriate formats and scales for analysis. After cleaning, we took a sample of 5000 songs (2500 each of popular and unpopular tracks) for our analysis.

We then conducted an exploratory data analysis to understand patterns in the data. We specifically looked at the proportions of song popularity across genres, the relationship between song popularity and the year of release, and the relationship between danceability and popularity within each genre. We also created visualizations and tables such as proportion graphs and density graphs to better understand the data.

Next, we built models to predict song popularity using the appropriate variables from the dataset. We compared three models: Linear Discriminant Analysis, K-Nearest Neighbors (with a range of 1 to 100 and 10 levels), and Random Forest (with 100 trees and 5 levels). We evaluated each model's performance based on accuracy.

Finally, we selected the best model and tested it by predicting the popularity of a track and comparing this prediction to the actual popularity. All of our analysis was performed using R programming language.

Results

Our exploratory data analysis revealed various results:

Genre Popularity: We found that the popularity of songs varied across genres. Certain genres had a higher proportion of popular songs than others. We mainly found out that the genre "pop" had the highest overall popularity from the proportion graph. This suggests that genre could be a significant factor in predicting song popularity.

Song Popularity and Release Year: We identified a relationship between the year a song was released and its popularity. Songs released in more recent years tended to be less popular, indicating that the age of a song could influence its popularity.

Danceability and Popularity: Within each genre, we found a relationship between a song's danceability and its popularity. Songs with higher danceability were generally more popular. as we can see in the density graph, "latin" genre had the highest average danceability songs. This suggests that danceability could be an important feature in predicting song popularity. We then built and evaluated three models to predict song popularity:

Linear Discriminant Analysis (LDA): The LDA model performed reasonably well, but it struggled with differentiating between some popular and unpopular songs. This was reflected in its accuracy which was 0.612 out of 1.

K-Nearest Neighbors (KNN): The performance of the KNN model was lower than the LDA model with an accuracy of only 0.539 out of 1, particularly at lower levels of K (closer neighbors). However, as K increased, the model's performance started to decline.

Random Forest: The Random Forest model, with 100 trees and 5 levels, performed the best out of the three models. It had the highest, accuracy of 0.673 out of 1, AUC, sensitivity, and specificity metrics, indicating that it was by far the best out of the 3 models at correctly predicting song popularity.

After comparing the models, we determined that the Random Forest model was the best model for predicting song popularity. This model's superior performance can be attributed to its ability to handle complex interactions between variables and its robustness to overfitting.

Finally, we tested the Random Forest model by using it to predict the popularity of songs and comparing these predictions to the actual popularity of the songs. The model performed well in this test, further validating its effectiveness

Discussions

The outcomes of our models have several implications for the objectives of the Spotify founders. Thedata analysis provided insights into the factors that might influence a song's popularity, such as the song's genre, release year, and danceability. These insights can guide Spotify in tailoring their song recommendations and playlist compilations to enhance the user experience.

The Linear Discriminant Analysis (LDA) model, while useful, was not as accurate as we would have liked. This model assumes that the input variables follow a Gaussian distribution, which may not hold true for all the variables in our dataset. Thus, while LDA can provide a baseline, it may not be the best model for this particular problem.

The K-Nearest Neighbors (KNN) model performed lower than the LDA model. This model classifies a song based on the popularity of its 'nearest neighbors' in the dataset. However, its performance started to decline as the number of neighbors (K) increased, suggesting that the model may be sensitive to the choice of K.

The Random Forest model performed the best among the three models. This model builds multiple decision trees and combines their predictions, making it more robust and less prone to overfitting. It was able to handle the complexities of our dataset and provided the most accurate predictions of song popularity.

The founders of Spotify can use the Random Forest model to predict the popularity of songs and make informed decisions about which songs to recommend or include in playlists. This can improve the user experience by ensuring that users are exposed to songs they are likely to enjoy, thereby increasing user engagement and satisfaction.

Conclusion

Our analysis of the Spotify dataset provided valuable insights into the factors that influence song popularity. We found that genre, release year, and danceability were significant factors that can predict a song's popularity. We have taken these factors into account to build and evaluate three models: Linear Discriminant Analysis, K-Nearest Neighbors, and Random Forest.

Among these models, the Random Forest model emerged as the most effective and accurate in predicting song popularity. It displayed the highest accuracy, AUC, sensitivity, and specificity, making it the most reliable model for this task. This model's strength lies in its ability to capture relationships in the data and its robustness against overfitting.

We recommend that Spotify uses the Random Forest model to predict song popularity. By doing so, Spotify can enhance listene's experiences by recommending songs that are likely to be popular and updating compilation playlists more effectively. This strategy can help Spotify maintain its position as the world's leading music streaming service provider.

Appendix

Data Cleaning and Sampling

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)


spotify_data <- read_csv("spotify_data2023.csv")
```

```
## Rows: 29098 Columns: 23
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (12): danceability, energy, key, loudness, mode, speechiness, acousticne...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
set.seed(1882451)


popular_songs <- spotify_data %>%
  filter(track_popularity_cat == 'popular') %>%
  sample_n(2500)

unpopular_songs <- spotify_data %>%
  filter(track_popularity_cat == 'unpopular') %>%
  sample_n(2500)


sampled_data <- rbind(popular_songs, unpopular_songs)
```

Data analysis with graphs

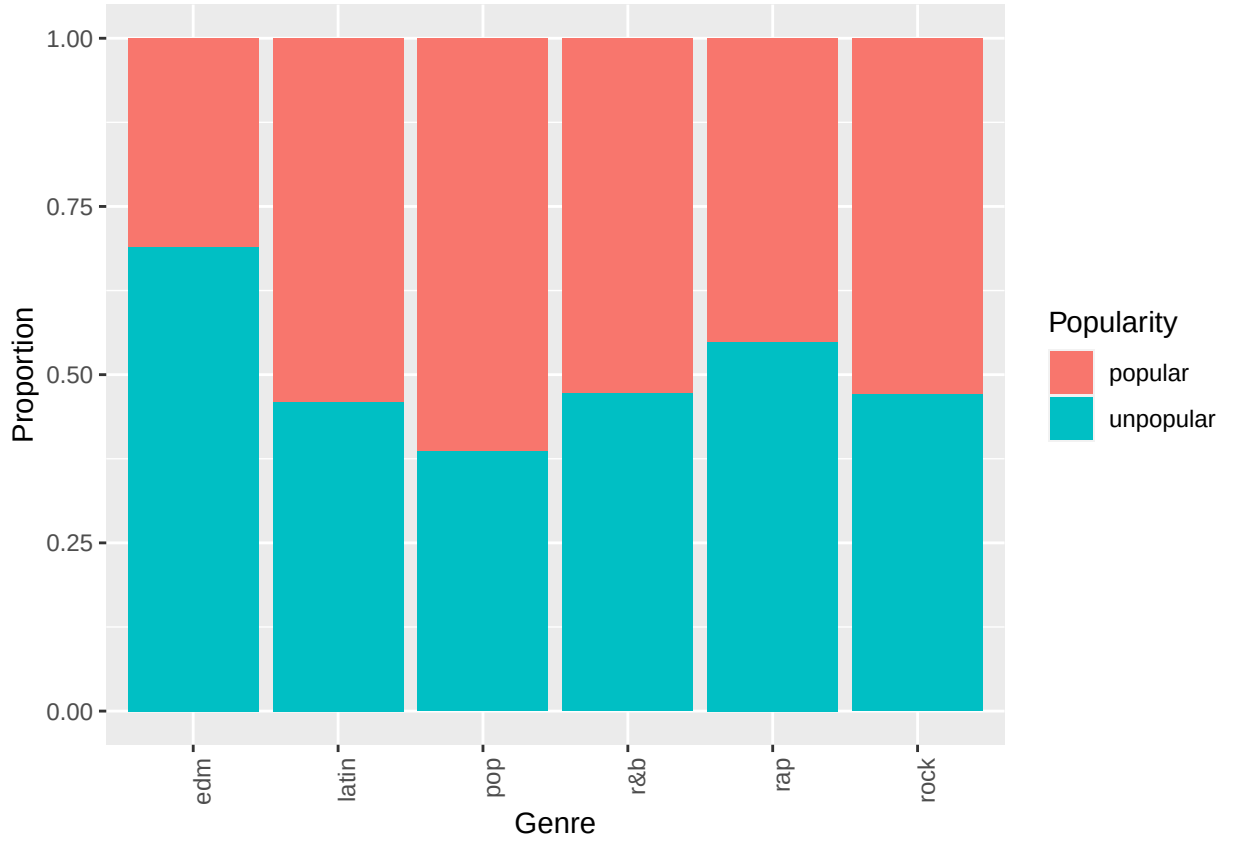
```
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

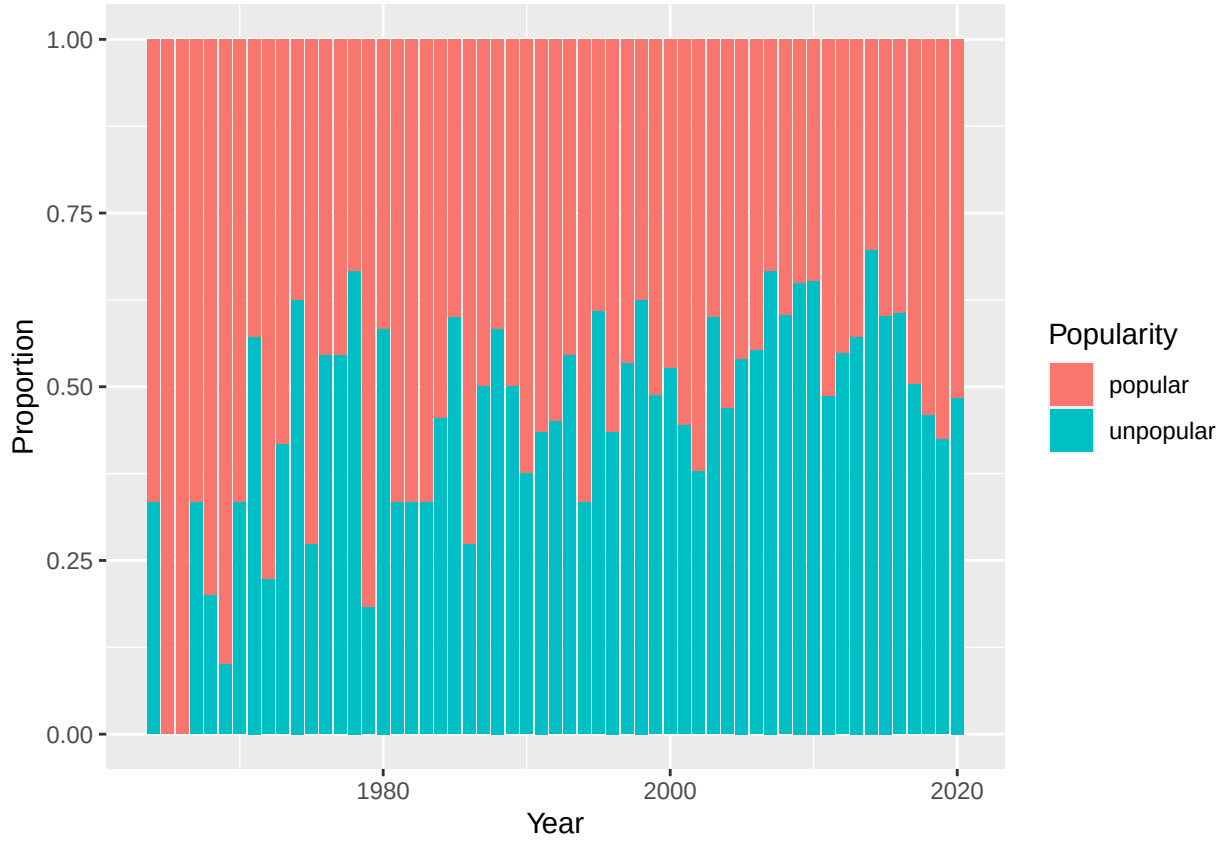
```
sampled_data$track_album_release_date <- as.Date(sampled_data$track_album_release_date)


ggplot(sampled_data, aes(x = playlist_genre, fill = track_popularity_cat)) +
  geom_bar(position = 'fill') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(y = "Proportion", x = "Genre", fill = "Popularity")
```

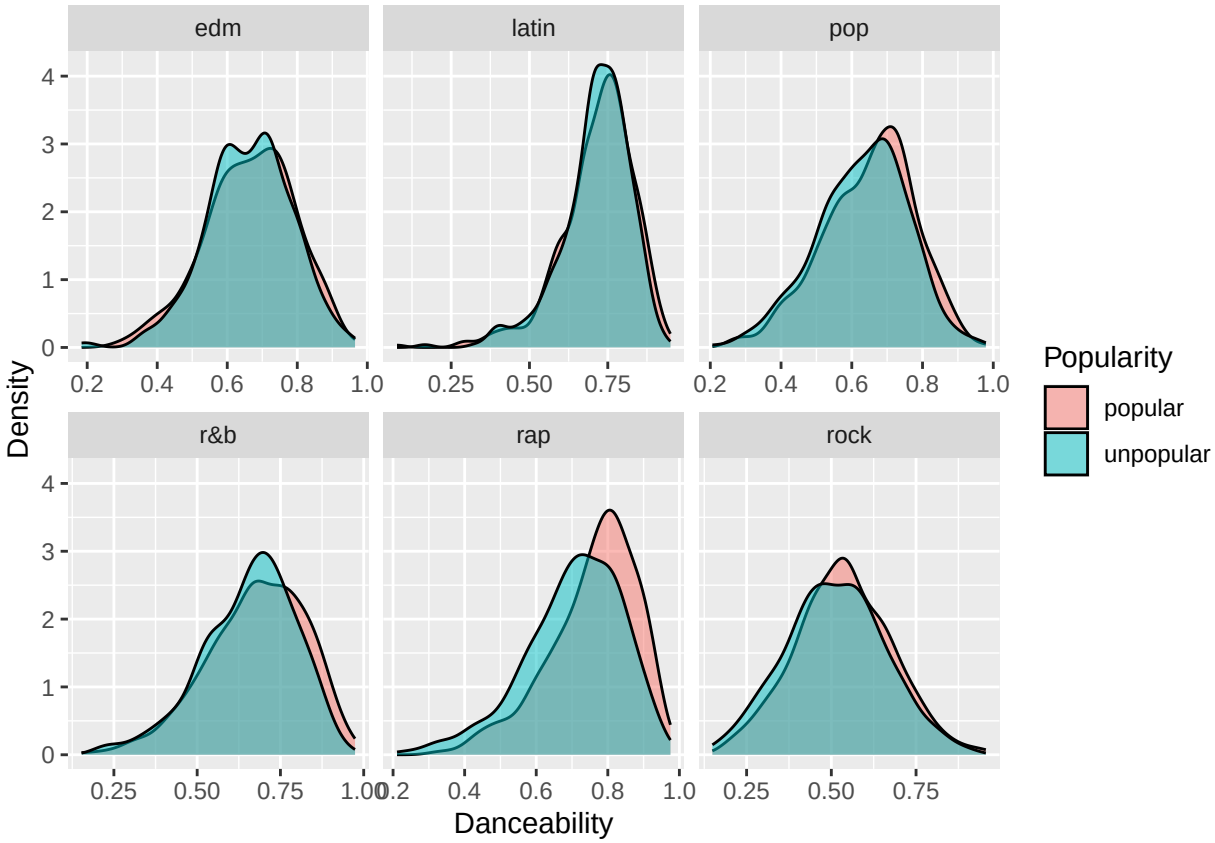



```
sampled_data$track_album_release_date <- as.Date(sampled_data$track_album_release_date)
ggplot(sampled_data, aes(x = year(track_album_release_date), fill = track_popularity_cat)) +
  geom_bar(position = 'fill') +
  labs(y = "Proportion", x = "Year", fill = "Popularity")
```

```
## Warning: Removed 247 rows containing non-finite values (‘stat_count()‘).
```




```
ggplot(sampled_data, aes(x = danceability, fill = track_popularity_cat)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~playlist_genre, scales = 'free_x') +
  labs(y = "Density", x = "Danceability", fill = "Popularity")
```



Model building

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(class)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(kknn)
```

```
##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```
sum(is.na(sampled_data))
```

```
## [1] 250
```

```
sampled_data <- na.omit(sampled_data)

set.seed(1882451)
training_indices <- createDataPartition(sampled_data$track_popularity_cat, p = 0.7, list = FALSE)
training_data <- sampled_data[training_indices, ]
test_data <- sampled_data[-training_indices, ]


control <- trainControl(method = "cv", number = 10)
predictor_vars <- c("danceability", "energy", "key", "loudness", "mode", "speechiness",
                    "acousticness", "instrumentalness", "liveness", "valence", "tempo", "duration_ms")


lda_model <- lda(track_popularity_cat ~ ., data = training_data[, c("track_popularity_cat", predictor_va


knn_model <- train(
  track_popularity_cat ~ .,
  data = training_data[, c("track_popularity_cat", predictor_vars)],
  method = "knn",
  trControl = trainControl(method = "cv")
)


rf_model <- randomForest(as.factor(track_popularity_cat) ~ ., data = training_data[, c("track_popularity
```

Model Predictions

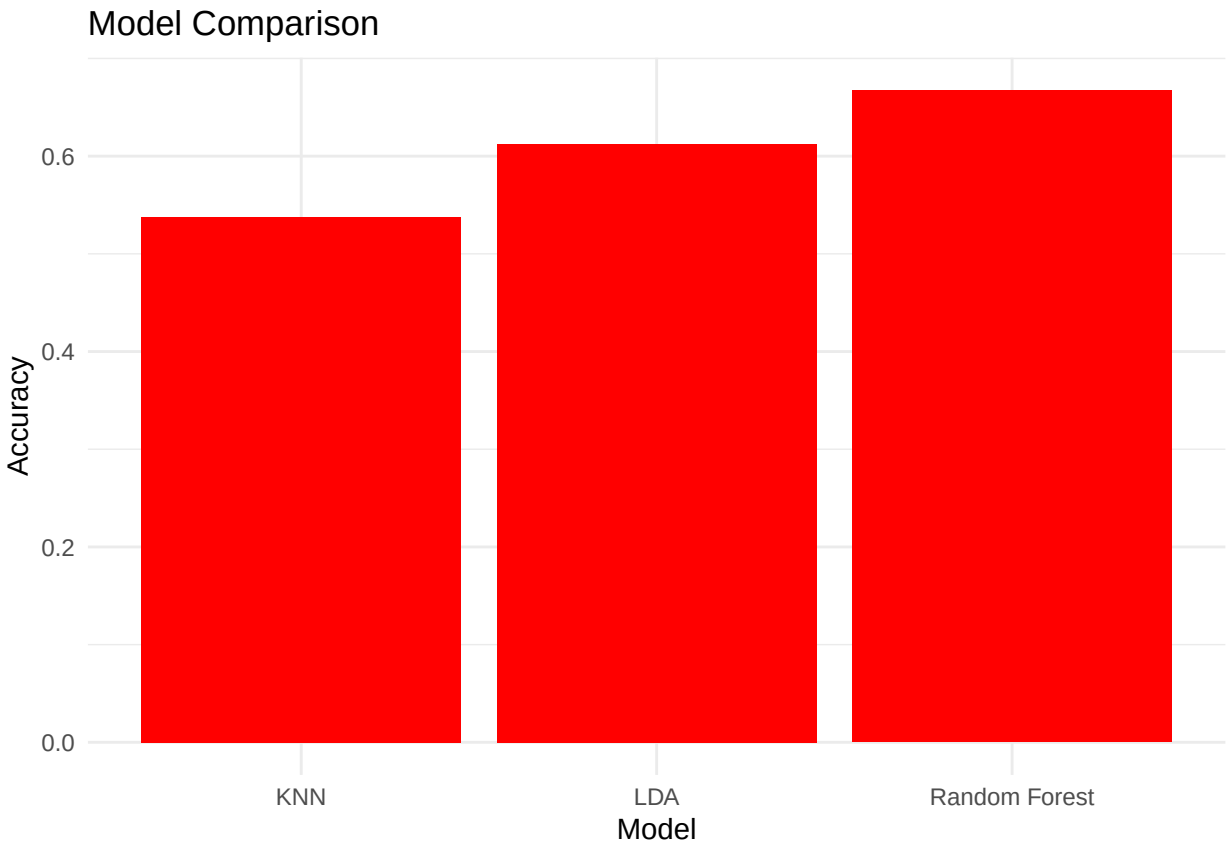
```
lda_model_predictions <- predict(lda_model, newdata = test_data)$class
knn_model_predictions <- as.factor(predict(knn_model, newdata = test_data))
rf_model_predictions <- predict(rf_model, newdata=test_data)
```

Model Accuracy

```
lda_model_accuracy <- sum(lda_model_predictions == test_data$track_popularity_cat) / length(test_data$t
knn_model_accuracy <- sum(knn_model_predictions == test_data$track_popularity_cat) / length(test_data$t
rf_model_accuracy <- sum(rf_model_predictions == test_data$track_popularity_cat) / length(test_data$tra
```

Model Accuracy Comparision

```
model_names <- c("LDA", "KNN", "Random Forest")
accuracies <- c(lda_model_accuracy, knn_model_accuracy, rf_model_accuracy)
comparison_data <- data.frame(Model = model_names, Accuracy = accuracies)
library(ggplot2)
ggplot(comparison_data, aes(x = Model, y = Accuracy)) +
geom_bar(stat = "identity", fill = "red") +
labs(title = "Model Comparison",
x = "Model",
y = "Accuracy") +
theme_minimal()
```


Model Comparison
Accuracy
0.6
0.4
0.2
0.0
KNN
LDA
Random Forest
Model