Genre Prediction Using Spotify Dataset

Ky Phong Mai

2024-07-14

Executive summary

Spotify is a leading audio streaming and media services provider, with a massive user base of over 365 million monthly active users, including 165 million paying subscribers. One of their strategies is to predict the genre of a song to better enhance their music recommendation and advertising algorithms. By accurately predicting a song's genre, Spotify can offer more personalized and relevant recommendations to its users and update its playlists more effectively.

Spotify web services department assisted in providing a dataset for analysis. Spotify founders are interested in how some factors including the release year, speechiness, danceability and tempo can predict a song's genre. Spotify also requires to explore the trending of track popularity over time, the difference of popularity as well as speechiness between genre.

After thorough the analysis, it is shown that popularity differs slightly and speechiness differs significantly between genre. For popularity trending, it is observed that it gradually declined until 2008 before starting to move upward. It was also identified that 12 factors including *track release year, track popularity, danceability, energy, loudness, mode, speechiness, acousticness, instrumentalness, valence, tempo* and *track duration* are needed for predicting genre. Linear discriminant analysis, K-nearest-neighbors and random forest model were compared. Random forest was determined to be the best model for genre prediction as it has the highest accuracy rate.

Methods

An extensive record of 32833 songs collated by Spotify web services department is used in this project. The dataset contains 23 variables. Among these 23 variables, there are many interesting variables that can be used to make prediction about a song's genre, for example:

- speechiness,
- danceability,
- tempo,
- popularity score,
- energy…

Analysis for this project was completed in R version 4.2.2 using tidyverse and dplyr packages.

The first step taken was skimming through the data to have a better understand of the dataset before tidying it up. From first look, there appears to have overlapping track_id in the dataset. While there are 32833 rows, there are only 28356 unique track_ids. The 2 explanations for this are: 1. One song can appear in multiple playlist and 2. They are truly duplicated rows. Even if the song comes from multiple playlist, it should be registered as one song. Hence, it is important to remove all redundant rows.

As one song can come from multiple playlists and each playlist can have different genre, then is it also possible that one song can be classified as multiple genre? If this is the case, it will be very difficult to perform a multi-class classification on the data. For example, if a song belongs to both pop and r&b genre, which genre it should be classified as? After further check, it is shown that the concern is indeed valid. Based on ***Figure 1***, it can be seen that among these songs, there are some that are classified as 2,3,4 or even 5 genres at once. However, these groups only account for a relatively small percentage of data as compared with those that are classified as single genre. As sometimes songs might contain some certain characteristics of more than one genre, which makes it hard to distinguish, even for music professionals. However, to standardize our prediction model, we only consider those that belong to one genre. Therefore, songs with more than one genre will be removed before analysis. Refer to *Appendix* ***Table 4*** for example of the song "bad guy" by "Billie Eilish" appearing in multiple playlists with different genre.

The data is then cleaned and reduced to 1000 songs per genre due to computing power. Exploratory data analysis was performed to have a general idea of the relationship between the genre and other indicating variables. Next, the standard steps were taken to build a prediction model from data splitting and data preprocessing to model tuning and model selection. From the initial predictors, all character variables such as *artist name, playlist_id, track_id…* are removed as they are not indicative of the genre of a song. Variable *key* and *liveness* are also removed as they are not closely related to the genre but more related to musicians who perform the song and the recording decision respectively. Eventually, the following 12 predictors were identified and used in the model: *track release year, track popularity, danceability, energy, loudness, mode, speechiness, acousticness, instrumentalness, valence, tempo* and *track duration.*

The three following models were tuned and compared:

- Linear discriminant analysis
- K-nearest neighbours model with a range of 1 to 100 and 20 levels
- Random forest with 100 trees and 5 levels

The best model is then trained and evaluated with more specific details in the upcoming sections.

Results

Exploratory Data Analysis

General Analysis

One objective of this project is to find out how song factors can predict a song's genre. Those factors include:

- the year the song was released
- how "speechy" the song is
- how danceable the song is
- the tempo of the song

This part of the report will visualise the relationship between genre and the above factors with some initial analysis

1. Year of release

Based on ***Figure 2***, it can be shown that the median year of release for some genres are significantly different from others. For example, with the median year of release at around 2002, this means that half of rock songs came from before the year 2002. At the same time, edm, rap, pop, latin genres have median year of release at around 2018, indicating that half of those songs were only recorded and released from the year 2018 onward. The distribution of songs by year for pop and latin looks fairly similar. From the spread of distribution of release year, distribution of rock tracks is exceptionally spread out as compared to the rest, followed by r&b and then rap. This indicates that rock songs have been evenly released for the longest period of time. On the opposite end, edm songs are released within the shortest amount of time (only a few years).

By comparing the median year of release with the spread, we can give a fairly reasonable comment that rock is the oldest genre while edm is the youngest genre. However, this does not mean that edm were only released recently because in fact, by looking at the outlier points, it can be shown that some edm songs were first recorded since early 1980s.

By looking at the proportion of genre overtime (***Figure 3***), we can once again confirm that before 1990, the music industry was mostly dominated by rock genre. From the year 1990 to 2010, it was the booming period for both r&b and rap. Though decreasing in proportion during these period, rock genre still accounted for a decent percentage of the industry. From 2010 onward, the industry was fairly shared by all 6 genres, with edm holding the highest share and rock holding the lowest share in the music industry.

Based on this preliminary analysis, it seems like year of release of a song is a potential indicator on which genre it belong to. For example, if a song was released before 1990, it is very likely that it belongs to rock genre.

2. Speechiness

As speechiness relates to how speechy (more spoken words) the track is, it can be easily predicted that rap song will be the winner for this category.

Similar to initial impression, ***Figure 4*** shows that rap genre is clearly more speechy than the rest, with the median of around 0.2. At the other end of the spectrum, rock and pop genres are the least speechy, with the lowest median (around 0.05) and smallest spread of level of speechiness. Median of level of speechiness of edm, latin, r&b are more or less similar; however, the spread of their distribution is different, with r&b having the highest spread, following by latin and then edm.

Based the mentioned analysis, it seems that speechiness level of a track is another potential indicator of a track genre. For example, if a song has a very low speechiness level, the probability that it belongs to either rock or pop music is much more higher than the probability of belonging to rap music.

3. Danceability

This self explanatory factor also appears to have some distinction between the genre. With rock being the least danceable and latin and rap being the most danceable, by considering the median value (rock at around 0.52 while latin and pop at around 0.74). Though there are some overlapping among the distribution of all the genre in terms of danceability, rock genre is considered as more significant from the rest.

Based on ***Figure 5***, it seems that danceability has some predictive power to determine the genre a song belongs to. For example, a song with low danceability of 0.4 is more likely to be a rock song than a latin song.

4. Tempo

From ***Figure 6***, it can be seen that edm has the highest median of around 130 beats per minute and r&b has the lowest median of around 110 beats per minute. However in this figure, the median is not the most interesting metric to look at. The spread of tempo is more distinctive with edm having exceptionally small spread of tempo and rap having the biggest spread while other genres having more or less the same spread.

Based on the analysis, it seems that tempo plays some role in predicting the song genre as well. For example, a song with very low tempo of 80 beats per minute will be very less likely to come from the edm genre.

Specific Analysis (requested by Founders)

The section of the report will explore some questions requested by Spotify Founders

1. Does the popularity of songs differ between genre?

From ***Figure 7***, it can be observed that popularity score of edm has the lowest median value of around 30 while that of latin has the highest median value of around 46. From the current dataset, the popularity score of more than 75 is assigned as "high popularity", from 25 to 75 as "medium popularity" and below 25 as "low popularity". ***Figure 8*** shows the number of songs by genre partitioned by the popularity ranking (high, medium, low). By breaking down the popularity level, it can also be seen that edm has the lowest number of highly popular song and highest number of low popular songs. Though latin has the highest median value of popularity score, it is not the genre with the most number of highly popular songs. The winner of having the most number of highly popular songs is no other than pop genre (as the name suggests). Other than that, it does not appear to be distinctive in certain cases, such as for songs with medium to low level of popularity. Based on both Figure, it can be seen that popularity does have some (but not very strong) predictive power over the genre of a song.

Hence, popularity of song differs between genre to some small degree. However, when used along with other indicator, popularity might still be able to contribute to predict genre of a song.

2. Is there a difference in speechiness for each genre?

Based on the ***General Analysis*** section, it is already established that there is difference in speechiness among the genres. However, besides the boxplot of speechiness by genre (***Figure 4***), to better visualise the difference, ***Figure 9*** illustrates the changes in speechiness over the years, which is partitioned by rap and the rest of the genre combined. From here, it is once again confirmed that speechiness of rap is significantly different from the rest. However, another interesting observation is that from 2010 onward, rap genre has became less speechy than before with a sharp drop in the year 2020, reaching around 0.11. This number is just slightly higher than the speechiness of other genre combined (around 0.95). For the remaining genres, it seems that the speechiness trend is going in an upward direction, especially from the year 2000. This indicates that, recently it has become harder to distinguish rap from other genre based on solely speechiness. This indicator also suggests the potential influence of rap to other music types and vice versa.

Generally, there is certainly a difference in speechiness for each genre, especially for rap music. By adding the track release year, it is easier to visualize the difference in speechiness between genres over the years as well.

3. How does track popularity change over time?

Overall, it is observed that average track popularity started out very high at around 49.5 (could be due to limited choice of songs at that time). However, over time, average popularity gradually decreased with fluctuations before hitting rock bottom at around 22 around 2008, which is also the year of recession. From there, the overall trend is recovering upward again, ending at the year 2020 at around 40.3. (***Figure 10***)

Looking at average popularity by genre in ***Figure 11***, trend seems to be also affected by the macro effect of the whole industry, resulting in some genres plummeting in popularity score around 2008, including edm, r&b, rap and rock. Interesting observation to note is that latin does not seem to be affected in the similar way. In fact, this genre reached its lowest point of popularity before the year 2000. Popularity trend of rock songs is mostly flat, indicating that people like this type of songs the same way they did in the past and that it is not affected by the macro effect as much as other genres. Although track popularity alone is not a very good indicator of the genre of a song, track popularity along with the year of release gives additional predictive power to the model

Model evaluation and selection

Three models were examined in order to find the best model for predicting the song genre:

- Linear discriminant analysis (LDA)
- K-nearest neighbours (KNN)
- Random forest

While linear discriminant analysis does not need tuning, the other two models require some tuning first to find the best set of parameters for each model. Refer to Appendix to see the final specifications of the models. Once the final specification of all models are obtained, the performance of the three models are then evaluated based on their accuracy. Accuracy is proportion of number of correct predictions to the total number of predictions. Note that during both tuning stage and evaluation stage, accuracy is the metric used to determine the best model (model with highest accuracy will be selected).

The reason that accuracy is used as the metric for evaluation is because it is suitable for dataset with balanced class. As can be seen from ***Figure 12***, from the clean dataset before reducing to 1000 songs per genre, the percentage of songs for each genre falls within the range from 14.93% (latin) to 18.73% (rap). If the genre is strictly balanced, there should be around 16.67% for each genre. As the variation between the each genre are not much, it is reasonable to consider that the given dataset has equally distributed number of songs per genre. In some other cases, for example, binary classification of COVID testing, accuracy is not a good measurement because the result will be biased toward the majority class. In those cases, sensitivity, specificity or ROC AUC (technical measurement of how well the model is based on different threshold settings) will be crucial. In other situation where the importance of detecting one specific genre is much more than the rest, class sensitivity for that genre should be used instead.

As the objective is to predict as many songs as possible, with all songs holding equal level of importance, accuracy is the most suitable metric to be used.

From ***Figure 13***, it is shown that LDA model has the lowest accuracy at 0.5 while random forest model has the highest accuracy at 0.582. Coming at second place in terms of accuracy, KNN model has an accuracy of 0.524. From here, it seems that random forest model's performance is significantly better than the other 2 models. Hence, random forest model is selected for genre prediction.

Discussion

After random forest is selected as the best model (highest accuracy), the model is trained and tested in an appropriate manner.

Figure 14 shows the importance of each variable in the random forest model in predicting the genre of a track. From top to bottom of the plot, the importance level is reduced. As can be seen from the plot, year of release is the most important variable for genre prediction. Interestingly, during our initial analysis, it is also observed that release year between genre is quite distinctive. The second, third and fourth important variables are danceability, speechiness and tempo respectively. It is worth noting that these 4 variables were all analysed to have differences between genre in our exploratory data analysis. Another interesting observation is that track popularity is not in the top 10 most important predictor. From the initial analysis, popularity was also shown to have some small differences between genre, and the main reason for that is because of how spread out the popularity is. This means that if given the popularity score of a song alone, it is almost impossible to predict which genre it belongs to.

After testing with the preprocessed testing data, the model is able to predict with overall accuracy of 59.3%. The accuracy is slightly than higher when validated with training data set, which is a good thing. This also indicates that our model does not overfit (overfitting refers to when the model only works for training dataset and is not able to generalize to unseen data). By pure random chance, the accuracy for predict a song genre is only 16.7%. Hence, the model's accuracy of 59.3 is considered as a great improvement.

Other than accuracy, sensitivity and specificity per genre also provide some interesting understanding of the prediction model. ***Figure 15*** shows that sensitivity of detecting pop is the lowest at 0.44, and the sensitivity of detecting rock is highest at 0.82. Similarly, the lowest specificity at 0.87 belongs to pop genre and the highest specificity at 0.95 belongs to rock genre. However, it is crucial to understand the implication of sensitivity and specificity in this context. For example, edm has the sensitivity of 0.68, which means that if a song is truly an edm song, the model will correctly predict it as edm approximately 68% of the time. Similarly, the same interpretation of sensitivity can be done for other genres. The meaning of specificity is opposite. For example, when rap has the specificity of 0.92, it means that if a song is not a rap song, the model will correctly predict it as a non-rap song approximately 92% of the time. However, predicting correctly a song as non-rap song does not necessarily mean that a song is correctly classified as one of the other 5 genres. In general, it is observed that specificity is much higher than sensitivity for all genre. Specificity level is quite consistent across all genre while sensitivity varies across all genre.

From the observations, it seems that rock music is the easiest to get classified correctly, followed by edm and rap. At the same time, it seems harder to correctly detect latin, r&b and pop, with pop being the hardest to differentiate. This aligns with some of our exploratory data analysis, with rock, edm and rap having many variables that can be used to separate it from the rest. To be more specific, based on just the top 3 important variables, it can be shown that rock is rather distinctive in all of them: 1. rock accounts for highest percentage for years before 1990 and lowest percentage for years after 2010; 2. rock is the least speechy genre; 3. rock is the least danceable genre. This clearly explains why the model could detect rock music very well.

Refer to the Appendix for confusion matrix (***Figure 16***) of the test data along with other overall statistics of prediction results

Conclusion

With over 365 million monthly active users, including 165 million paying subscribers, Spotify is a major provider of audio streaming and media services. Spotify wishes to enhance user experience by recommending personalized songs to its users and updating playlists effectively. In order to achieve this objective, Spotify aims to generate a model that is able to predict and identify which genre a song belongs to.

With the extensive dataset provided by web services department, Spotify wants to understand how factors such as year of release, speechiness, danceability and tempo can be used to identify genre of a song. Upon requested by Spotify founders, the relationship between genre and factors such as track popularity and speechiness is analyzed along with the popularity trending over the years

After careful analysis, it was shown that speechiness differs significant between genre while track popularity differs to a smaller degree. Throughout the years, average track popularity experienced downward trending with fluctuations from 1970 to around 2010 before increasing at a faster rate. The sharp drop of popularity around 2008 coincides with the economy recession period, and it appears to be an industry-wide effect to the many genres with an exception of latin music.

Three models including linear discriminant analysis, k-nearest-neighbours and random forest were compared in terms of accuracy: random forest has the highest accuracy of 0.582; K-nearest neighbours and LDA have accuracy rate of 0.524 and 0.5 respectively. Hence, random forest is chosen as the best model.

Using the random forest model for testing, the actual accuracy of the model is 59.3%, which is significantly better than 16.7% (accuracy when predicting genre randomly). Noticeably, the reported accuracy is also higher than the accuracy measured at training and validation. It was also illustrated that the year a song was released are the most important factor in predicting the genre. Speechiness, danceability and tempo are the second, third and fourth important factors for genre classification. Furthermore, the model is able to detect rock song easily while their ability to identify pop genre is weakest among all genre. The ability for classifying rock song can be explained by the distinction of rock songs compared with other genres by the top 3 important factors.

Though the accuracy is higher than random prediction, likelihood of correctly classifying each individual genre is fairly different. This could be the actual underlying differences between the genre. Some genre might be more distinctive than other, and some might be harder to distinguish. A limitation of the model is that it can only predict songs that belong to a single genre. However in reality, a song can actually belong multiple genres (pop and r&b) at the same time, especially with the evolution of the music industry in recent years. In the near future, more genres will be mixed together to create a new genre and the distinctive characteristic of one genre will be less and less noticeable. Hence, the genre classification issue will be even more challenging.

Future work should consider the mentioned fact to employ a multi-label classification model instead of a simple multiclass classification model like the one examined in this report. The results presented in this report could be used as preliminary analysis and thresholds for future work.

Appendix

(Loading library and data)

```r
pacman::p_load(tidyverse, tidymodels,readr, skimr,discrim, ggthemes, ggsci, vip, caret, knitr)
```

```r
spotify_songs_raw <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/mas
spotify_songs <- spotify_songs_raw
skim_without_charts(spotify_songs)
```

Table 1: Data summary

Name	spotify_songs
Number of rows	32833
Number of columns	23
Column type frequency:	
character	10
numeric	13
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
track_id	0	1	22	22	0	28356	0
track_name	5	1	1	144	0	23449	0
track_artist	5	1	2	69	0	10692	0
track_album_id	0	1	22	22	0	22545	0
track_album_name	5	1	1	151	0	19743	0
track_album_release_date	0	1	4	10	0	4530	0
playlist_name	0	1	6	120	0	449	0
playlist_id	0	1	22	22	0	471	0
playlist_genre	0	1	3	5	0	6	0
playlist_subgenre	0	1	4	25	0	24	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
track_popularity	0	1	42.48	24.98	0.00	24.00	45.00	62.00	100.00
danceability	0	1	0.65	0.15	0.00	0.56	0.67	0.76	0.98
energy	0	1	0.70	0.18	0.00	0.58	0.72	0.84	1.00
key	0	1	5.37	3.61	0.00	2.00	6.00	9.00	11.00
loudness	0	1	-6.72	2.99	-46.45	-8.17	-6.17	-4.64	1.27
mode	0	1	0.57	0.50	0.00	0.00	1.00	1.00	1.00
speechiness	0	1	0.11	0.10	0.00	0.04	0.06	0.13	0.92
acousticness	0	1	0.18	0.22	0.00	0.02	0.08	0.26	0.99
instrumentalness	0	1	0.08	0.22	0.00	0.00	0.00	0.00	0.99

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
liveness	0	1	0.19	0.15	0.00	0.09	0.13	0.25	1.00
valence	0	1	0.51	0.23	0.00	0.33	0.51	0.69	0.99
tempo	0	1	120.88	26.90	0.00	99.96	121.98	133.92	239.44
duration_ms	0	1	225799.81	59834.01	4000.00	187819.00	216000.00	253585.00	517810.00

(Data Cleaning)

```
spotify_songs %>%
  distinct(track_id,playlist_genre)%>%
  count(track_id, playlist_genre) %>%
  pivot_wider(names_from = playlist_genre, values_from = n, values_fill = 0)%>%
  mutate(total_genre = rock + pop +`r&b`+ latin + edm+rap)%>%
  ggplot(aes(x=total_genre, fill = factor(total_genre)))+
  geom_bar(color = 'black')+
  geom_text(aes(label = ..count..), stat = "count", vjust = -0.5)+
  labs(x = "Number of genres per song",
       y = "Number of songs")+
  theme_clean()+
  scale_fill_npg(name = "Number of Genres")
```

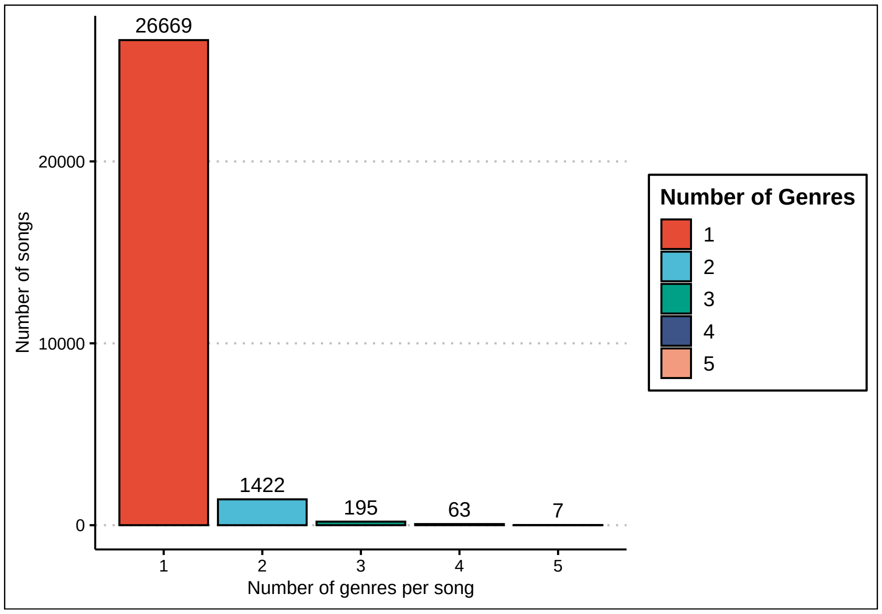

Figure 1: Number of songs categorised by number of distinct genres per song

```
spotify_songs %>%
  distinct(track_id,playlist_genre)%>%
  count(track_id, playlist_genre) %>%
  pivot_wider(names_from = playlist_genre, values_from = n, values_fill = 0)%>%
  mutate(total_genre = rock + pop +`r&b`+ latin + edm+rap) %>%
  filter(total_genre ==5) %>%
  inner_join(spotify_songs,by="track_id")%>%
  dplyr::select(track_name, track_artist, playlist_name, playlist_genre,playlist_subgenre) %>%
  head() %>%
  kable (caption = "Example of one track with multiple genre")
```

Table 4: Example of one track with multiple genre

track_name	track_artist	playlist_name	playlist_genre	playlist_subgenre
bad guy	Billie Eilish	Pop Warmup 130 BPM	pop	dance pop
bad guy	Billie Eilish	Pop - Pop UK - 2019 - Canadian Pop - 2019 - Pop	pop	post-teen pop
bad guy	Billie Eilish	Music&Other Drugs	pop	indie poptimism
bad guy	Billie Eilish	Permanent Wave	rock	permanent wave
bad guy	Billie Eilish	Global Top 50 | 2020 Hits	latin	latin hip hop
bad guy	Billie Eilish	urban CONTEMPORARY	r&b	urban contemporary

```
# Select songs that is only classified as a single genre
spotify_songs_single_genre <-spotify_songs %>%
  distinct(track_id,playlist_genre)%>%
  count(track_id, playlist_genre) %>%
  pivot_wider(names_from = playlist_genre, values_from = n, values_fill = 0)%>%
  mutate(total_genre = rock + pop +`r&b`+ latin + edm+rap) %>%
  filter(total_genre ==1)%>%
  dplyr::select(track_id)

# Clean the data
spotify_songs_cleaned <- spotify_songs %>%
  distinct(track_id, track_popularity,track_album_release_date,playlist_genre,
           danceability,energy,key,loudness,mode,speechiness,acousticness,
           instrumentalness,liveness,valence,tempo,duration_ms)%>%
  mutate (release_year =as.integer(str_sub(track_album_release_date,1,4)),
          playlist_genre = factor(playlist_genre),
          key = factor(key),
          mode = factor(mode))%>%
  dplyr::select(-track_album_release_date)

# Combine to final data
spotify_songs_cleaned <- spotify_songs_cleaned %>%
  inner_join(spotify_songs_single_genre)
```

```
## Joining with ‘by = join_by(track_id)‘
```

```
# Reduce to 1000 samples per genre
set.seed(1874837)
spotify_songs <- spotify_songs_cleaned %>%
  group_by(playlist_genre) %>%
  slice_sample(n = 1000)%>%
  ungroup()
```

(Exploratory Data Analysis)

Year the song was released

```r
spotify_songs %>%
  ggplot(aes(x = fct_reorder(playlist_genre, release_year), y = release_year, fill = playlist_genre)) +
  geom_boxplot(alpha = 0.85)+
  labs (x = "Genre",
        y = "Release Year")+
  theme_clean()+
  scale_fill_npg(name = "Genre")
```

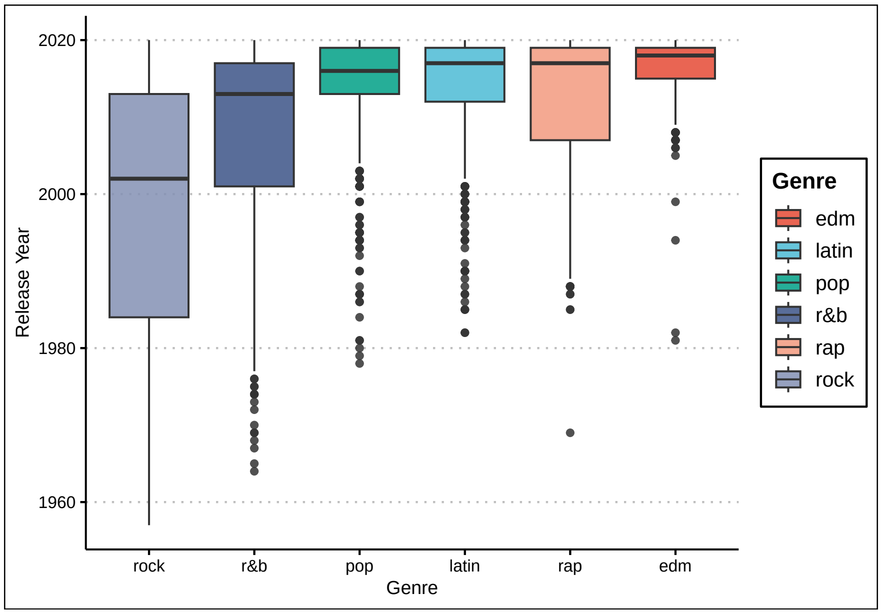

Figure 2: Boxplot of track release year by Genre

```r
spotify_songs %>%
  filter(release_year > 1970)%>%
  ggplot(aes (x = release_year, fill = playlist_genre)) +
  geom_bar(position = 'fill',col = "black", alpha = 0.85)+
  labs (x = "Release Year",
        y = "Proportion of songs")+
  theme_clean() +
  scale_fill_npg(name = "Genre")
```

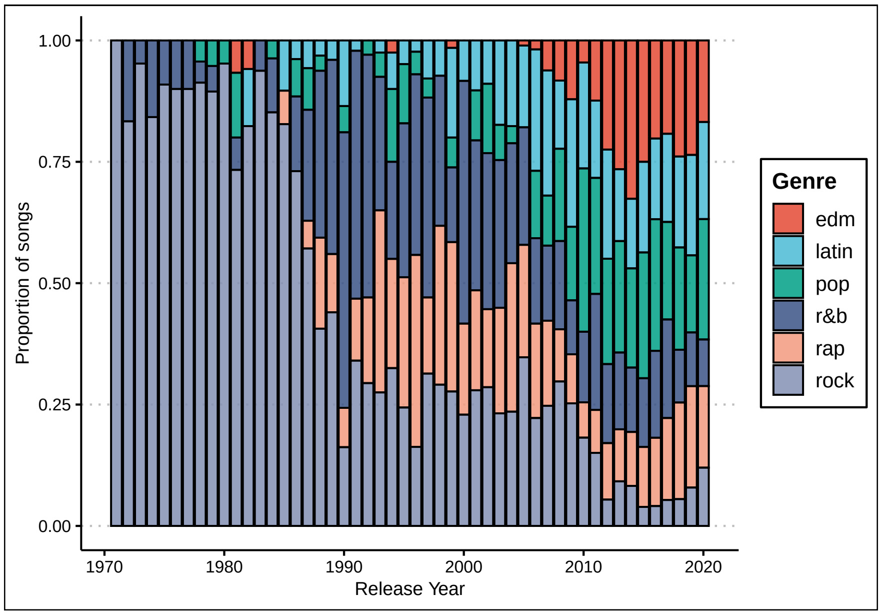

Figure 3: Proportion of Genre from the year 1970 to 2020

Speechiness

```
spotify_songs %>%
  ggplot(aes(x = fct_reorder(playlist_genre,speechiness), y = speechiness, fill = playlist_genre))+
  geom_boxplot(alpha = 0.85)+
  labs(x = "Genre",
       y = "Speechiness")+
  theme_clean() +
  scale_fill_npg(name = "Genre")
```

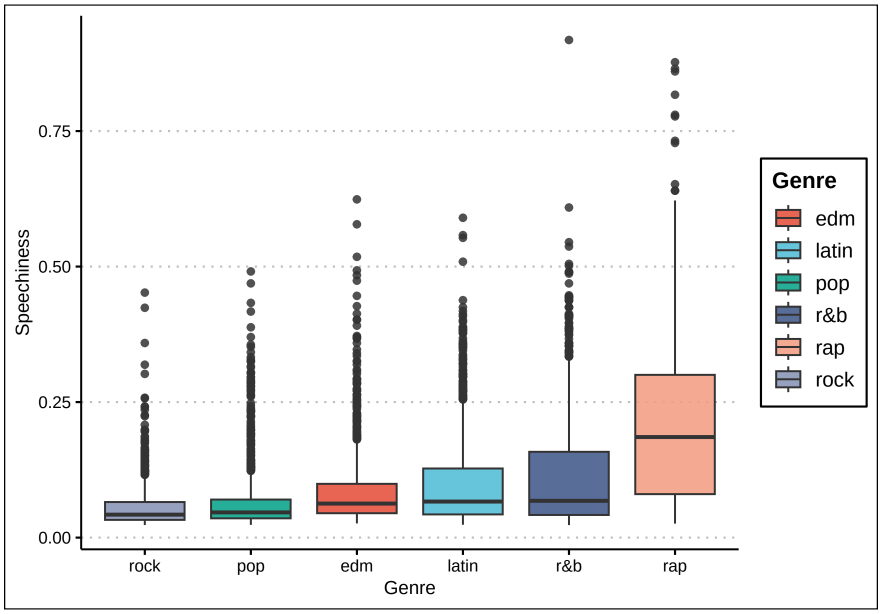

Figure 4: Boxplot of speechiness by Genre

Danceability

```
spotify_songs %>%
  ggplot(aes(x = fct_reorder(playlist_genre,danceability), y = danceability, fill = playlist_genre))+
  geom_boxplot(alpha = 0.85)+
  labs(x = "Genre",
       y = "Danceability")+
  theme_clean() +
  scale_fill_npg(name = "Genre")
```

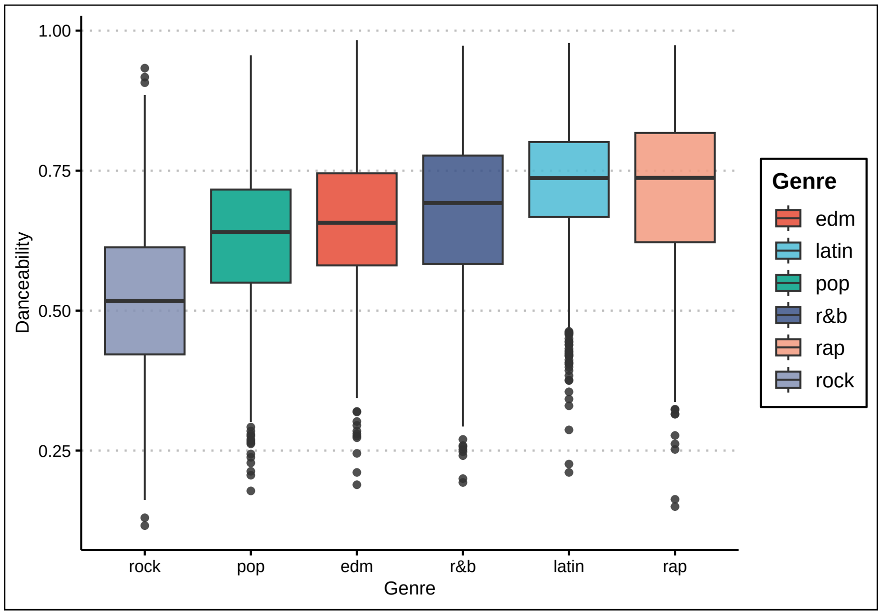

Figure 5: Boxplot of danceability by Genre

Tempo

```
spotify_songs %>%
  ggplot(aes(x = fct_reorder(playlist_genre,tempo), y = tempo, fill = playlist_genre))+
  geom_boxplot(alpha = 0.85)+
  labs (x = "Genre",
        y = "Tempo (beats per minute)")+
  theme_clean() +
  scale_fill_npg(name = "Genre")
```

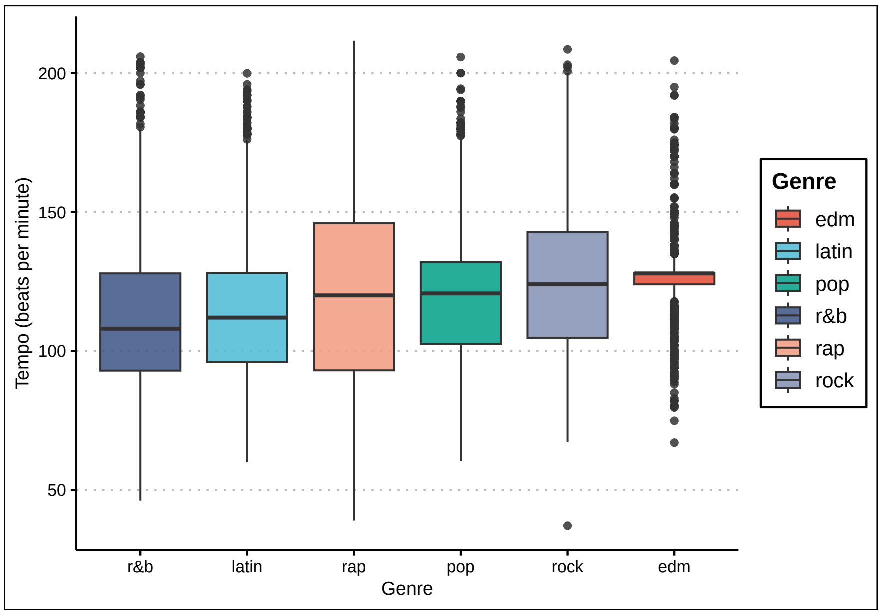

Figure 6: Boxplot of tempo by Genre

(Specific Analysis)

Does the popularity of songs differ between genres?

```
spotify_songs %>%
  ggplot(aes(x = fct_reorder(playlist_genre,track_popularity), y = track_popularity, fill = playlist_ger
  labs(x = "Genre",
       y = "Popularity score")+
  theme_clean()+
  scale_fill_npg(name = "Genre")
```

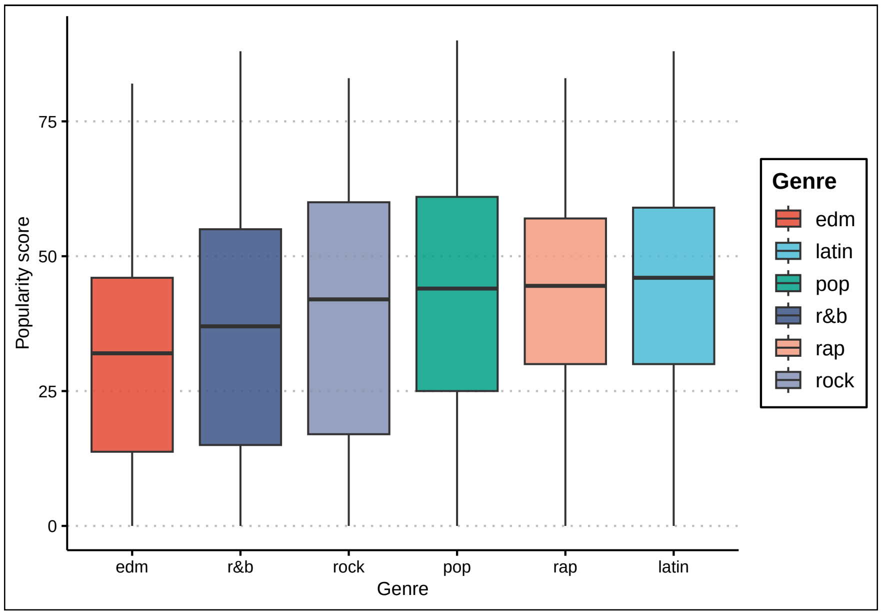

Figure 7: Boxplot of track popularity by Genre

```r
spotify_songs %>%
  mutate (popularity_rank = factor(case_when(track_popularity >75 ~ 'high: > 75',
                                      track_popularity %in% 25:75 ~ 'medium: 25-75',
                                      track_popularity < 25 ~ 'low: < 25'))) %>%
  ggplot(aes(x= playlist_genre, fill = popularity_rank)) + geom_bar(color = "black", position = "dodge"
  labs (x = "Genre",
        y = "Number of songs")+
  theme_clean()+
  scale_fill_npg(name = "Popularity")
```

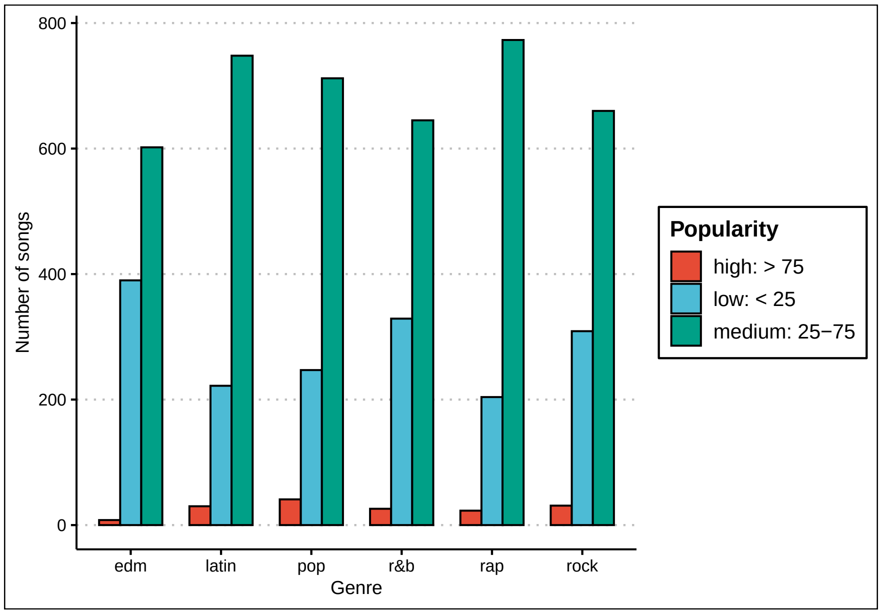

Figure 8: Number of songs by Genre and popularity ranking

Is there difference in speechiness for each genre

```r
spotify_songs %>%
  mutate(year =release_year,
         playlist_genre = ifelse(playlist_genre=='rap', 'rap','other')) %>%
  filter(year >=1970)%>%
  group_by(year,playlist_genre) %>%
  summarise(mean = mean(speechiness))%>%
  ungroup() %>%
  ggplot(aes(x =year, y = mean, color = playlist_genre))+
  geom_line(size = 0.8, alpha = 0.3)+
  geom_smooth(size = 1.5,se=FALSE)+
  labs(x = "Release Year",
       y = "Speechiness")+
  theme_clean()+
  scale_color_npg(name = "Genre")
```

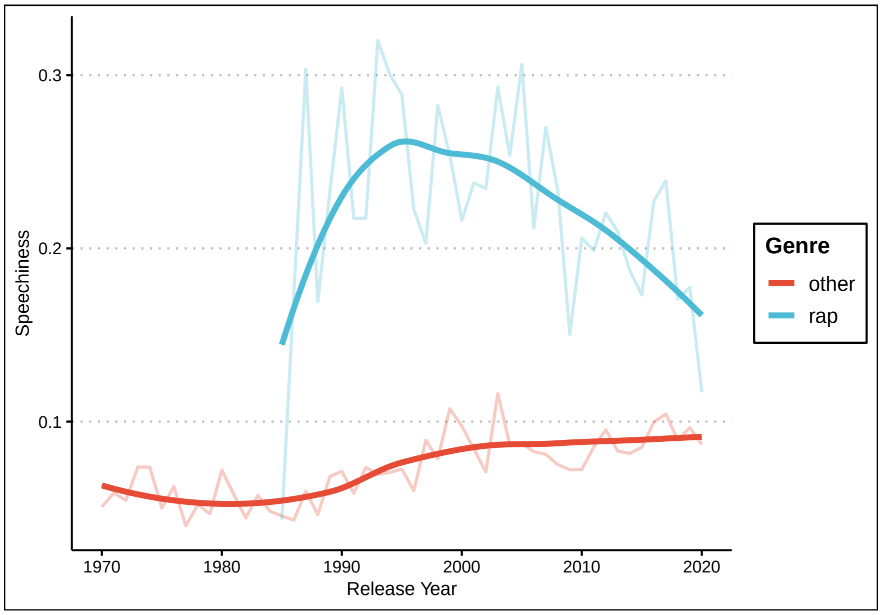

Figure 9: Speechiness trending from 1970 to 2020 partitioned by rap vs the rest

How does track popularity change over time?

```r
spotify_songs %>%
  mutate(year =release_year) %>%
  filter(year >=1970)%>%
  group_by(year) %>%
  summarise(mean = mean(track_popularity))%>%
  ungroup() %>%
  ggplot(aes(x =year, y = mean))+
  geom_line(size = 0.8, alpha = 0.3, color = pal_npg()(9)[6])+
  geom_smooth(size = 1.5,se=FALSE, color = pal_npg()(9)[6])+
  geom_label(color = pal_npg()(9)[9],alpha = 0.9,size = 3,aes(label = ifelse(year %% 10==0,round(mean,1)
  labs(x = "Release Year",
       y = "Average Popularity")+
  theme_clean()+
  scale_color_npg()
```

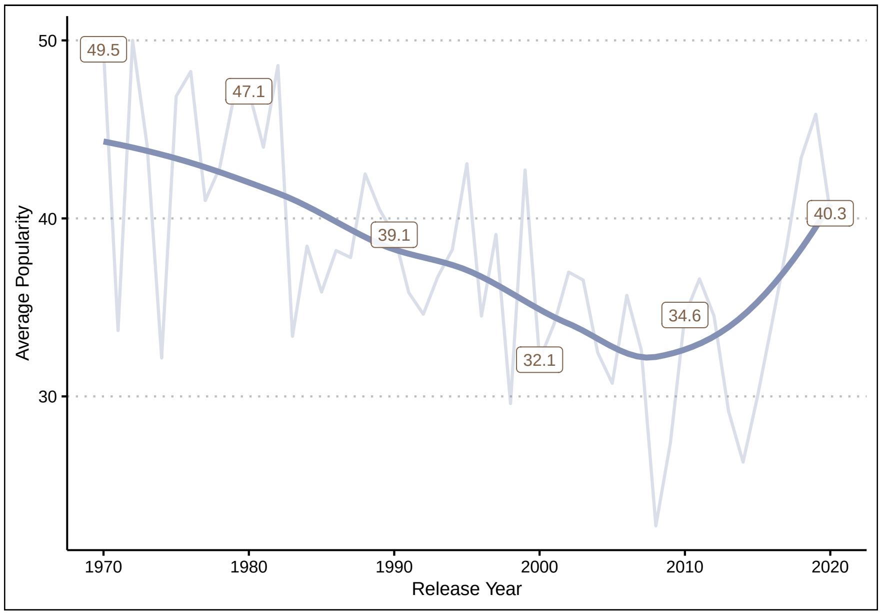

Figure 10: Average track popularity trending from 1970 to 2020

```
spotify_songs %>%
  mutate(year = release_year) %>%
  filter(year >=1970) %>%
  group_by(year, playlist_genre) %>%
  summarise(mean = mean(track_popularity))%>%
  ungroup() %>%
  ggplot(aes(x =year, y = mean, color = playlist_genre))+
  geom_line(size = 0.5, alpha = 0.2, color = "black")+
  geom_smooth(se=FALSE)+
  labs(x = "Release Year",
       y = "Average Popularity")+
  facet_wrap(~playlist_genre)+
  theme_clean()+
  theme(panel.spacing = unit(1.5, "lines"),
        axis.text.x = element_text(size = 6))+
  scale_color_npg(name = "Genre")
```

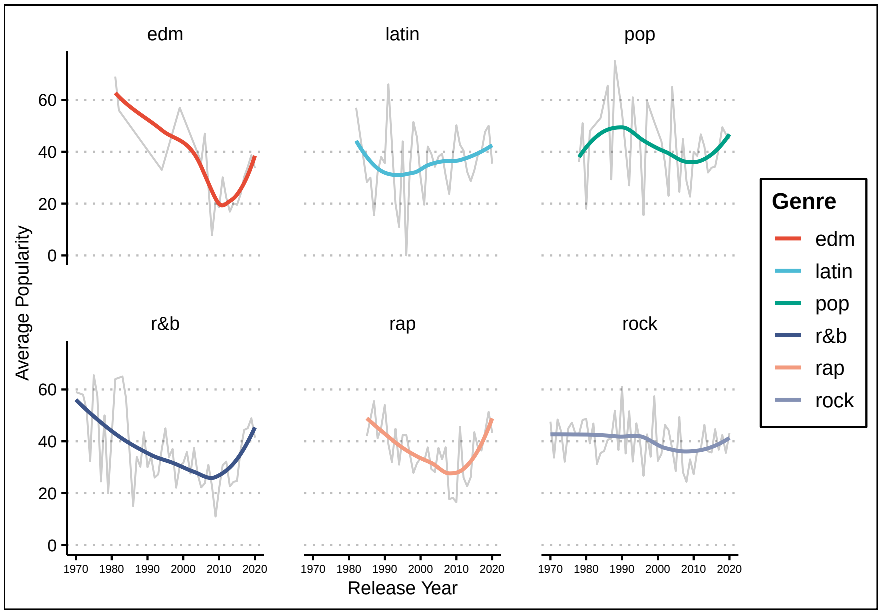

Figure 11: Average track popularity trending from 1970 to 2020 by Genre

(Data Splitting)

```
set.seed(1874837)
spotify_split <- initial_split(spotify_songs, strata = playlist_genre)
spotify_train <- training (spotify_split)
spotify_test <- testing (spotify_split)
```

(Data Preprocessing)

```
spotify_recipe <- recipe (playlist_genre ~., data = spotify_train) %>%
  step_rm(all_string_predictors(), key,liveness) %>%
  step_zv(all_predictors()) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_normalize(all_predictors()) %>%
  step_corr (all_predictors()) %>%
  prep()

spotify_recipe
```

```
##

## -- Recipe ----------------------------------------------------------------------

##

## -- Inputs
## Number of variables by role

## outcome:    1
## predictor: 15

##

## -- Training information

## Training data contained 4500 data points and no incomplete rows.

##

## -- Operations

## * Variables removed: track_id, key, liveness | Trained

## * Zero variance filter removed: <none> | Trained

## * Dummy variables from: mode | Trained

## * Centering and scaling for: track_popularity, danceability, ... | Trained

## * Correlation filter on: <none> | Trained
```

```r
spotify_train_preproc <- juice(spotify_recipe)
```

(Model Specifications)

```r
#LDA spec
lda_spec <- discrim_linear(mode ="classification") %>%
  set_engine("MASS")

#K-nearest neighbors
knn_spec <- nearest_neighbor(mode = "classification",
                             neighbors = tune()) %>%
  set_engine("kknn")

#Random Forest
rf_spec <- rand_forest(mode = "classification",
                       trees = 100,
                       mtry = tune(),
                       min_n = tune()) %>%
  set_engine("ranger", importance ="permutation")
```

(Model Tuning)

```r
# Create 10 bootstraps resamples
set.seed(1874837)
spotify_boots <- bootstraps(spotify_train_preproc, times = 10, strata = playlist_genre)
```

```r
#For k-nearest neighbors
k_grid <- grid_regular(neighbors (range = c(1,100)),
                       levels = 20)

doParallel::registerDoParallel()
knn_tune <- tune_grid(object = knn_spec,
                      preprocessor = recipe(playlist_genre ~ ., data = spotify_train_preproc),
                      resamples = spotify_boots,
                      grid = k_grid)

best_knn_acc <- select_best(knn_tune, "accuracy")

knn_spec_final <- finalize_model(knn_spec,best_knn_acc)

#For random forest
rand_grid <- grid_regular(finalize (mtry(),
                                    spotify_train_preproc %>% dplyr::select(-playlist_genre)),
                          min_n(),
                          levels = 5)
doParallel::registerDoParallel()
set.seed(1874837)
rand_tune <- tune_grid (object = rf_spec,
                        preprocessor = recipe(playlist_genre~., data = spotify_train_preproc),
                        resamples = spotify_boots,
                        grid = rand_grid)

best_rand_acc <- select_best(rand_tune, "accuracy")

rf_spec_final <- finalize_model(rf_spec, best_rand_acc)
```

```r
# Final specs of all models
lda_spec
```

```
## Linear Discriminant Model Specification (classification)
##
## Computational engine: MASS
```

```r
knn_spec_final
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 100
##
## Computational engine: kknn
```

```r
rf_spec_final
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 3
##   trees = 100
##   min_n = 2
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

(Model Selection)

```r
# Cross validation
set.seed(1874837)
spotify_cv <-vfold_cv(spotify_train_preproc, v =10, strata = playlist_genre)

# Fit
lda_cv <- fit_resamples(object = lda_spec,
                        preprocessor = recipe (playlist_genre~., data = spotify_train_preproc),
                        resamples = spotify_cv)

knn_cv <- fit_resamples (object = knn_spec_final,
                         preprocessor = recipe (playlist_genre~.,data = spotify_train_preproc),
                         resamples = spotify_cv)

set.seed(1874837)
rf_cv <- fit_resamples (object = rf_spec_final,
                        preprocessor = recipe (playlist_genre~., data = spotify_train_preproc),
                        resamples = spotify_cv)
```

```r
spotify_songs_cleaned %>%
  count(playlist_genre) %>%
  mutate (prop = round(n/sum(n)*100,2))%>%
  ggplot(aes(x = playlist_genre, y = prop, fill = playlist_genre))+
  geom_col(col = "black",alpha = 0.85)+
  geom_hline(yintercept = 16.67, linetype = "longdash", size = 1, alpha = 0.3, color = "purple")+
  geom_text(aes(label = prop), vjust = -0.3, size = 4)+
  labs(x="Genre",
       y = "Percentage (%)")+
  theme_clean()+
  scale_fill_npg(name = "Genre")
```

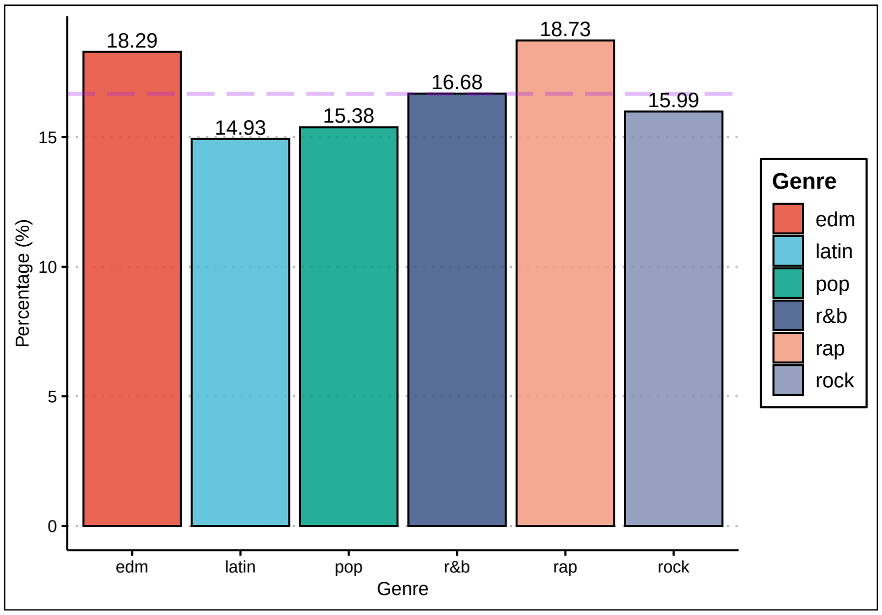

Figure 12: Percentage of songs by genre

```
# Comparing model
lda_cv %>% collect_metrics() %>% bind_cols (model = factor("lda")) %>%
  bind_rows(knn_cv %>% collect_metrics() %>% bind_cols (model = factor("knn")))%>%
  bind_rows(rf_cv %>% collect_metrics() %>% bind_cols (model = factor("ranndom forest")))%>%
  dplyr::select(-.config, -n, -.estimator)%>%
  filter (.metric =="accuracy")
```

```
## # A tibble: 3 x 4
##   .metric   mean std_err model
##   <chr>    <dbl>   <dbl> <fct>
## 1 accuracy 0.501 0.00995 lda
## 2 accuracy 0.524 0.00771 knn
## 3 accuracy 0.582 0.00669 ranndom forest
```

```
lda_cv %>% collect_metrics() %>% bind_cols (model = factor("lda")) %>%
  bind_rows(knn_cv %>% collect_metrics() %>% bind_cols (model = factor("knn")))%>%
  bind_rows(rf_cv %>% collect_metrics() %>% bind_cols (model = factor("ranndom forest")))%>%
  dplyr::select(-.config, -n, -.estimator)%>%
  filter (.metric =="accuracy") %>%
  ggplot(aes(x = model, y = mean, fill = model))+
  geom_col(width = 0.5, col = "black")+
  geom_errorbar(aes(ymin = mean-std_err, ymax = mean+std_err), width = 0.1)+
  coord_cartesian(ylim = c(0.4,0.6))+
  labs(x="Model",
       y = "Accuracy")+
  theme_clean()+
  scale_fill_npg()
```

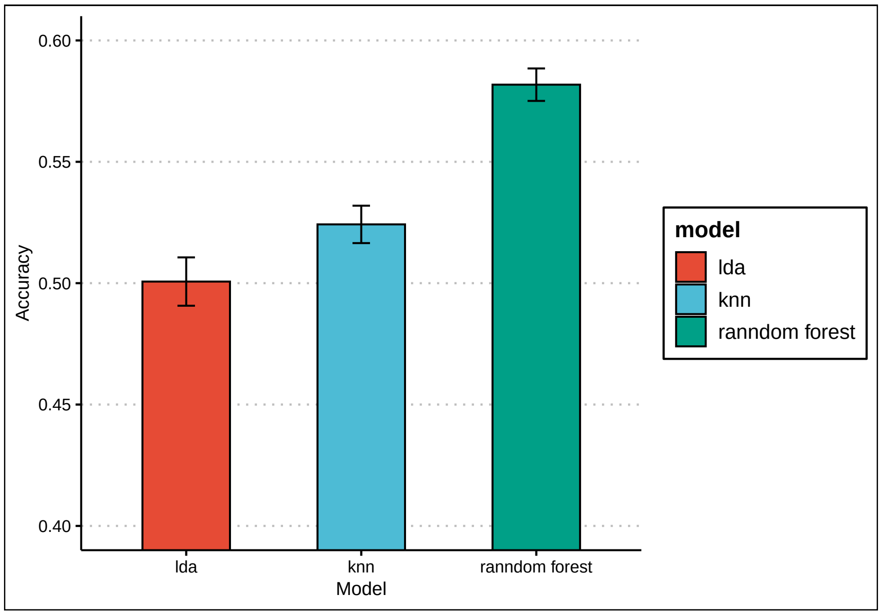

Figure 13: Accuracy metric for all 3 models

(Model Evaluation)

```r
# Preprocess test data
spotify_test_preproc <- bake(spotify_recipe, spotify_test)

# Fit random forest model to training data
spotify_rf <- rf_spec_final %>%
  fit(playlist_genre ~., data = spotify_train_preproc)
```

```r
# Variable importance plot
spotify_rf
```

```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~3L,      x), num.trees = ~100, min.
##
## Type:                             Probability estimation
## Number of trees:                  100
## Sample size:                      4500
## Number of independent variables:  12
## Mtry:                             3
## Target node size:                 2
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4188353
```

```r
spotify_rf %>% vip()
```

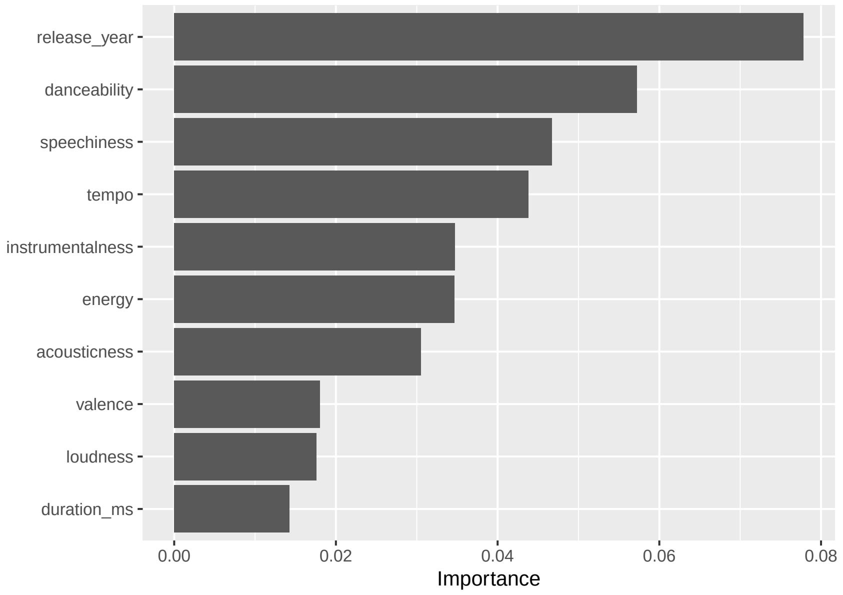

Figure 14: Variable importance for random forest model

```r
# Use model to predict test data
spotify_preds <- predict(spotify_rf,
                         new_data = spotify_test_preproc) %>%
  bind_cols (spotify_test_preproc %>%
               dplyr::select(playlist_genre))
```

```r
# Sens and Specs table
conf_mat <- confusionMatrix(data= spotify_preds$.pred_class,reference = spotify_preds$playlist_genre)
conf_mat$byClass %>% as_tibble()%>%
  bind_cols(conf_mat$byClass%>% rownames()) %>%
  dplyr::select(...12,Sensitivity,Specificity) %>%
  rename(Genre = ...12) %>%
  mutate (Genre = str_sub(Genre,8)) %>%
  kable(caption = "Sensitivity and Specificity by Genre")
```

Table 5: Sensitivity and Specificity by Genre

Genre	Sensitivity	Specificity
edm	0.680	0.9264
latin	0.496	0.9240
pop	0.444	0.8712
r&b	0.480	0.9240
rap	0.640	0.9160
rock	0.816	0.9496

```
conf_mat$byClass %>% as_tibble()%>%
  bind_cols(conf_mat$byClass%>% rownames()) %>%
  dplyr::select(...12,Sensitivity,Specificity) %>%
  rename(Genre = ...12) %>%
  mutate (Genre = str_sub(Genre,8)) %>%
  gather (key = "Metric", value ="Value", Sensitivity:Specificity) %>%
  ggplot (aes (x =Genre, y = Value, fill = Genre))+
  geom_col(col = "black", alpha = 0.85, aes(linetype = Metric))+
  facet_wrap(~Metric) +
  geom_text(aes(label = round(Value,2)), vjust = -0.5)+
  theme_clean()+
  scale_fill_npg()
```

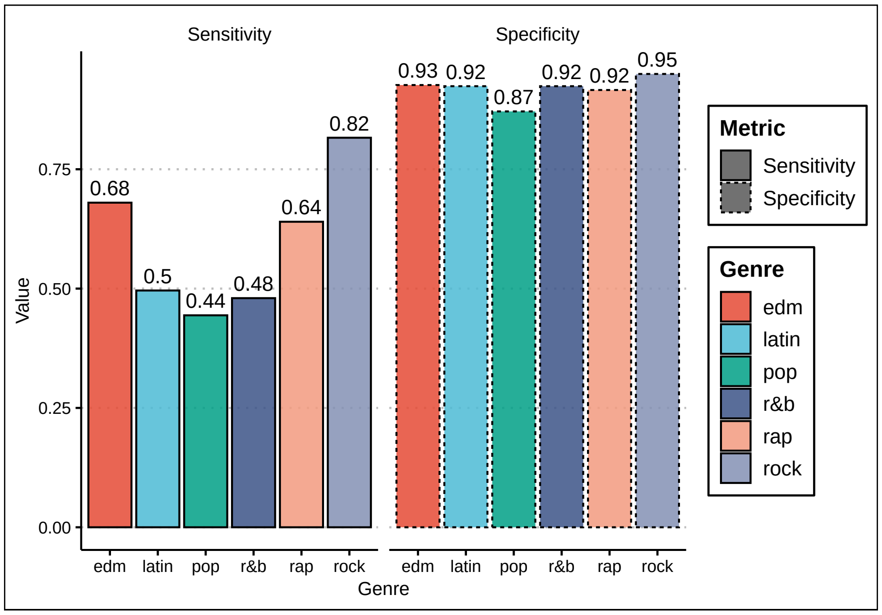

Figure 15: Sensitivity and Specificity by Genre

```
# plot confusion matrix
spotify_preds %>%
  conf_mat(truth = playlist_genre, estimate = .pred_class) %>%
  autoplot(type = "heatmap") + theme_clean()
```

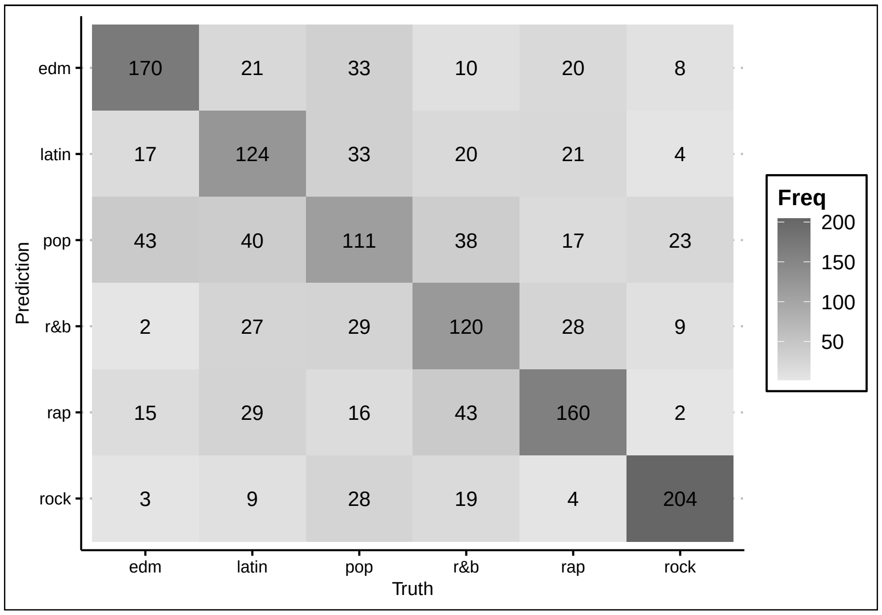

Figure 16: Confusion Matrix of predicting 6 different genre

```
conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction edm latin pop r&b rap rock
##      edm   170    21  33  10  20    8
##      latin  17   124  33  20  21    4
##      pop    43    40 111  38  17   23
##      r&b     2    27  29 120  28    9
##      rap    15    29  16  43 160    2
##      rock    3     9  28  19   4  204
##
## Overall Statistics
##
##                Accuracy : 0.5927
##                  95% CI : (0.5673, 0.6177)
##     No Information Rate : 0.1667
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5112
##
##  Mcnemar's Test P-Value : 0.06324
##
## Statistics by Class:
##
##                      Class: edm Class: latin Class: pop Class: r&b Class: rap
## Sensitivity              0.6800      0.49600     0.4440     0.4800     0.6400
## Specificity              0.9264      0.92400     0.8712     0.9240     0.9160
## Pos Pred Value           0.6489      0.56621     0.4081     0.5581     0.6038
## Neg Pred Value           0.9354      0.90164     0.8868     0.8988     0.9271
## Prevalence               0.1667      0.16667     0.1667     0.1667     0.1667
## Detection Rate           0.1133      0.08267     0.0740     0.0800     0.1067
## Detection Prevalence     0.1747      0.14600     0.1813     0.1433     0.1767
## Balanced Accuracy        0.8032      0.71000     0.6576     0.7020     0.7780
##                      Class: rock
## Sensitivity               0.8160
## Specificity               0.9496
## Pos Pred Value            0.7640
## Neg Pred Value            0.9627
## Prevalence                0.1667
## Detection Rate            0.1360
## Detection Prevalence      0.1780
## Balanced Accuracy         0.8828
```