Using Supervised Machine Learning Models to Classify Music into Genres

GDAA 1001 - Fundamentals of Spatial Analytics - Group Project

Mohit Francis (W0476572), Mahdieh Raoofpanah (W0480495), Earla Smith (W0472661)

2022-12-15

Contents

List of Figures

List of Tables

Introduction

Music is easy to understand, but difficult to authoritatively define. There are as many definitions of music as there are people on Earth. Many are wonderfully poetic: music has been called the 'universal language', or the 'ultimate form of self-expression' (Davies, 2012). Others focus on the psychological, cognitive, philosophical, or cultural aspects or effects of music. (Davies, 2012; Trehub et al., 2015).

From a data modelling perspective, a more functional definition is required which utilises the differences in musical characteristics to classify them into genres or subgenres. The following dictionary definition was used: 'an art of sound in time that expresses *ideas* and *emotions* in significant forms through the elements of *rhythm*, *melody*, *harmony*, and *color* [emphasis added]' (Dictionary.com, 2022). Rhythm, melody, harmony and colour (though colour is a nebuluous term) are some technical or theoretical characteristics of music, whereas ideas and emotions are perceptual characteristics.

Humans are incredibly adept at interpreting and feeling ideas and emotions present in music, primarily through perceiving the differences between various types of music, without necessarily understanding the the technicalities or theories underlying music (Malloch & Trevarthen, 2018). These differences can then be used to qualify or categorise music. For instance, listeners can intuitively understand the differences in ideas and emotions between a jazz song and a nursery rhyme, without necessarily understanding the various technical (i.e. rhythmic, harmonic or melodic) intricacies of each.

Additionally, listeners may not know if a song is written in Major or Minor mode, but they can easily 'feel' these differences: songs written in Major mode may make listeners 'happy'; songs written in Minor mode may make listeners 'sad' (Curtis & Bharucha, 2010). Listeners may not know the *Danceability* rating (see **Data Description**) of a song, but they can easily perceive which songs are more appropriate for dancing, relaxing, or exercising. In this way, listeners may easily classify music into genres (types of similarly related music), based on how they interpret music without necessarily understanding music.

In contrast, machine learning models cannot understand music on a perceptual level. However, they can easily understand the technicalities of music if these technicalities are defined for them, and if they are trained to do so. Machine learning models can find differences in volume, tempo, rhythm, frequency (and many others), and can differentiate between, and classify songs into, genres of music based on these characteristics. Musical characteristics, if not already quantified, must be quantified by people (i.e transformed into a numeric value) so models can utilise them. Even perceptual measures, such as the emotion in a song, can be transformed into numeric values such that machine learning models can make use of them.

Data Selection

Data Origins

This data set was downloaded from the *R for Data Science: Tidy Tuesday GitHub Repository* (Harmon & Grantham, 2020). Charlie Thompson, Josiah Parry, Donal Phipps, and Tom Wolff are the authors of `spotifyr` (Thompson, n.d.), an `R` package for pulling song audio features and other information from *Spotify*'s web application programming interface (API). Kaylan Pavlik (2019) used the `spotifyr` package to collect around 5000 songs and analyse them. Finally, Jon Harmon and Neal Grantham collected the rest of the songs in this data set (2020). While this data set was compiled by the above individuals, the data itself is from *Spotify*. This data set was chosen because it was already collected and complied, though the `Spotifyr` package could have been used to create a new data set.

Spotify is one of the largest music streaming services in the world, with over 456 million monthly active users, and over 76 million songs and counting (Spotify, 2022). On the back-end, *Spotify* collects various metrics for each song, some of which are included in this data set and described below, and many more which are not (Web API Reference | Spotify for Developers, n.d.). These audio features drive the many algorithms which *Spotify* uses to track the listening habits of its users, from their favourite songs, artists, and genres, to the duration of time spent listening to music (Web API Reference | Spotify for Developers, n.d.). *Spotify* also has

machine learning models trained on user-derived data which creates playlists of new music recommendations specific to each user (Web API Reference | Spotify for Developers, n.d.).

Purpose of Exercise

The purpose of this exercise is straightforward in theory. Many songs on *Spotify* are classified into many genres and sub-genres. In this data set, they have been assigned to a playlist of only one genre (six total) and one sub-genre (twenty-four total). The first part of this exercise focuses on which audio features best differentiate songs assigned to the six playlist genres. The second part of this exercise runs supervised machine learning models to classify the songs in this data set into genres using those audio features. Next, in order to find the best performing model, confusion matrices are derived and plotted. Finally, classification accuracies, sensitivities (true positive rate) and specificities (true negative rate), and variable importances are compared and analysed for each model.

In practice, this exercise is likely more difficult than it seems. Overall accuracy is expected to be quite low, with high rates of misclassification. Variable classification success by genre is also expected - some genres may be similar to each other in terms of audio features, whereas other genres may display greater separability. For the former group a lower classification accuracy is expected. For the latter group, a higher higher classification accuracy is expected.

Data Description

First the various libraries utilised over the course of this project are imported into the current session of `R`.

Next, the data set is imported as a comma-separated values file (.csv) from the *R for Data Science: Tidy Tuesday GitHub Repository*[1] into the current session of `R`. It is assigned to a variable called `spotify_songs`.

Using the `glimpse()` function, the data set is examined. There are twenty-three variables in this data set, comprising eleven categories (character string data types) and twelve audio features (numeric data types) which describe each song in the data set. The following descriptions for each variable are found on both the *R for Data Science Tidy Tuesday Readme* for this data set, as well as the *Spotify Developer Blog* (Harmon & Grantham, 2020, Web API Reference | Spotify for Developers, n.d.).

The variables can be divided into categorical descriptors, numeric descriptors, perceptual measures, and confidence measures (Pavlik, 2019). The categorical descriptors of this data set are the following:

- *track_id* (character string) is the unique identification code *Spotify* generates for each track on its service.
- *track_name* (character string) is the name of the track.
- *track_artist* (character string) is the name of the artist (or names of artists) performing the song.
- *track_popularity* (numeric) is the popularity of the song, rated on a scale from 0 to 100. This is based on total streams of a song, the frequency of song plays, and how recently the song has been played.
- *track_album_id* (character string) is the unique identification code *Spotify* generates for each album on its service.
- *track_album_name* (character string) is the name of the album on which the track was released.
- *track_album_release_date* (character string, but should be a date) is the release date of the album, and by extension, the track itself.

[1] https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-21/spotify_songs.csv

- *playlist_name* (character string) is the name of the playlist in which the track is found. Playlists are a list of songs which may or may not be organised according to type.
- *playlist_id* (character string) is the unique identification code *Spotify* generates for each playlist on its service.
- *playlist_genre* (character string) is the genre of the playlist. In this data set, each song has been assigned to one playlist of a specific genre.
- *playlist_subgenre* (character string) is the sub-genre of a playlist. In this data set, each song has been assigned to one sub-genre within a playlist of a specific genre.

The numeric descriptors of this data set are the following:

- *tempo* (numeric) is the overall average estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- *duration_ms* (numeric) is the length of a song in milliseconds.
- *key* (numeric) is the estimated overall key of the track. The coded integers map to pitches using standard twelve-note pitch class notation, for instance: 0 = C, 1 = C sharp/D flat, 2 = D, etc. If no key is detected, as in the case of atonal music, the value is -1.
- *mode* (numeric, but should be boolean) indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Minor is represented by 0 and major is 1.

The perceptual measures of this data set are the following:

- *danceability* (numeric) describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability (regular or irregular beat), beat strength (proportion of stressed to unstressed beats), and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- *energy* (numeric) is a measure from 0.0 to 1.0 and represents perceived intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while folk music scores lower. Perceptual features contributing to this attribute include dynamic range (the difference between the quietest and loudest sounds of a song), perceived loudness, timbre (the voicing or sound of an instrument), onset rate (the rate at which new sounds develop), and general entropy (chaos).
- *loudness* (numeric) refers to the overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 dB, where 0 dB is the point above which audio clipping (distortion) occurs, and there is an increased risk of digital audio equipment failure (if not hearing damage) occurring. This usage of loudness is in contrast to colloquial human usage, where 0 dB is the the quietest sound a human can perceive.
- *valence* (numeric) is a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

The confidence (probability) measures are the following:

- *speechiness* (numeric) detects the presence of spoken words in a track. The more exclusively speech-like the recording, the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- *acousticness* (numeric) is a confidence measure from 0.0 to 1.0 of whether the track uses acoustic instrumentation, i.e. 'actual' live instrumentation. 1.0 represents high confidence the track is acoustic, whereas 0 represents high confidence the track uses 'digital' or 'synthetic' instrumentation.
- *instrumentalness* (numeric) predicts whether a track contains no vocals. "Oohs" and "aahs" are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- *liveness* (numeric) detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

Of particular interest are the values within *playlist_genre* and *playlist_subgenre*. Using the `unique()` function, we examine the playlist genres and sub-genres to which each song has been assigned. The genres in this data set are: Pop, Rap, Rock, Latin, R&B (Rhythm and Blues), and EDM (Electronic Dance Music) (**Table 1**).

Table 1: The Six Genres in the Data Set.

x
pop
rap
rock
latin
r&b
edm

In this data set, there are four sub-genres for each playlist genre, comprising twenty-four sub-genres in total (**Table 2**). (This is by no means an exhaustive list of every sub-genre of each genre.) The four sub-genres of Pop are: Dance Pop, Post-Teen Pop, Electropop, and Indie Poptimism. The four sub-genres of Rap are Hip-Hop, Southern Hip-Hop, Gangster Rap and Trap. The four sub-genres of Rock are: Album Rock, Classic Rock, Permanent Wave, and Album Rock. The four sub-genres of Latin are: Tropical, Latin Pop, Reggaeton, Latin Hip-Hop. The four sub-genres of R&B are Urban Contemporary, Hip-Pop, New Jack Swing, and Neo Soul. The four sub-genres of EDM are: Electro House, Big Room, Pop EDM, and Progressive Electro House. Note the similarity in sub-genres across the genres, e.g. Pop EDM in the EDM genre may just as easily be a sub-genre of Pop.

Data Preparation

The data set is mostly tidy, but additional tidying is required before it can be utilised. First, a new column called *Duration* is mutated, which is the duration of each song in seconds. The values in *duration_ms* are taken and divided by 1000. While it is best to retain units in variable names, variable names must be kept as simple as possible (i.e., single words with no spaces or special characters) when creating and executing classification models. As a result, the units for *Duration* are not included.

Next, all columns unnecessary for this classification exercise are removed. These columns include: *track_id*, *track_name*, *track_artist*, *track_popularity*, *track_album_id*, *track_album_name*,

Table 2: The Twenty-Four Sub-Genres in the Data Set.

x
dance pop
post-teen pop
electropop
indie poptimism
hip hop
southern hip hop
gangster rap
trap
album rock
classic rock
permanent wave
hard rock
tropical
latin pop
reggaeton
latin hip hop
urban contemporary
hip pop
new jack swing
neo soul
electro house
big room
pop edm
progressive electro house

track_album_release_date, *playlist_name*, *playlist_id*, and *duration_ms*. This leaves *playlist_genre* and *playlist_subgenre* as the categorical variables, and all of the audio features: *danceability*, *energy*, *key*, *loudness*, *mode*, *speechiness*, *acousticness*, *instrumentalness*, *liveness*, *valence*, *tempo*, and *Duration*. These selected columns are stored in a new variable called `spotify_songs_tidy`.

After selecting the necessary columns, the column names are renamed. As described above for *Duration*, the names are simplified to single words and capitalised. For example, *playlist_genre* becomes *Genre*, and *liveness* becomes *Liveness*.

The final part of the initial data tidying process involves changing some of the values within the data set to make them more meaningful or more aesthetically pleasing. This saves time when plotting charts during the later portions of this exercise. Several `ifelse()` functions are used (nesting them if necessary) to accomplish this.

The values in the *Key* column are numbers ranging from 0 to 11, where the numbers are coded values referring to the keys: 0 is C, 1 is C sharp/D flat, 2 is D, 3 is D sharp/E flat, 4 is E, 5 is F, 6 is F sharp/G flat, 7 is G, 8 is G sharp/A flat, 9 is A, 10 is A sharp/B flat, and 11 is B. These numbers are replaced with the names of the actual keys.

Additionally, the values in the *Mode* column are either 0 or 1, each a numeric code where 0 refers to songs in Minor mode and 1 refers to songs in Major mode. These numbers are replaced with the modality of the song. Finally, all *Genre* and *SubGenre* values are capitalised.

The audio features used for classification are already numeric, so additional transformations are unnecessary. Furthermore, many of the the audio features in this data set are already normalised, so no additional tidying and processing is necessary because these values are in a format which makes them easy to compare. However, there may be potential outliers in *Duration*. These need to be visualised first before they can be normalised or removed.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) can be simply defined as 'the critical process of performing initial investigations on the data so as to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations' (Behrens, 1997). While statistical models may be used, EDA is primarily for exploring and examining the the distribution of the data to formulate hypotheses which could lead to new data collection and experiments. Visualising the data allows for easier comparison of different plots.

EDA is different from Initial Data Analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, handling missing values, and making transformations of variables as needed. EDA is also separate from Statistical Graphing, as the latter is mainly performed for presentation and publication purposes, though EDA uses many techniques of Statistical Graphing.

Bar graphs are used to examine variation in categorical data. Histograms, box plots, and density plots are used to explore variation in numeric data. Heat maps are used to examine covariation between categorical variables. Scatter plots and scatter plot matrices are used to explore covariation between numeric variables.

Examining Variation in Categorical Data

Visualising a variable depends on whether the variable is categorical (discrete) or continuous in nature. A variable is categorical if it takes a small number of discrete or binned values. In `R`, categorical variables are typically saved as 'factors' or 'character' variables. Examining variation of categorical data requires the use of bar charts.

There are only two categorical variables in the tided data set: *Genres* and *SubGenres*. **Figure 1** displays the number of songs (on the y-axis) in each of the six genres (on the x-axis) in the data set in order of decreasing frequency. The majority of tracks (approximately 6000) belong to the EDM genre, while Rock contains the fewest songs.

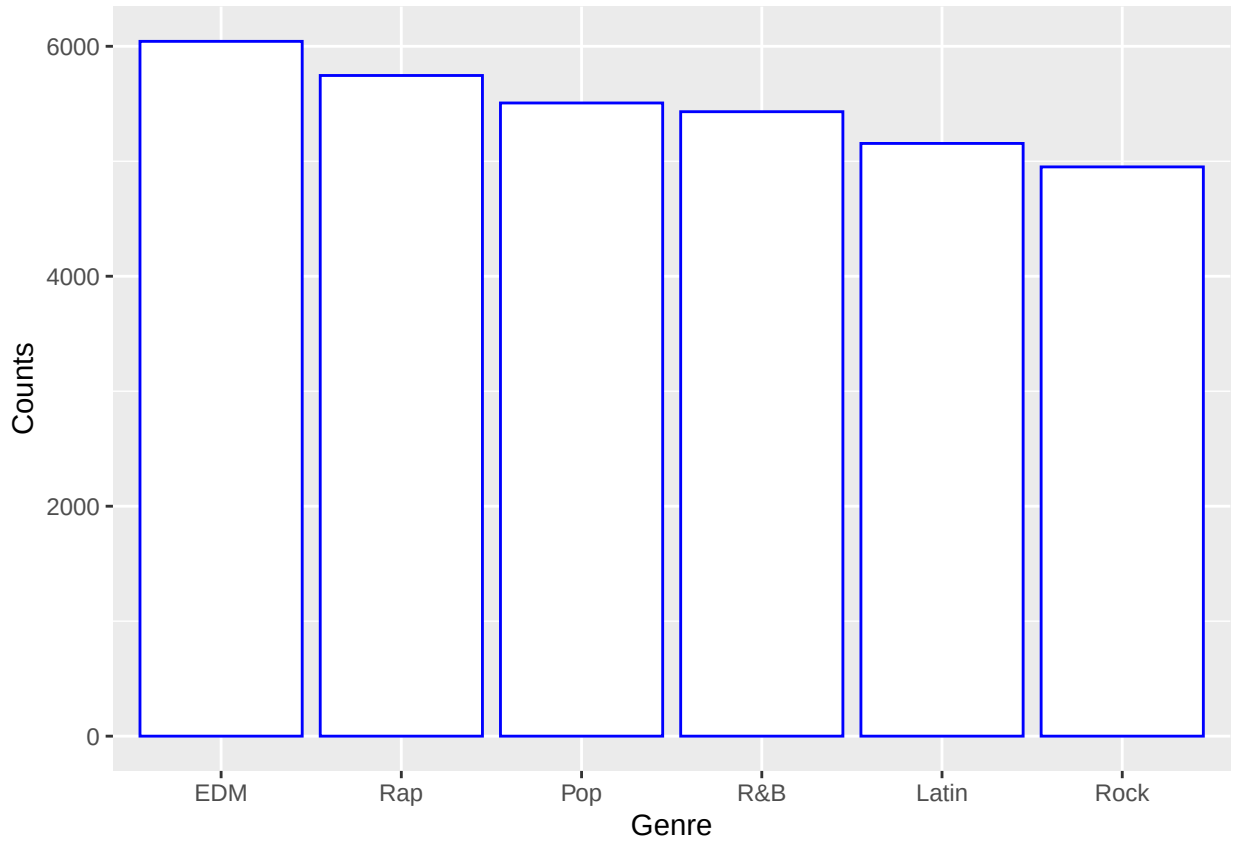

Figure 1: Bar Graph of Number of Songs in each Genre

Figure 2 displays the number of songs (on the y-axis) in each of the twenty-four sub-genres (x-axis). Progressive Electronic House contains 1750 songs, the most of any sub-genre. Reggaeton (blended reggae music with Latin American dance hall music) has the fewest number of songs. Southern Hip-Hop, Neo Soul (current R&B), and Indie Poptimism each have over 1500 tracks allocated to them. Most other sub-genres have fewer than 1500 songs assigned to them.

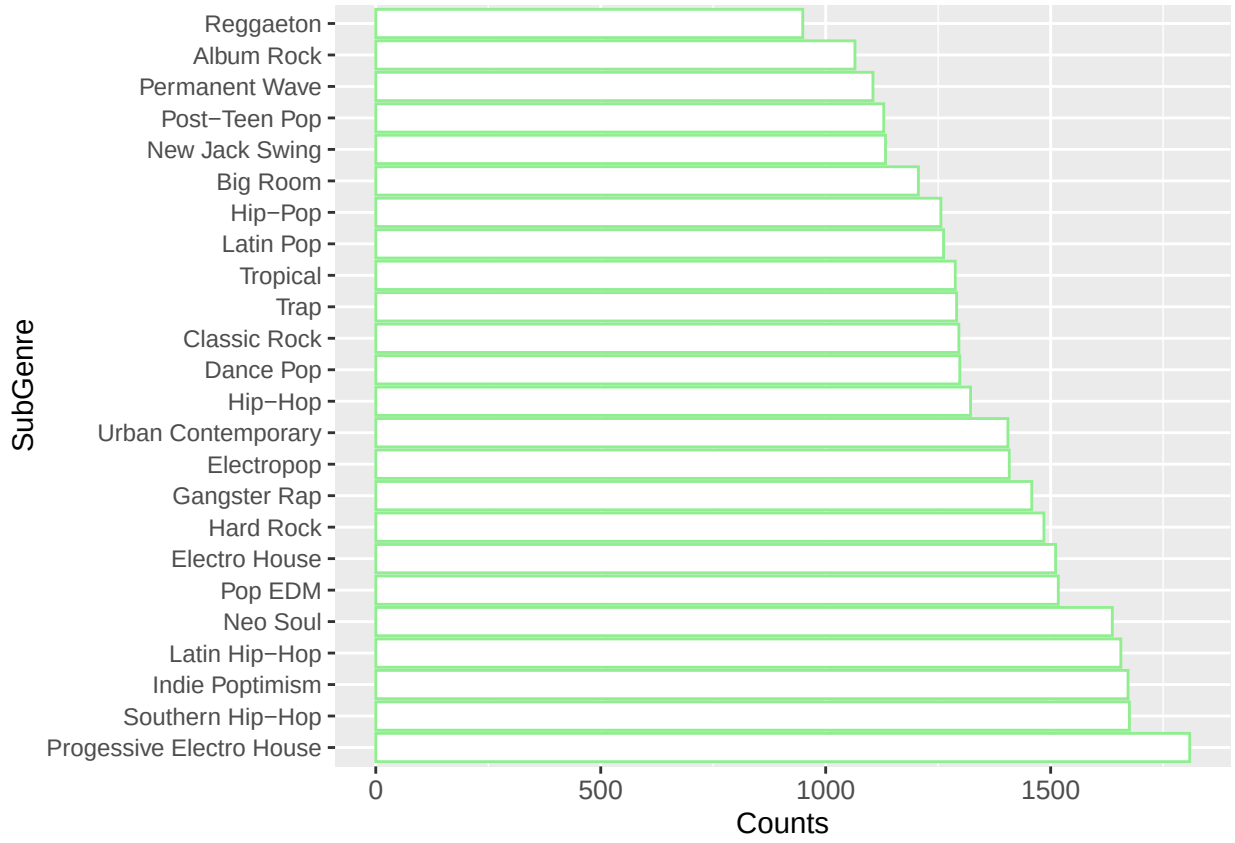

Figure 2: Bar Graph of Number of Songs in each Subgenre

Examining Variation in Numeric Data

For continuous variables, a greater variety of statistical tools are available. For example, histograms can be plotted to examine the frequency and distribution of the values within a continuous variable.

Figure 3 shows the distribution of the *Loudness* of each song in the data set as a histogram. Initially, the graph increases exponentially till -6 dB. The number of songs declines once -6 dB of *Loudness* is reached, demonstrating that as *Loudness* rises, the number of songs increases. There are a few songs above 0 dB and below -35 dB. Additionally, no songs with a *Loudness* below -45 dB are observed.

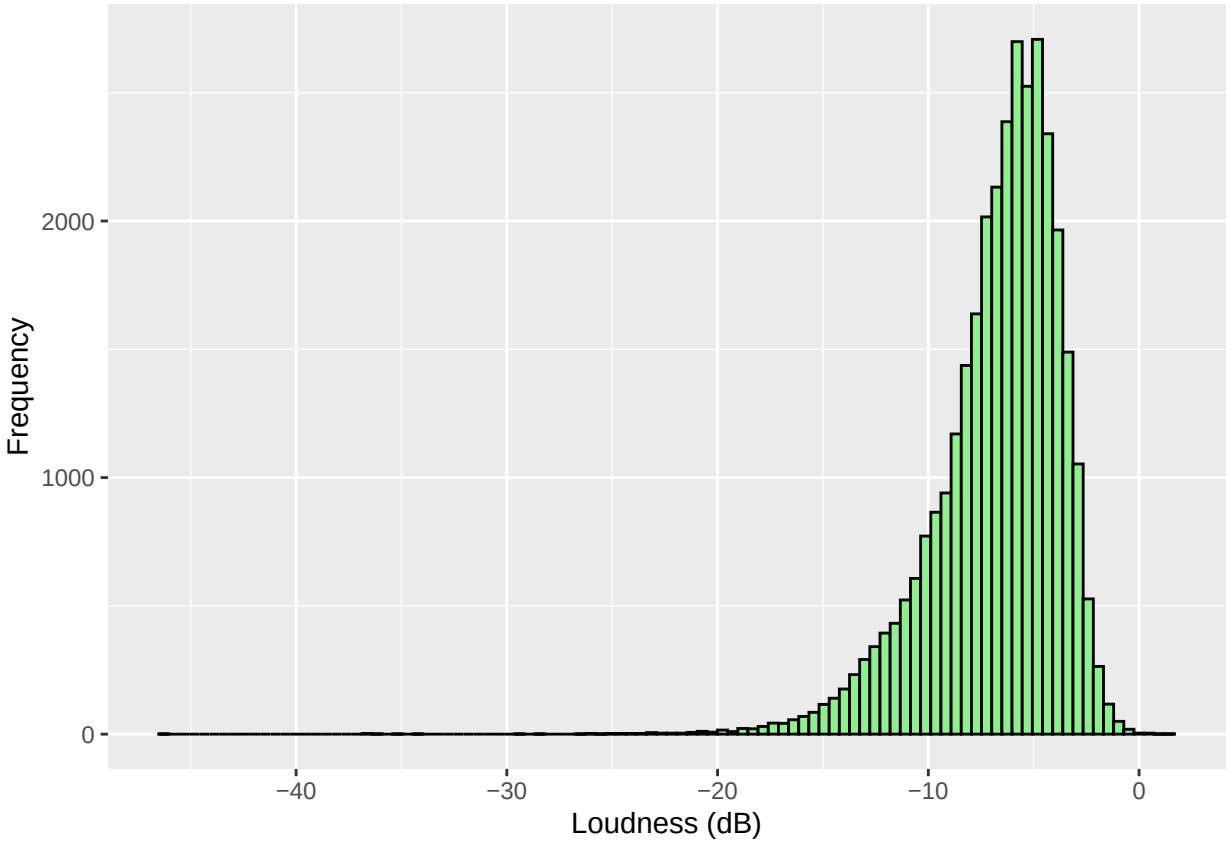

Figure 3: Histogram Displaying the Distribution of Loudness (dB)

Figure 4 displays the distribution of *Energy* of the songs in this data set, also as a histogram. *Energy* is the perceived intensity and activity of a song and ranges from 0.0 to 1.0. A left-skewed distribution is clearly visible, suggesting that most tracks in this data set are active and energetic. Most songs have an *Energy* value between 0.37 to 0.87.

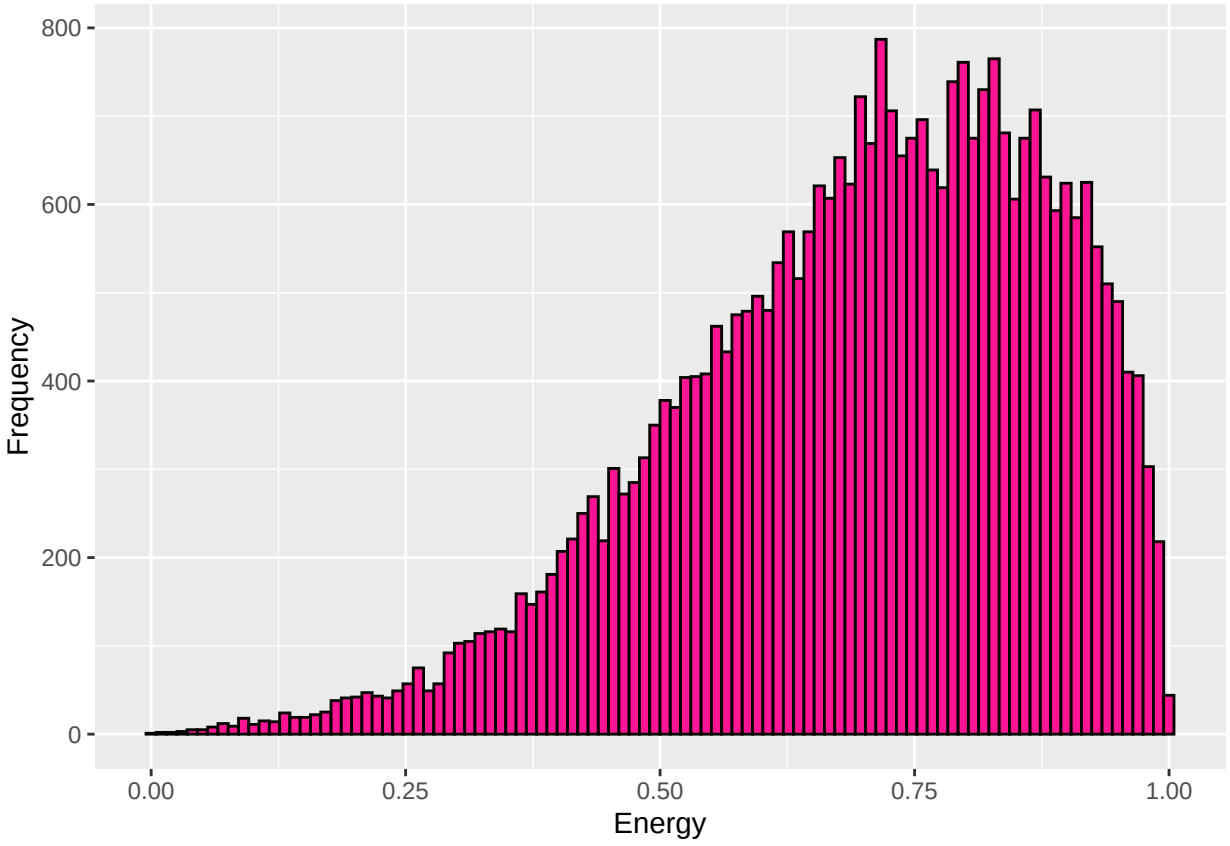

Figure 4: Histogram Displaying the Distribution of Energy

A density plot, like a histogram, displays the distribution of a continuous variable. Instead of frequency, it displays a probability density function of the variable using a kernel density estimate. A multiplicate bandwidth modification is applied to this density plot. As a result, the bandwidth can be changed while still using a bandwidth estimator.

Figure 5 is a density plot of the *Tempo* of the songs in this data set. The distribution appears to be bimodal, with the main peak around 125 BPM and a lower peak around 90 BPM. The plot starts at 0 BPM and ends at 240 BPM. The number of songs with *Tempo* higher than 200 is insignificant.

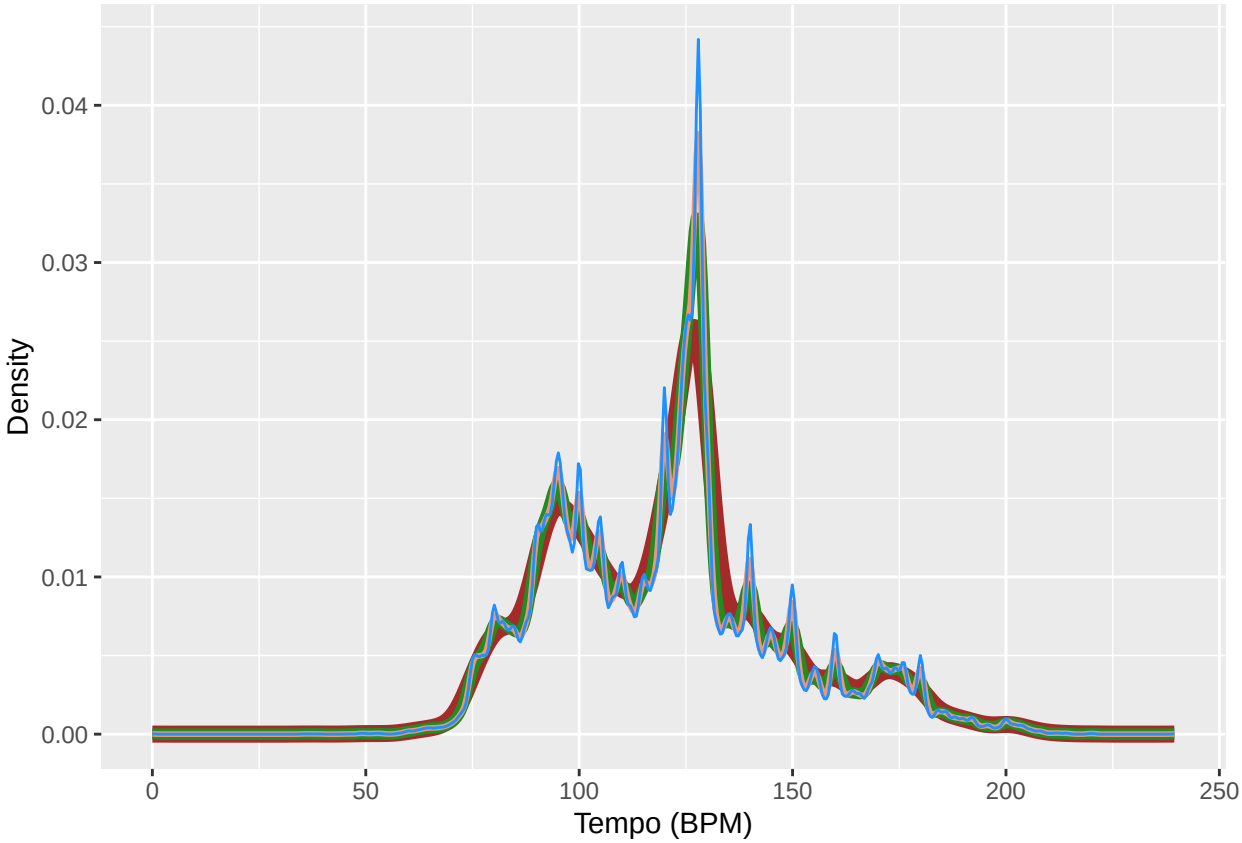

Figure 5: Density Plot of Tempo (BPM)

As seen in **Figure 5**, when data for a variable corresponds to multiple groups, visualising the data for each group separately can be much more useful. For **Figure 6**, the `facet_wrap()` argument was used to create six separate density plots - one for each *Genre* - and their respective *Tempo* densities.

In almost all *Genres*, *Tempo* follows a fairly normal distribution. However, the *Tempo* of EDM rises sharply at 125 BPM before falling again, indicating that most EDM songs have a higher probability of having a *Tempo* of 125 BPM.

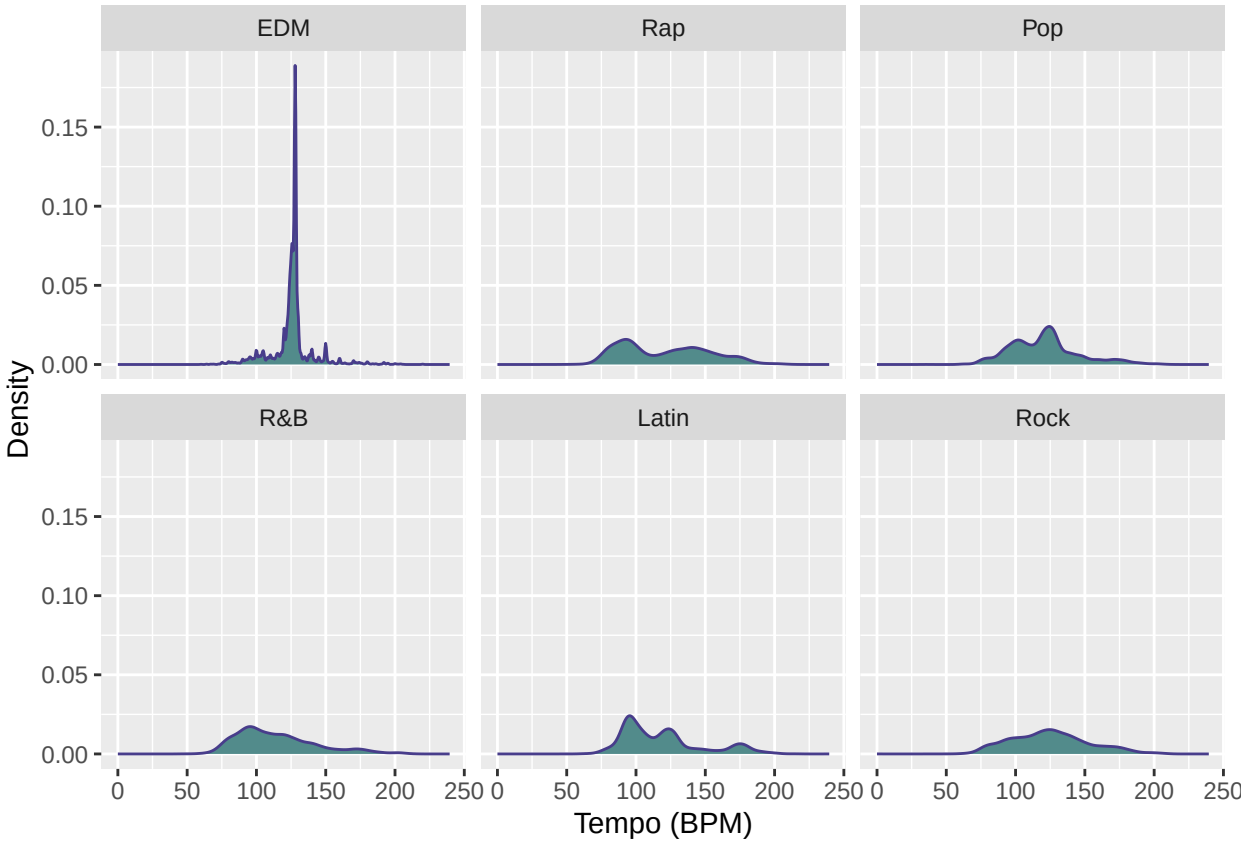

Figure 6: Density Plot of Tempo (BPM) of Songs by Genre

A box plot is a type of visual shorthand, popular among statisticians, to examine the five number summary statistics of the values in a variable. Each box plot consists of a box that stretches from the 25th Percentile (First Quartile) of the distribution to the 75th Percentile (Third Quartile), a distance known as the Interquartile Range (IQR). In the middle of the box is a line which displays the median - the central value of the distribution.

These three lines provide a sense of the spread of the distribution, and whether or not the distribution is symmetric about the median or skewed to one side. Visual points display observations greater than 1.5 times the IQR from either edge of the box. These outlying points - known as outliers - are unusual so are plotted individually. A line (or whisker) extends from each end of the box to the farthest non-outlier point in the distribution.

Figure 7 displays a box plot of songs and their *Duration*. The First Quartile point is at 187.8 seconds and the Third Quartile is at 253.6 seconds. The median *Duration* is 216 seconds, which is little skewed to the left. The purpose of utilizing a box plot for *Duration* is to discover outliers. The farthest non-outliers have a *Duration* of 100 seconds or 350 seconds, indicating songs which are less than 1.5 minutes or more than 5.8 minutes in *Duration* are outliers. The outliers have a *Duration* more than 1.5 times the IQR, i.e. a *Duration* lower than 187.8 seconds, or higher than 253.6 seconds. The shortest song duration is 4 seconds and the longest is 517.8 seconds.

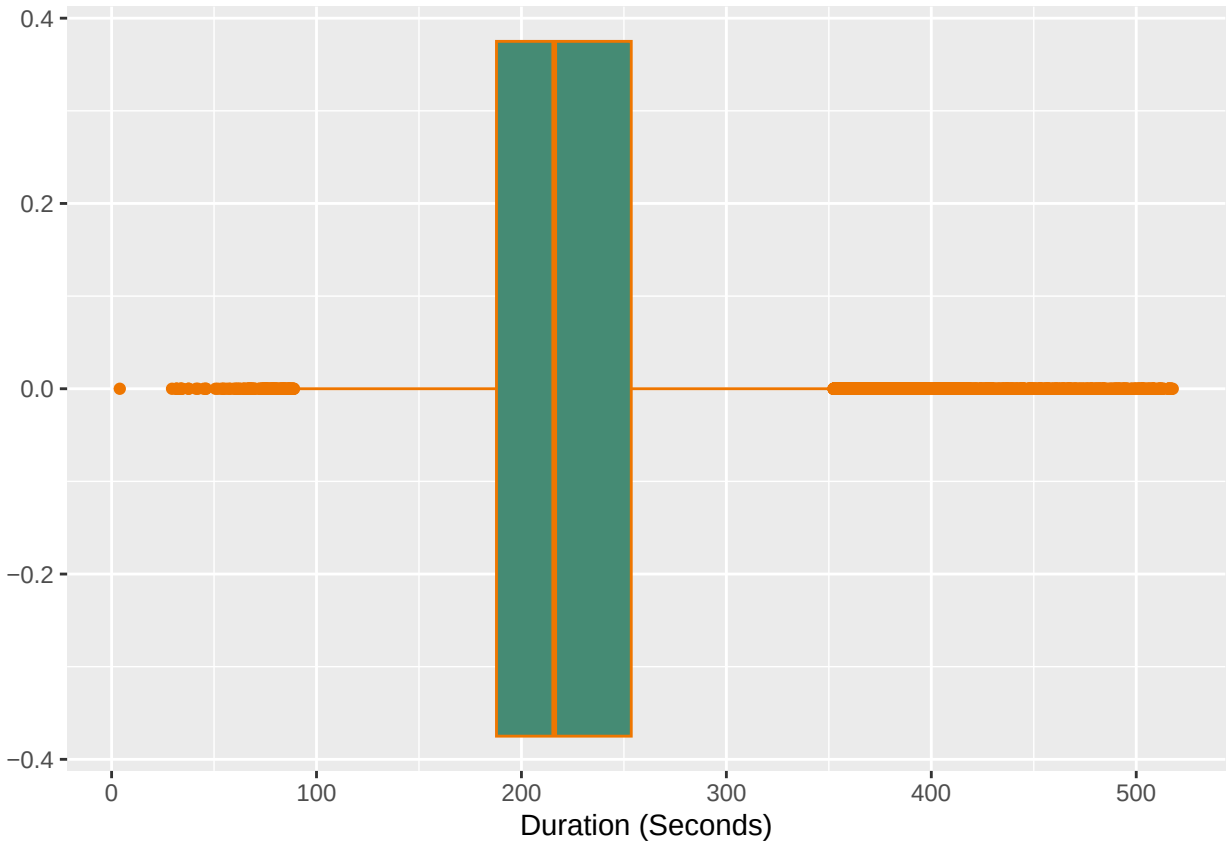

Figure 7: Box Plot of Song Duration (Seconds)

Figure 8 displays another box plot, visualising which *Genres* to which the extreme outliers in song *Duration* belong. *Genre* is plotted on the y-axis and *Duration* on the x-axis. The 4 second track is a Rock song and the longest is an EDM song. The white star also shows the mean song *Duration* for each *Genre*, which was always higher than the median, demonstrating the adverse effect of the outliers on the shape of the data set. Generally, the mean stayed between 200 to 250 seconds.

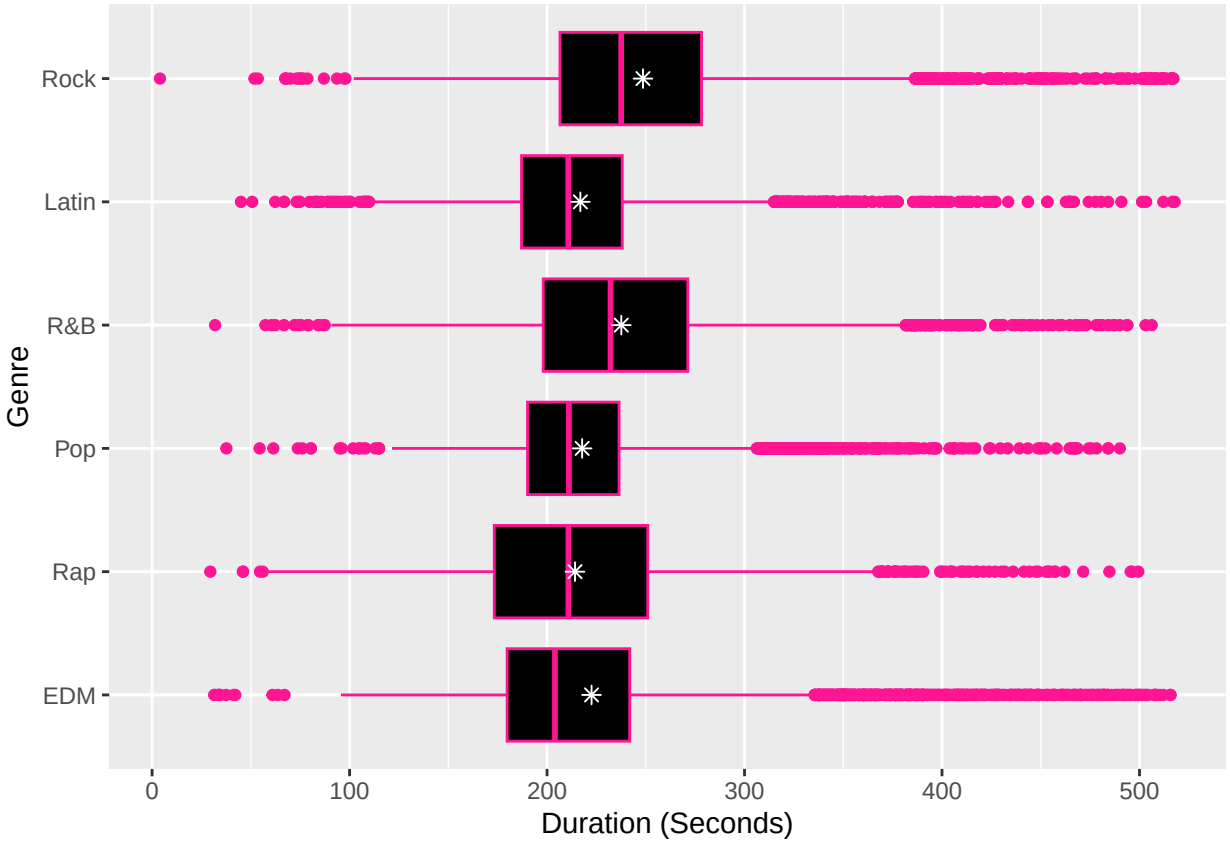

Figure 8: Box Plot of Song Duration (Seconds) by Genre

Examining Covariation Using Categorical Data and Binned Continuous Data

A heat map is a data visualization technique showing the magnitude of a phenomenon as colour in two dimensions (i.e. for two variables). The colour may differ in hue or intensity, but it provides obvious visual cues of how the phenomenon is clustered or varies over space. Typically it is used for categorical data.

Figure 9 displays the distribution of *Danceability Deciles* (i.e. *Danceability* binned into the following ranges: 0-0.1, 0.11-0.20, 0.21-0.30, 0.31-0.40, 0.41-0.50, 0.51-0.60, 0.61-0.70, 0.71-0.80, 0.81-0.90, 0.91-1) by *Genre*. 'Warmer' colours (brown) indicate a higher frequency of songs in that particular *Danceability Decile*, whereas 'cooler' colours (yellow) indicate a lower frequency of songs in that particular *Danceability Decile*. It is immediately apparent that Rock music is rated the lowest in *Danceability*, whereas Rap music is rated the highest in *Danceability*, followed by Latin and R&B. Interestingly, despite its name, EDM is mostly moderate, but variable in terms of *Danceability*.

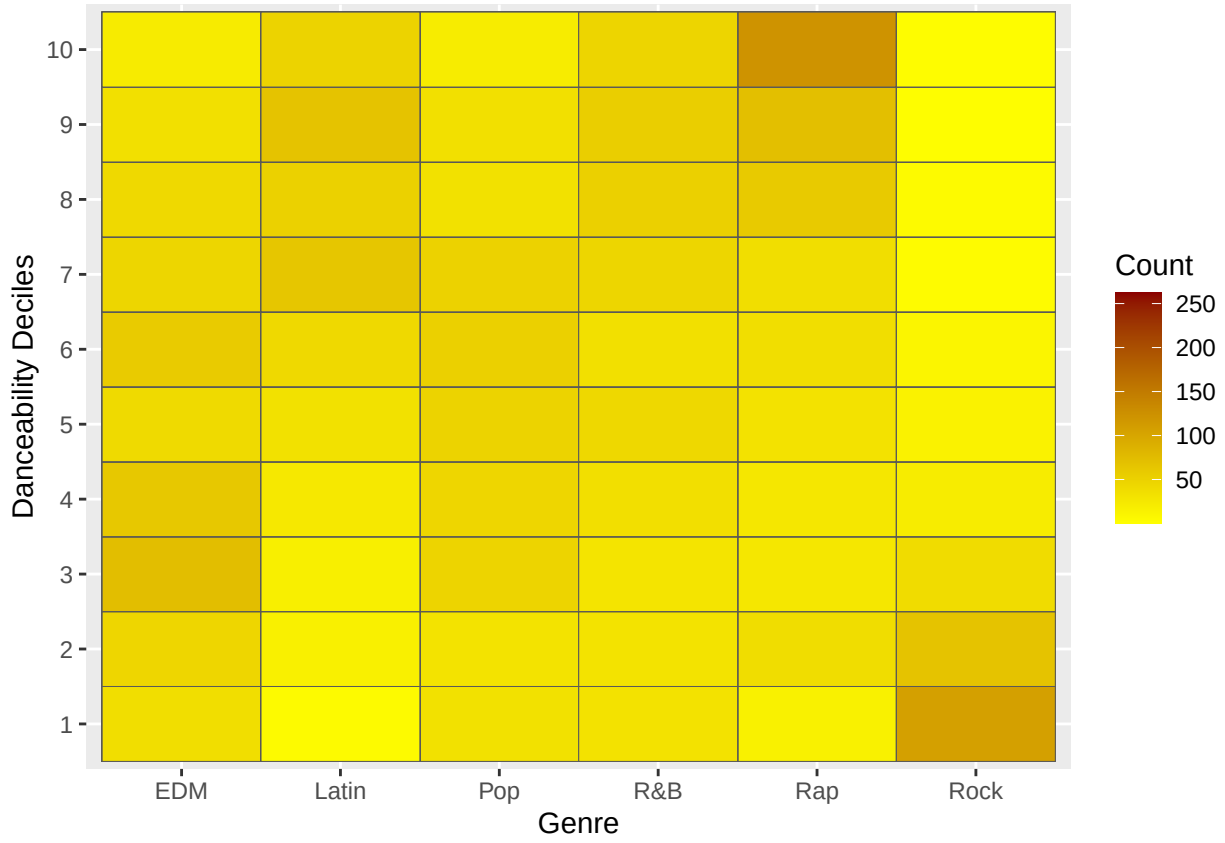

Figure 9: Heat Map of Genre and Danceability Deciles

In the absence of adequate categorical data, continuous data may also be binned into discrete values. **Figure 10** displays the relationship between *Energy* and *Valence*, two continuous variables which were binned. There is a positive relationship between the two: the higher the *Energy* of a song, the more 'positive' the song. Simply stated, songs with higher *Energy* levels, approximately 0.5 or above, are more cheerful and happy.

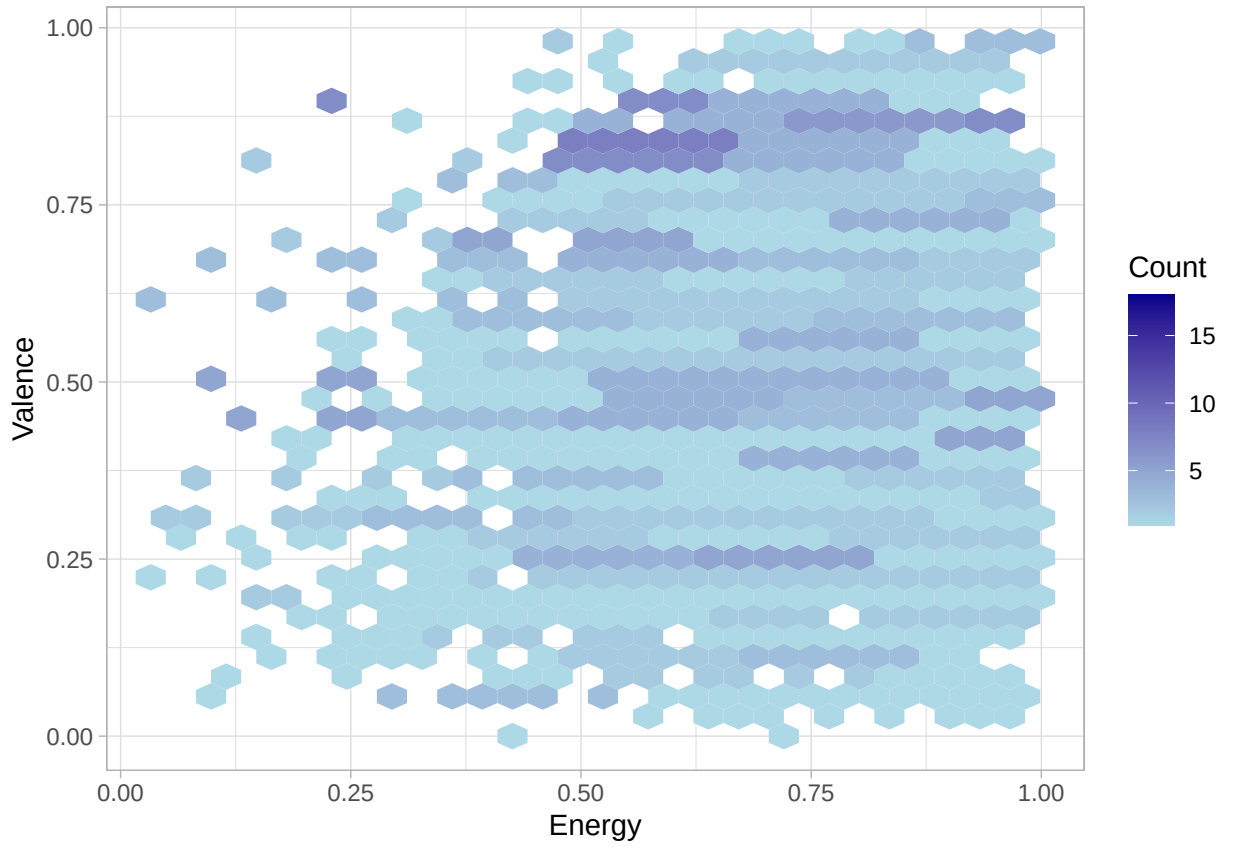

Figure 10: Heat Map Displaying the Relationship Energy and Valence

Examining Covariation in Continuous Data

Scatter plots are commonly used in EDA. They are essentially two-dimensional histograms (i.e. the histograms of two variables) plotted together as points to understand the relationship between the two variables. A scatter plot matrix is a grid (or matrix) of scatter plots used to depict bivariate relationships between combinations of several variables. Each scatter plot in the matrix depicts the relationship between two variables, allowing several correlations to be investigated in a single graphic.

Figure 11 is a scatter plot matrix of all audio features except *Key* and *Mode*, i.e. *Tempo*, *Liveness*, *Valence*, *Loudness*, *Acousticness*, *Instrumentalness*, *Danceability*, *Energy*, *Speechiness*, and *Duration*, with *Genres* highlighted. *Key* and *Mode* were not plotted because while they are numeric, they are discrete in nature. The matrix was plotted first to discover interesting relationships which required further examination. Most variables have a negative or inverse correlation, which means that when one variable increases, the other variable decreases, and vice versa.

As previously observed, there is a negative correlation between *Loudness* and *Acousticness*: as *Loudness* increases, *Acousticness* decreases. The loudest *Genre* is EDM, which has the lowest *Acousticness* rating. This makes sense because 'acoustic' means 'without electrical amplification', i.e. the use of live, unamplified instrumentation. EDM is defined by the use of digital instrumentation, which explains its more electronic sounds.

Nonetheless, some positive correlations exist, where if one variable increases, the other variable also increases. This can be observed in the relationships between *Liveness* and *Loudness*, and *Instrumentalness* and *Liveness*. These relationships also make sense. 'Live' music, or music performed in front of an audience, must be played

at a high enough volume to fill the venue. Additionally, with the exception of EDM and Rap music, most music played live uses a high degree of instrumentation.

Certain outliers in the data set are also observed, such as those relating to *Acousticness* and *Loudness* or *Loudness* and *Valence*, where the shapes of the relationships are almost vertical and skewed entirely to one side.

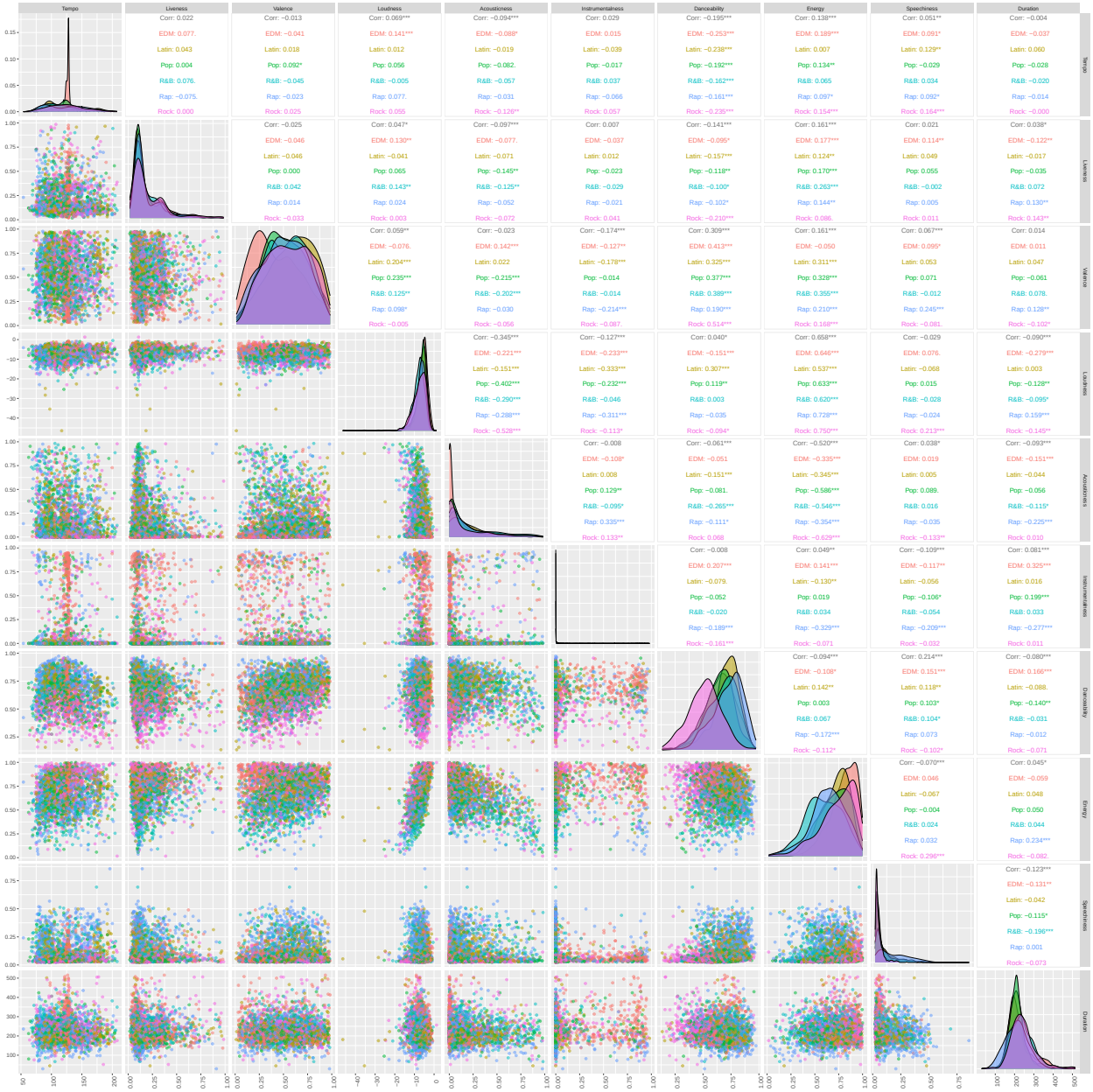

Figure 11: Scatter Plot Matrix of Tempo (BPM), Liveness, Valence, Loudness (dB), Acousticness, Instrumentalness, Danceability, Energy, Speechiness, and Duration (s)

Figure 12 is a scatter plot further examining the relationship between *Loudness* and *Energy*, colourised by *Genre*. with stacked histograms for each variable placed on the top and right margins. There is a strong positive relationship between the two variables: when the *Loudness* of a track increases, the *Energy* of a track also increases. Almost all EDM tracks have -15 dB *Loudness* and a value of more than 0.25 for *Energy*

- EDM is probably most energetic *Genre* compared to the others.

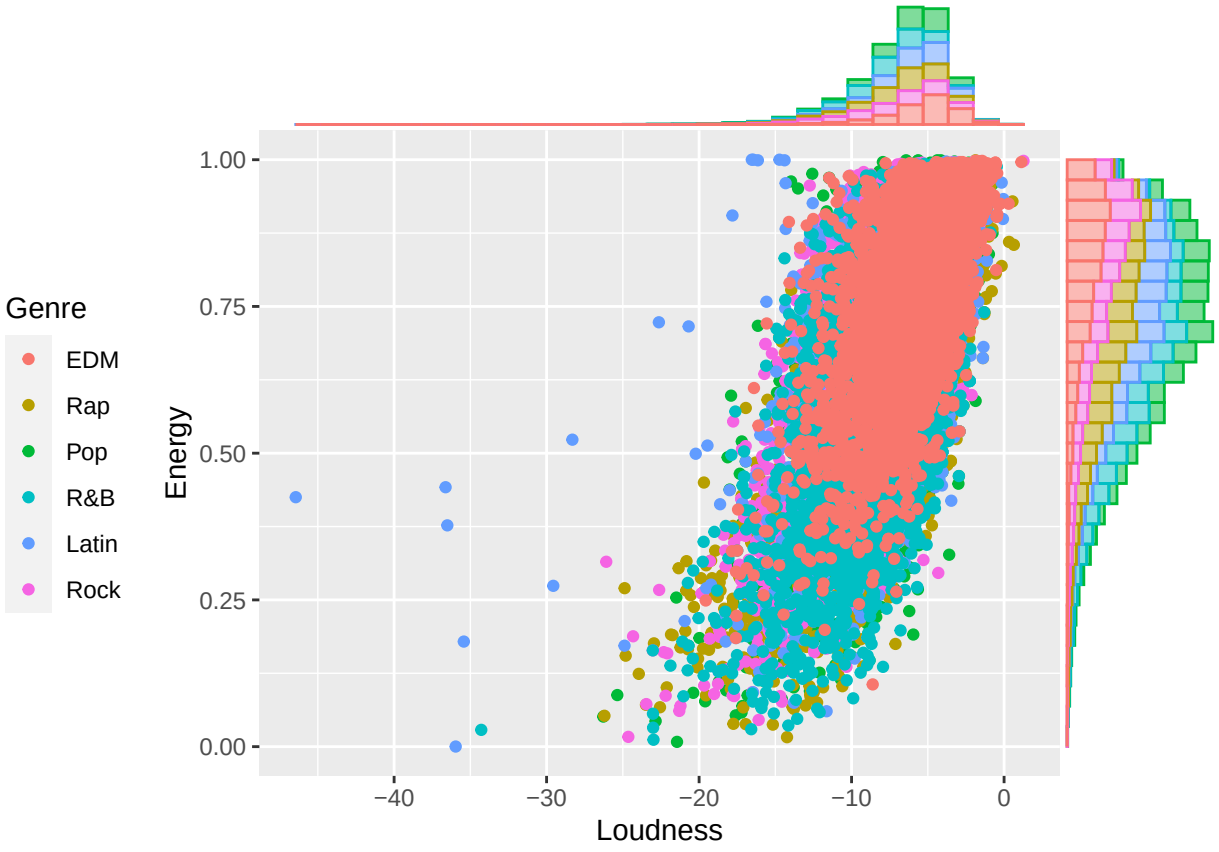

Figure 12: Scatter Plot of Loudness (dB) and Energy Coloured by Genre (with Marginal Histograms)

Figure 13 is a scatter plot showing the relationship between *Tempo* and *Acousticness* colourised by *Genre*, with stacked histograms for each variable placed on the top and right margins. Most of the songs in this data set score low in *Acousticness*, meaning they use digital instrumentation. As seen previously, *Tempo* shows a bimodal distribution - at 90 BPM and 125 BPM. There is no strong relationship observed between these two variables. Because of the strong left skew in the distribution of *Acousticness*, this scatter plot seems to mimic the *Tempo* density plot for each *Genre*, with the greatest concentration of all songs in lower range of *Acousticness* and the mid-*Tempo* region.

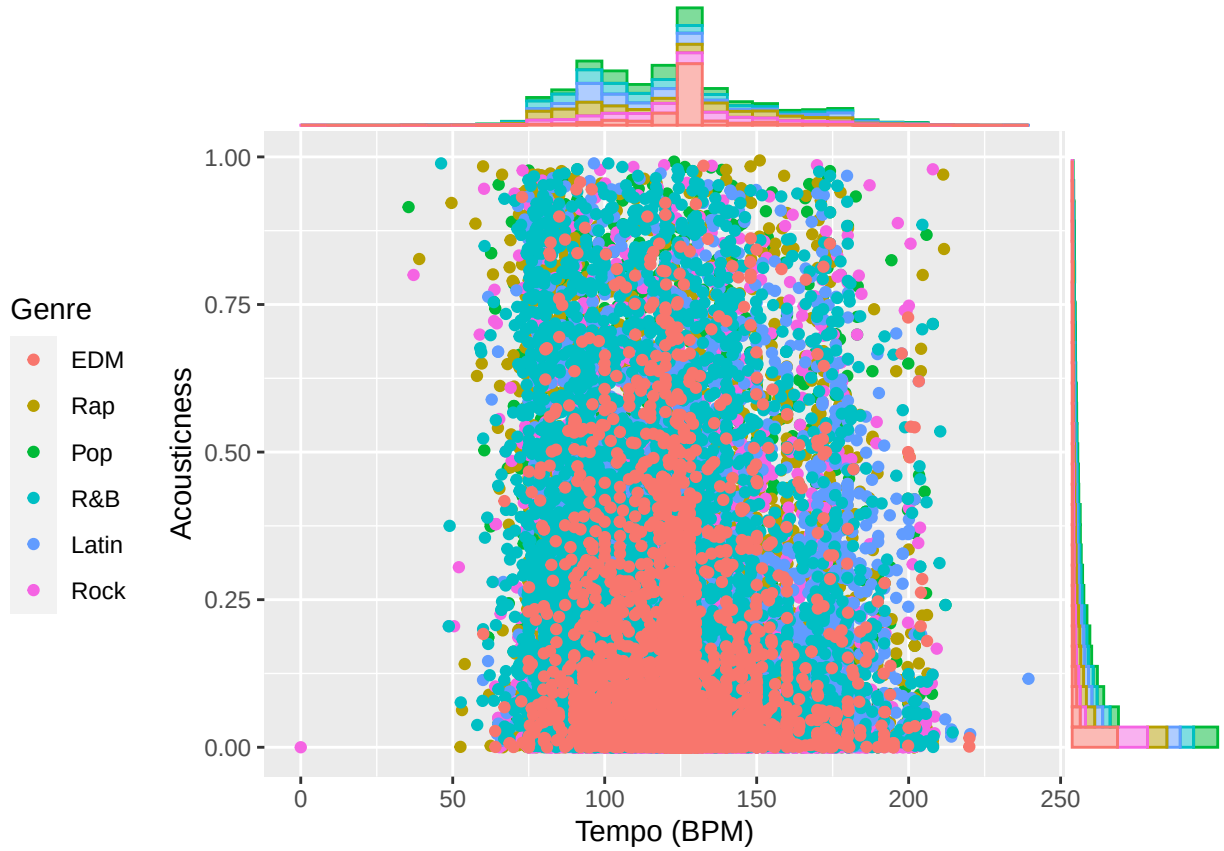

Figure 13: Scatter Plot of Tempo (BPM) and Acousticness Coloured by Genre (with Marginal Histograms)

However, if the scatter plot for *Tempo* and *Acousticness* is facet wrapped by *Genre*, a better sense of the relationship between the two variables can be observed. **Figure 14** shows a slight negative trend between *Tempo* and *Acousticness*, except in the case of Latin music, where no positive or negative trend is observed. This negative trend is more evident in Pop and Rock songs. It appears *Tempo* and *Acousticness* have a negative or inverse relationship.

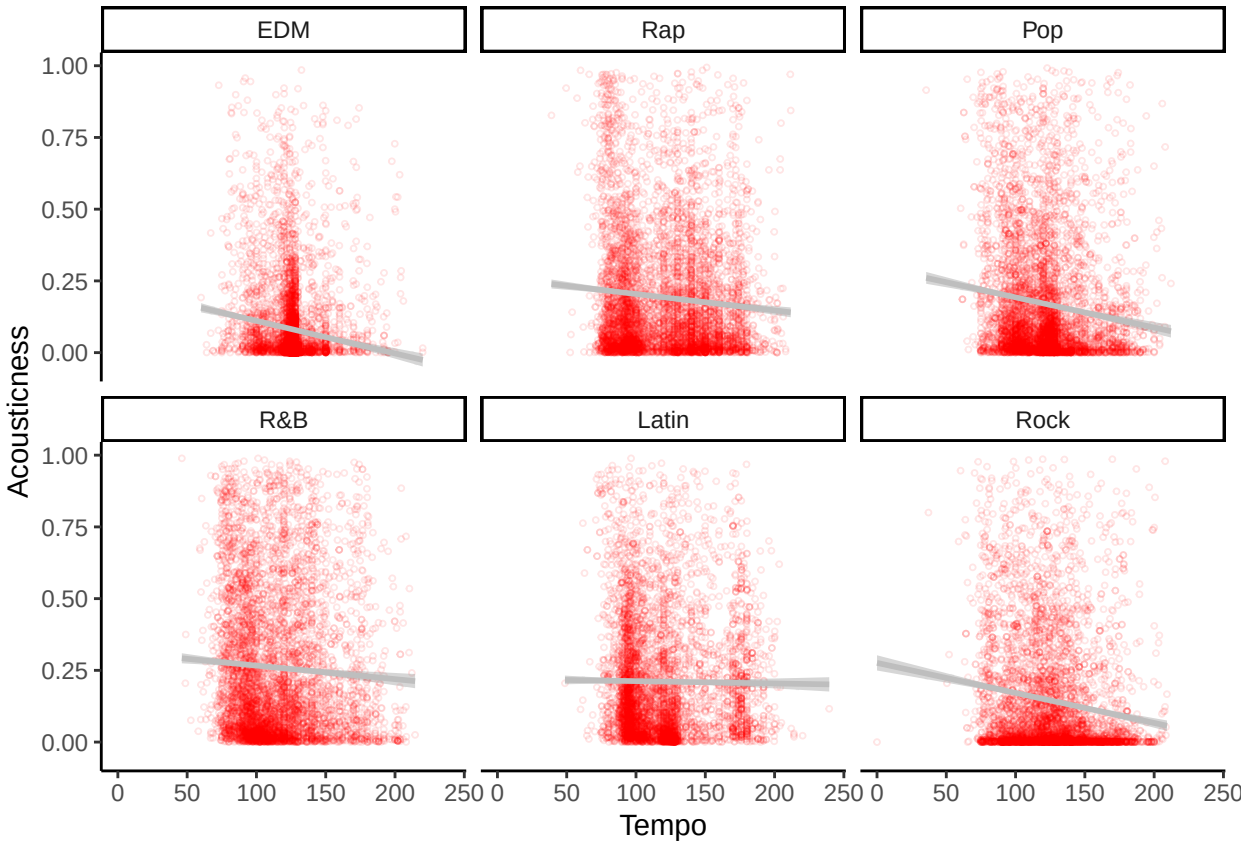

Figure 14: Scatter Plots of Tempo (BPM) and Acousticness Separated by Genre

Predictive Modelling

Downsampling Data Set, Creating Training and Validation Data

In order to further explore covariation and relationships within this data set, a number of predictive modelling processes were performed. *Genre* was used as the target variable, as this is a key quality of each song. *SubGenre* could also have been used, but there were twenty-four *SubGenres* as opposed to six *Genres* - too many target variables can greatly reduce classification accuracy. In the previous section, faceted plots explored the relationships of different variables by *Genre*, and this section expands on those findings.

In order to prepare the data for modelling, a new subset was created containing three thousand songs. The subset was assigned to `spotify_modelling`. Subsetting used a randomised process, and the same subset was used for each model. The proportion of songs per *Genre* was not exactly the same as the original data set, but was close enough such that substantial bias would not be introduced. *SubGenre* was also removed from the subset, as it is directly related to *Genre* and retaining it would confound the models. By removing *SubGenre*, the models would instead draw on the other variables in the data set, allowing for further exploration of the relationships between variables.

The seed, or initialisation state of a pseudo-random number generator, was set to 123. This maintains repeatability of the models. Additionally, *Genre*, *Key*, and *Mode* were factorised. Next, 80% of `spotify_modelling` was assigned to the training data and the rest to the validation data. The training data 'trains' the models, showing each model the songs belonging to each *Genre* (and song features). The

validation data is used to test model training and performance - observations from the validation data are assigned to *Genres* from the training data set.

Next, a count of the songs by *Genre* in the validation data set was undertaken to make sure the counts were relatively even across *Genres*. With the exception of EDM, of which there are 111 songs in the validation data set, there were approximately 100 songs for the other *Genres*. Finally, a 10-fold cross validation was set for the data set to estimate the 'skill' of the data set, i.e. the classification accuracy.

Decision Tree (Classification and Regression Tree)

The decision tree [2] incorporates five distinct variables in order to predict the *Genre* of a song: *Speechiness*, *Danceability*, *Energy*, *Instrumentalness*, *Duration*, and *Tempo*. *Speechiness* and *Instrumentalness* are both used twice. Each *Genre* within the data set is returned at least once, with Rap, EDM, and Latin being returned twice. The branches a song follows indicates qualities which generally describe the *Genre*.

These results indicate *Key*, *Mode* and *Loudness* may not be strongly correlated with a given *Genre*. The results also hint at the identifying qualities of each *Genre*, such as EDM displaying higher levels of *Instrumentalness* or Rap showing high values of *Speechiness*. This provides interesting, more complex insights into the relationships of these variables to *Genre* than the previous plots can provide. It is important to note that there were issues with viewing the confusion matrix of this model. As a result, classification accuracy could not derived; therefore, the efficacy of this model could not be evaluated. The decision tree model should be taken as a source of potential indicators, but not all songs will be correctly categorised if it were followed.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a linear model used for classification and dimensionality reduction (Dash, 2022). It is an improvement on *Logisitic Regression*, allowing for better analysis when using multiple classes. It makes two assumptions about a dataset: that all data has either a normal or Gaussian distribution, and that each class has identical relationships between covariants. These assumptions are what makes it a linear model, but it can still perform well if the data violates these assumptions (Xiaozhou, 2020; Dash, 2022).

[2] Note: visTree() does not display its outputs to a PDF document, therefore an image of the decision tree is included instead.

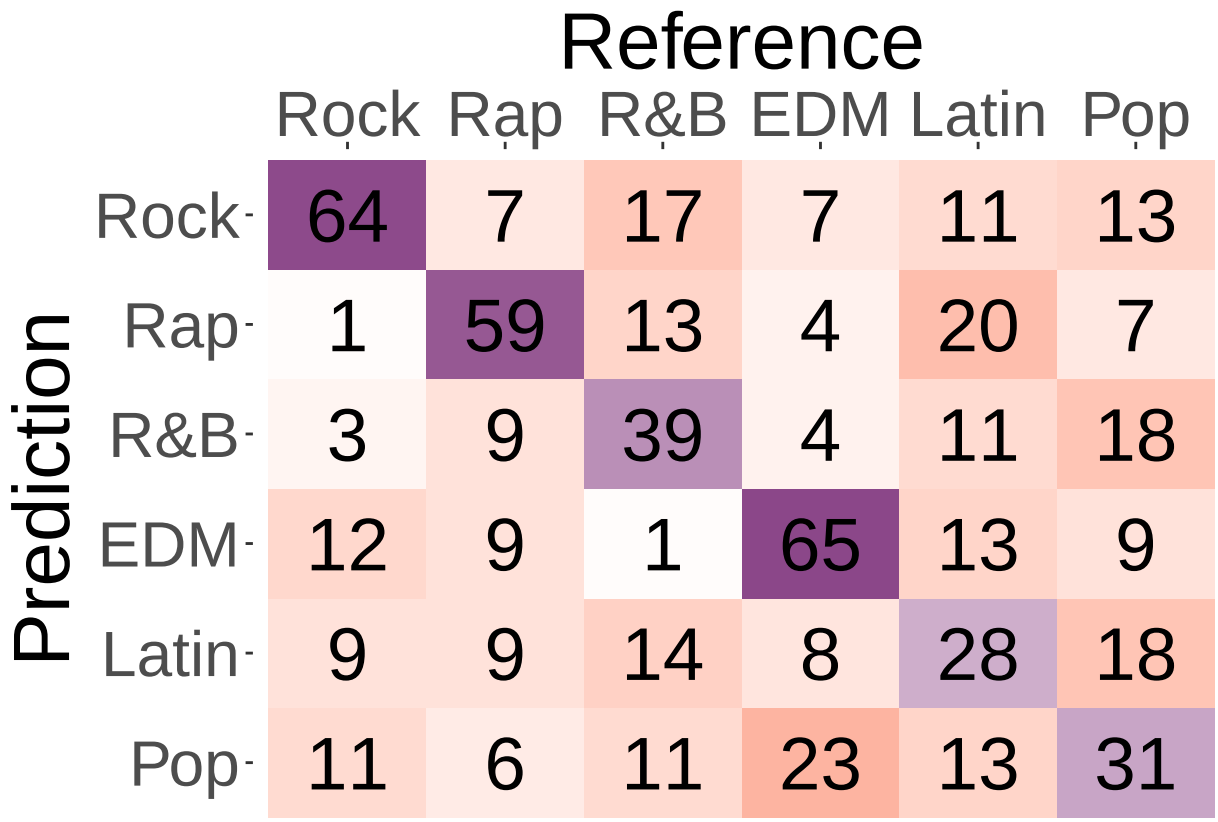

Figure 15: Confusion Matrix of the Linear Discriminant Analysis (LDA) Model

k-Nearest Neighbours (*k*NN) Algorithm

The *k-Nearest Neighbours* algorithm performs classifications by assigning a class to a data point based on the values of the data surrounding it. The classifiers checks k number of neighbouring points to determine which value should be assigned to the point. This algorithm does not make assumptions about the data, which makes it suitable for non-linear data. Due to the nature of this algorithm, it is particularly sensitive to outliers which can have an outsize influence on the classification (Yıldırım, 2020).

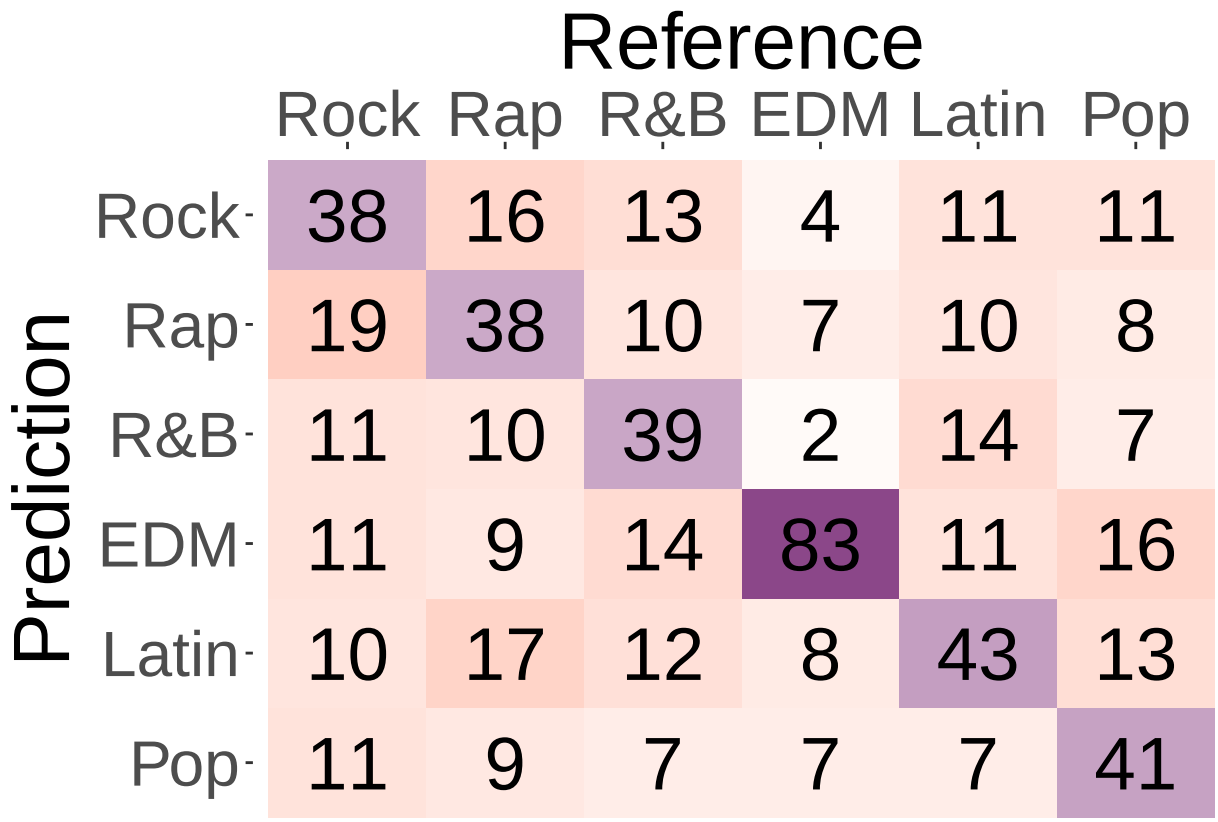

Figure 16: Confusion Matrix of the k-Nearest Neighbours (kNN) Model

Support Vector Machines (SVM) Algorithm

The *Support Vector Machines* algorithm classifies data by determining the hyperplane, or line, which best divides the data set into classes in three-dimensional space. As points get further from the hyperplane, the confidence in their classification increases. *Support Vectors* refers to the data points nearest to the hyperplane which if removed would change the location of the hyperplane. They are considered critical elements of the data set (Bambrick, 2016).

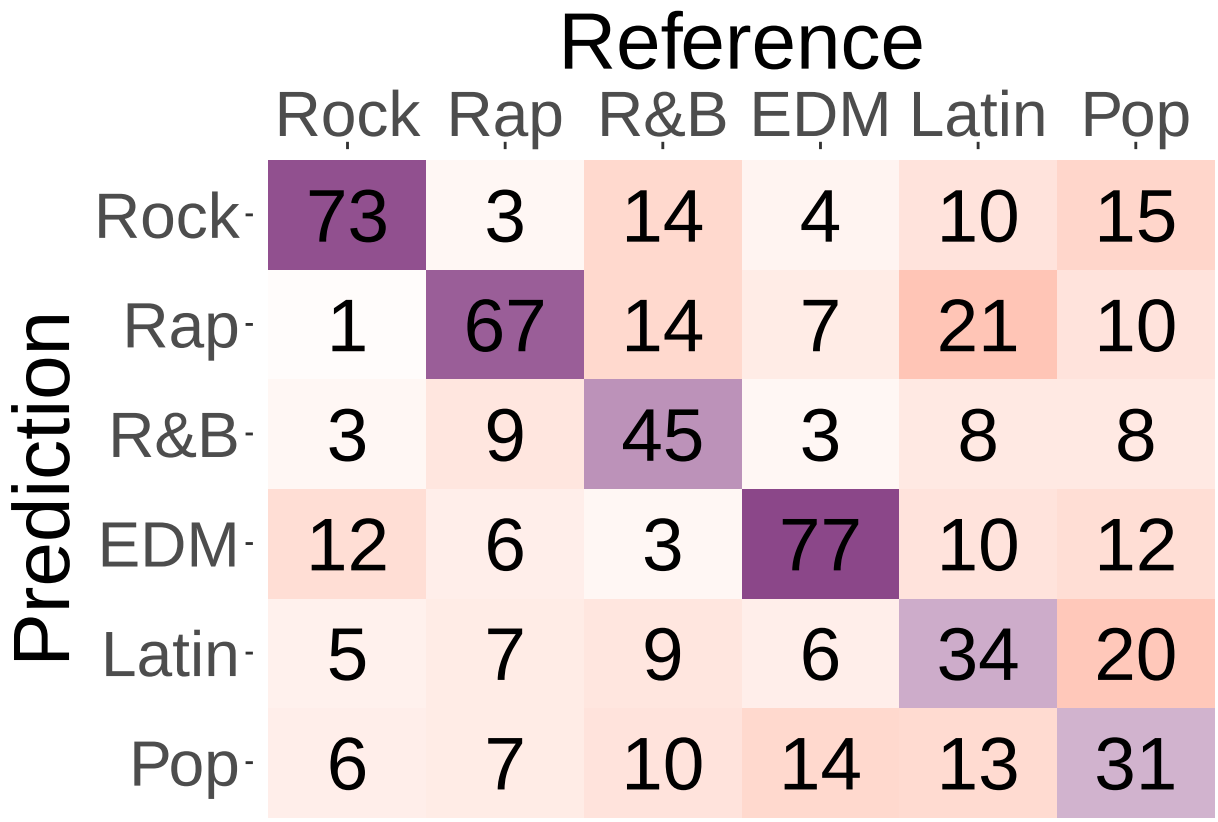

Figure 17: Confusion Matrix of the Support Vectors Machine (SVM) Model

Analysis of Confusion Matrices

Three predictive modelling algorithms (or supervised machine learning models) were used in order to compare their results and accuracy in identifying the *Genre* to which a song belongs. The algorithms were *Linear Discriminant Analysis (LDA)*, the *k-Nearest Neighbors (kNN)* algorithm, and the *Support Vector Machine (SVM)* algorithm. These identified the *Genre* of the songs in the validation data set with differing levels of accuracy. Their results are presented in the confusion matrices above (**Figures 15 to 17**).

The matrices display the results of each model running 600 samples from the test data set, derived from the 3000-example subset. It is important to note that while a high number of correct categorizations denotes a *Genre* is being more successfully classified than another, this may also be because of a *Genre* having more variables present in the data set.

From the confusion matrices, a better understanding can be gained of how the models perform, and how their accuracy may change between different *Genres*. The *Genres* are listed along the top and left. The squares in the matrix compare the number of songs from a *Genre*, and the *Genre* to which each model assigned the songs. The centre diagonal shows how many songs were correctly categorised in each *Genre* (i.e. the number of true positives) - the darker the colour, the greater the number of successful classifications.

The squares outside of this diagonal show songs which were incorrectly categorised. Values in a column (i.e. above or below the centre diagonal) are the false positives (FP) in the given *Genre*. These are songs which were assigned to a *Genre* but which were not part of that *Genre*, e.g. a Pop song incorrectly classified as Rock. Values in a row (i.e. left or right of the centre diagonal) are the false negatives (FN) in the given

Genre. These are songs not correctly assigned to the right *Genre*, but which belong to the *Genre*. For instance, a Rock song mistakenly assigned to Pop instead.

At a glance, a number of findings can be seen. For example, in the *LDA* model (**Figure 15**), the highest number of correct categorisations were for Rock, EDM, and Rap music, as they are the darkest purple. Very few EDM songs were identified as R&B and very few R&B songs were identified as EDM. This may indicate these *Genres* are musically distinct from each other. In the *kNN* model (**Figure 16**), EDM experienced the highest number of correct categorisations at 85. The same pattern between R&B and EDM is also seen again. But there are clear differences between the two models. The *SVM* model (**Figure 17**) also shows differences, with all *Genres* having a correct categorization of 40 or above - *SVM* was the best performing model overall. The three most successfully classified *Genres* in the *SVM* model are the same as the three most successfully classified *Genres* in the *LDA* model. This is an interesting observation which could indicate that those *Genres* are generally more distinct or are easier for the algorithms to identify.

These observations explore only a portion of the confusion matrices, but they show the types of insights which can be gained from performing and analysing these predictive modelling algorithms. They also only explore a small portion of the overall data set. If further analysis was desired, this subset could help indicate the types of modelling techniques to use in order to achieve research goals.

Evaluation of Results

Summary of Classification Accuracy

Because 10-fold cross validation was assigned, the validation data was resampled ten times. Therefore, in order to find the *Mean Accuracy* of each of the three models, some additional tidying and calculation was required. After calling the list of resampled results and assigning it to a variable named `results`, the `results` were transformed into a data frame called `results_df`.

This data frame was not in a tidy format. Our data was pivoted to increase the number of rows and reduce the number of columns. The *LDA*, *kNN*, and *SVM* column names became values in a new column called *Model*; the accuracy values under each of the old columns were gathered into their respective rows in the new *Accuracy* column; and the data was sorted by the number of resamples. Next, the data was grouped by model, and *Mean Accuracy* was calculated for each model, and assigned to a variable named `results_tidy`. Finally, `results_tidy` as used to plot a bar graph ordered by decreasing *Mean Accuracy* (**Figure 18**).

Overall model accuracy was poor: the best performing model was *Support Vectors Machine (SVM)* with a *Mean Accuracy* of 47.7%, followed by *Linear Discriminant Analysis (LDA)* with a *Mean Accuracy* of 44.6%, and lastly *k-Nearest Neighbours (kNN)* with a *Mean Accuracy* of 33.2%.

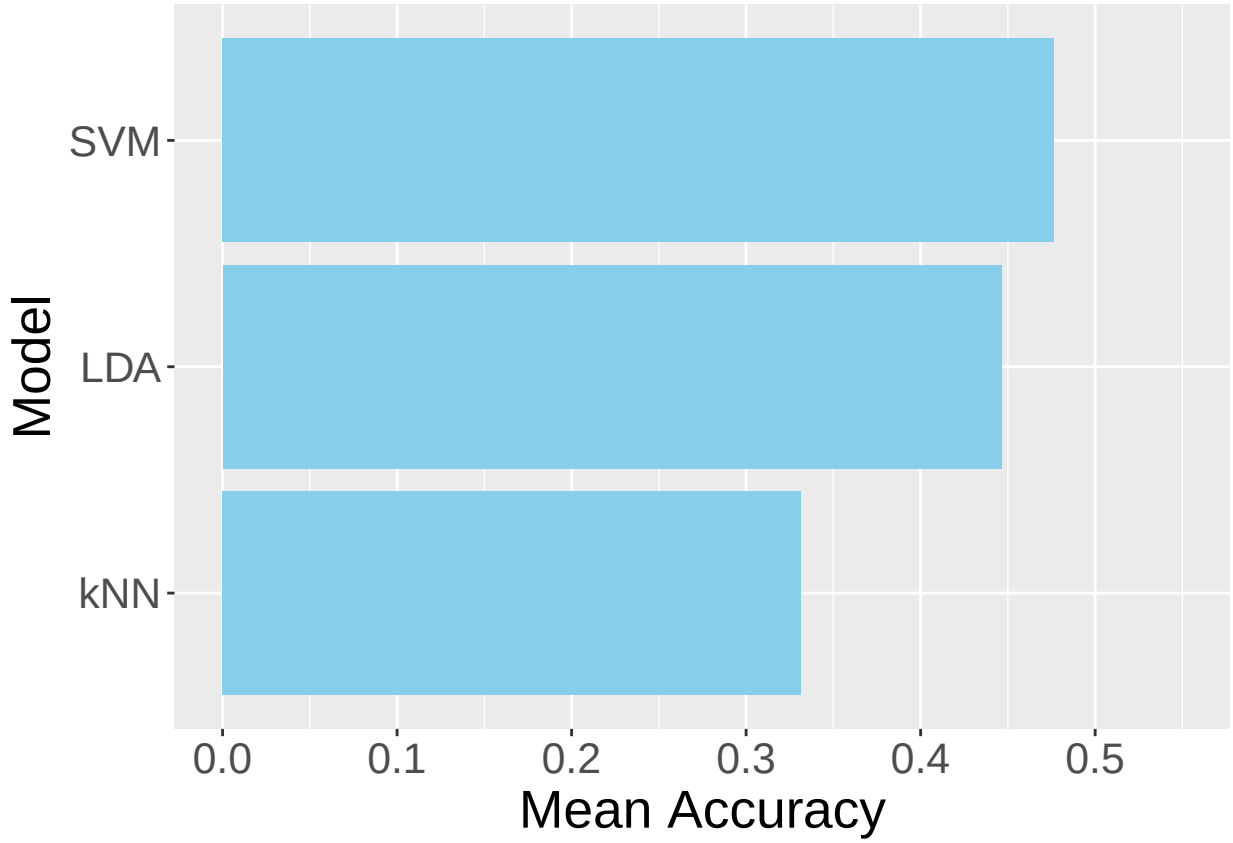

Figure 18: Bar Chart of Mean Model Accuracy (%) of SVM, LDA, and *k*NN Models

Model-Specific Sensitivity and Specificity by Genre

Sensitivity and *Specificity* are both descriptors of model accuracy. *Sensitivity* describes the True Positive Rate (TPR), the ability of a model to correctly classify a song into a *Genre* (for example, correctly classifying a Rock song as Rock). *Specificity* describes the True Negative Rate (TNR), the ability of a model to correctly identify songs not belonging to a *Genre* (for example, correctly classifying a Pop song as not Rock). *Sensitivity* and *Specificity* are inversely related - as one increases, the other decreases. High values in both descriptors of model accuracy are desirable, but the choice between high *Sensitivity* or high *Specificity* depends on the nature of the test performed (Parikh et al., 2008). If high rates of successful classification are needed (such as in this exercise), then high *Sensitivity* is required (Parikh et al., 2008). If low rates of misclassification are desired, then high *Specificity* is required (Parikh et al., 2008).

Once again, a fair bit of tidying was required to plot *Sensitivity* and *Specificity* for each model. *Sensitivity* and *Specificity* was accessed by subsetting the `byClass` objects of the confusion matrices and assigning them to separate variables. Next, these variables were transformed into tibbles, and new columns for *Genre* were added to each. The *Genre* values were then factorised to specify an order to the factors. This was performed at this stage to organise the eventual bar charts by decreasing *Sensitivity*.

As before, the data was pivoted to increase the number of rows and reduce the number of columns. *Sensitivity* and *Specificity* became new values in a column called *Descriptor*, and the values in the former *Sensitivity* and *Specificity* columns were moved to a new column called *Values*. The *Values* was multiplied by 100 to determine the percentage values. Finally, *Sensitivity* and *Specificity* was charted for each model type - the bar graphs were grouped by *Genre* and arranged into one plot (**Figure 19**).

Generally, *Sensitivity* was much lower than *Specificity* for all three models. If this exercise was based around successfully avoiding the misclassification of songs into the wrong *Genre*, then this exercise would be a resounding success, since *Specificity* does not dip below 85% for all *Genres* in all models. But the goal was to examine how well each model fared at correctly classifying each song into a *Genre*. Due to the lower values for *Sensitivity*, the models did not meet this goal.

However, some *Genres* displayed higher *Sensitivity* than others. In *SVM* and *LDA* models (the two best performing models), Rock, EDM and Rap displayed the highest *Sensitivity*, whereas R&B, Latin, and Pop displayed the lowest *Sensitivity*. This means the *SVM* and *LDA* models were able to successfully classify more Rock, EDM and Rap songs than R&B, Latin, and Pop songs.

In the *SVM* model (**Figure 18, Top**), *Sensitivity* for Rock, EDM and Rap ranged from 67% to 73%, whereas *Sensitivity* for the other three *Genres* was below 50%. Additionally, Pop was the worst performing *Genre*.

In the *LDA* model (**Figure 18, Middle**), *Sensitivity* was lower: for Rock, EDM, and Rap, *Sensitivity* ranged from 58% to 64%, whereas for the other three *Genres*, *Sensitivity* was below 45%. Latin was the worst performing *Genre* in the *LDA* model.

In the *kNN* model (**Figure 18, Bottom**), EDM displayed by far the highest *Sensitivity* seen in all three models with a value of 74.8%. However, *Sensitivity* for all other *Genres* was below 45%. As seen previously, the *kNN* model was able to correctly classify more EDM songs than any other *Genre*. Interestingly, Rock and Rap showed the worst *Sensitivity* in the *kNN* model, an inverse of the pattern seen in the *LDA* and *SVM* models, followed by R&B, Pop, and Latin.

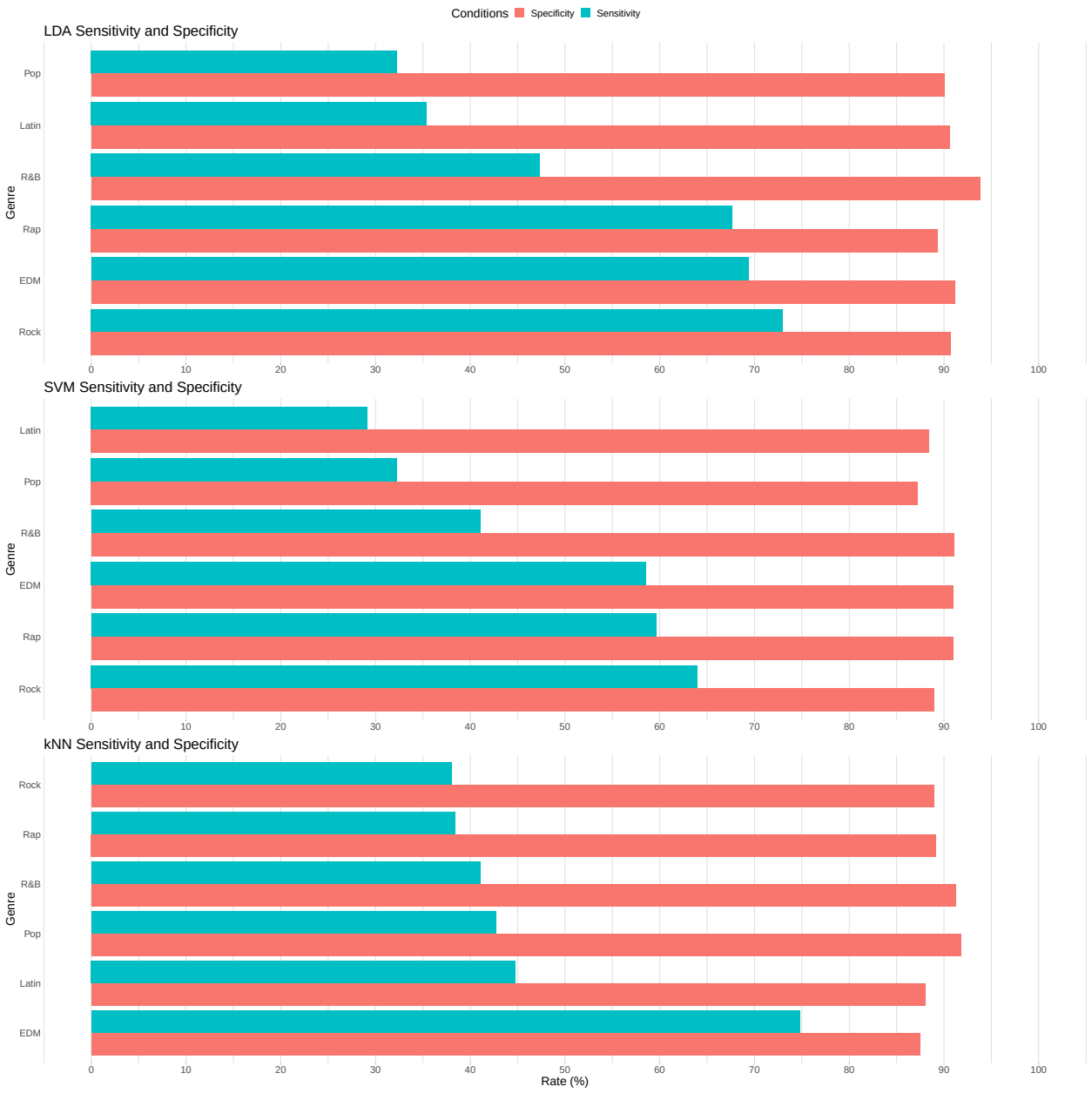

Figure 19: Bar Charts Displaying Sensitivity and Specificity (%) of the SVM (Top), LDA (Middle), and *k*NN (Bottom) Models

Variable Importance

Variable importance refers to how often a model relies on a variable (or variables) to make classification predictions. The more a model uses a variable to make predictions, the more important that variable is to the efficacy of a model. However, with the `varimp()` function in the `caret` package, variable importance can only be directly computed for certain model types (Kuhn, 2019). Variable importance cannot be calculated for *LDA*, *SVM*, and *kNN* models because by definition (i.e. based on how they function), they do not assign importance to variables when they run (Kuhn, 2019). When variable importance for each of the models was first calculated and plotted, the resulting variable importance plots were identical. Because the `varImp`

function does not support *LDA*, *SVM*, or *kNN* models, it instead calculated a generalised 'model-free' variable importance (Kuhn, 2019).

An alternate approach to calculating variable importance, albeit indirectly, is to use the variable drop-out method. This method requires running the model several times, excluding a different *Predictor* each attempt. By comparing the differences in prediction scores when certain variables are dropped, the effects of each variable on classification can be calculated. This is a long-winded approach. There is a way to automate this process, but calculating variable drop-outs was beyond the scope of this project. The generalised variable importance plot was used instead.

In order to plot the *Variable Importance* values, a fair bit of tidying was required. The results of the `varImp()` function were subsetted and stored it as the tibble `imp1`. Next, the data was pivoted to increase the number of rows and reduce the number of columns: the various *Genres* in the column headings became new values in a column called *Genre*, and the *Importance* scores in the *Genre* columns were moved to a new column called *Importance*. Next R.B was renamed to R&B.

The next series of steps was performed to find which *Predictors* were the most and least important for classifying a song into a *Genre*. This required separating the *Predictors* into new tibbles, and finding the *Genre* with the highest and lowest *Importance* scores per *Predictor*. The highest values were appended to a tibble named `maxImp`, and the lowest values were appended to a tibble named `minImp`. Both were plotted together as lollipop charts. The *Predictors* were also factorised to organise each chart from highest to lowest *Importance* (**Figure 20**).

Some patterns emerged in plotting the highest and lowest *Importance* scores per *Predictor*. *Duration* played no role in the classification of Pop songs, but it played the greatest role in the classification of Latin songs. *Speechiness* was incredibly important in classifying EDM music, but it was least important in classifying Rock music. Generally, the 'floor' and the 'ceiling' of *Speechiness* in terms of its *Importance* was much higher than the other *Predictors*.

Interestingly, the *Genres* which appear on the 'Highest Variable Importance' (**Figure 20, Bottom**) chart were the *Genres* which were more successfully classified. The two exceptions were R&B and Rap songs. Despite many *Predictors* being greatly important in the classification of R&B songs, their overall classification accuracy was quite low. On the other hand, no *Predictor* was highly important for the classification of Rap music, but their overall classification accuracy was relatively high (though low overall), which likely means that many *Predictors* were of average *Importance* in classifying Rap music.

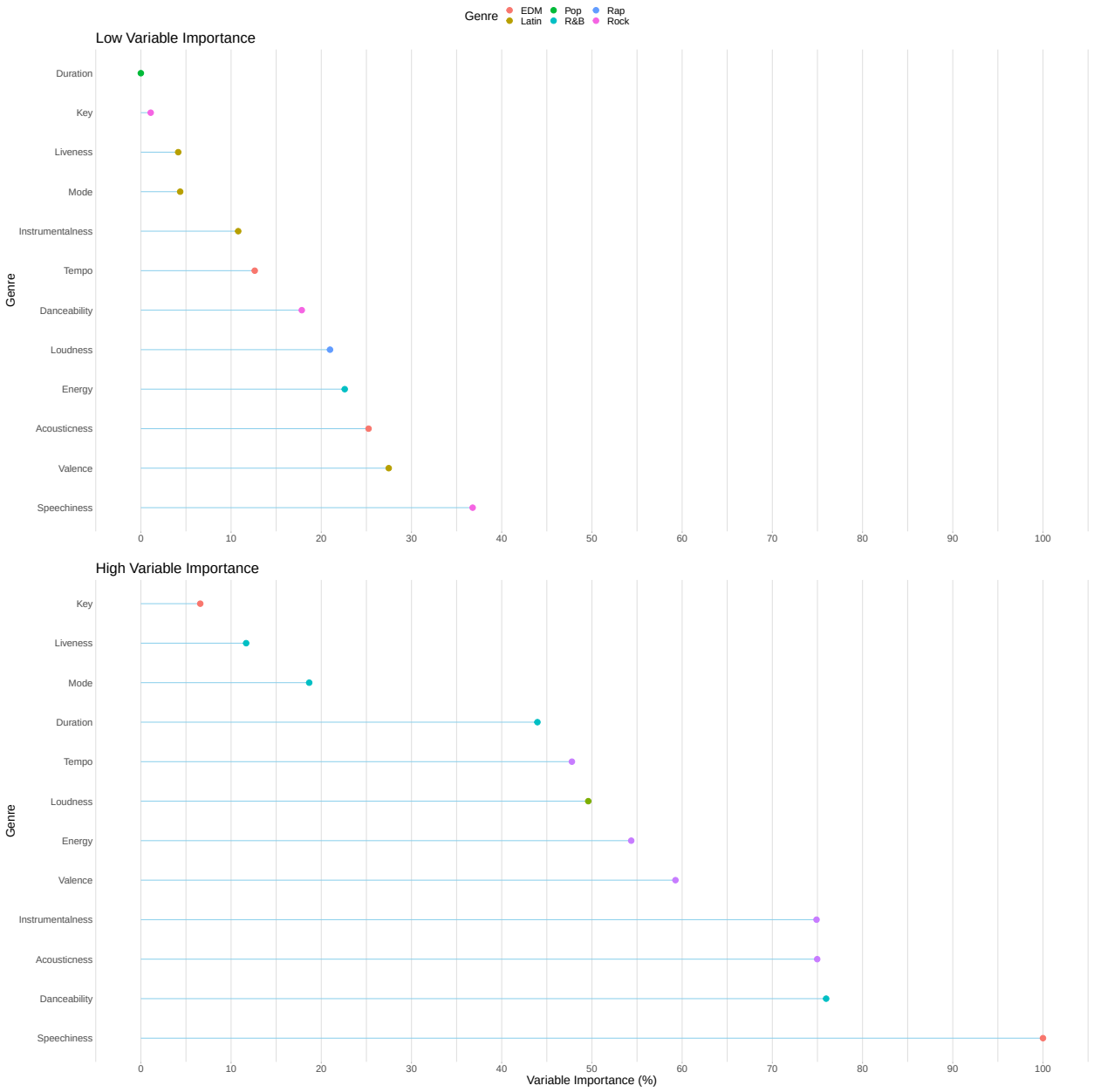

Figure 20: Bar Charts Displaying the Generalised Least Important (Top) and Most Important (Bottom) Predictors for Genre Classification

Next, *Mean Importance* per *Predictor* was calculated and plotted across all *Genres* as a bar chart. The factor levels of the *Predictors* were modified to organise the bar chart in order of increasing *Mean Importance*. **Figure 21** shows the least and most important *Predictors* overall for *Genre* classification - *Key*, *Liveness*, and *Mode* were the least important by far, likely because these are broad, general audio features common to all *Genres*. On the other hand, *Speechiness* and *Danceability* were by far the most important, likely because more separation was seen between *Genres* using these *Predictors*.

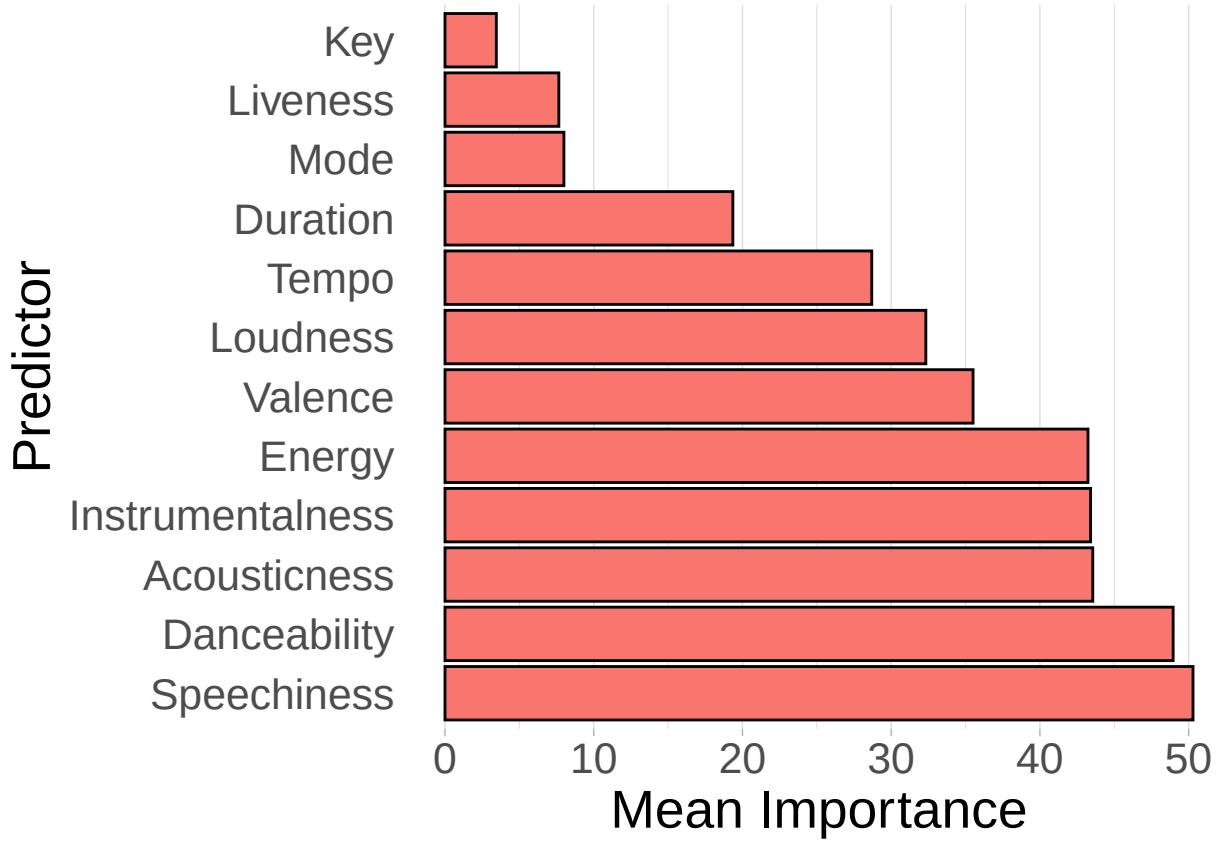

Figure 21: Bar Charts Displaying Generalised Mean Importance per Predictor

However, as seen in **Figure 20, Bottom**, *Speechiness* was 100% important for classifying EDM music, but its mean was just above 50%, suggesting low values in the *Importance* of *Speechiness* for the other *Genres*. Therefore, the breakdown of *Variable Importance* by *Predictors* and *Genre* was visualised, using the `facet_wrap()` argument to separate the plots by *Predictor*. The plots were organised by *Mean Importance*: the *Predictors* with the highest *Importance* were at the top, the *Predictors* with moderate *Importance* were in the middle, and the *Predictors* with the lowest *Importance* were at the bottom (**Figure 22**).

The overall trend of the results seen here matched the trend observed in the **Figure 21**. However, several interesting trends were observed in relative *Importance* for each *Predictor*. *Duration* played almost no role in classifying EDM, Pop, and Rap music, but it played a much greater role in classifying Latin, R&B and Rock music, likely because the latter three *Genres* have songs of greater length than the former three *Genres*. *Instrumentalness* played a lesser role in classifying EDM and Latin music compared to the other *Genres*, but it played a greater role in classifying Rap music, likely because Rap music is more likely to score low on *Instrumentalness*. Surprisingly, *Tempo* did not play a strong role in classifying EDM music, despite the highly modal distribution in the tempo of EDM music in this data set. *Speechiness* was 100% important in classifying EDM music, while being less than 45% important in classifying all other *Genres*, likely because EDM music is less likely to contain 'spoken-word' or 'rapping' performances (at least in this data set).

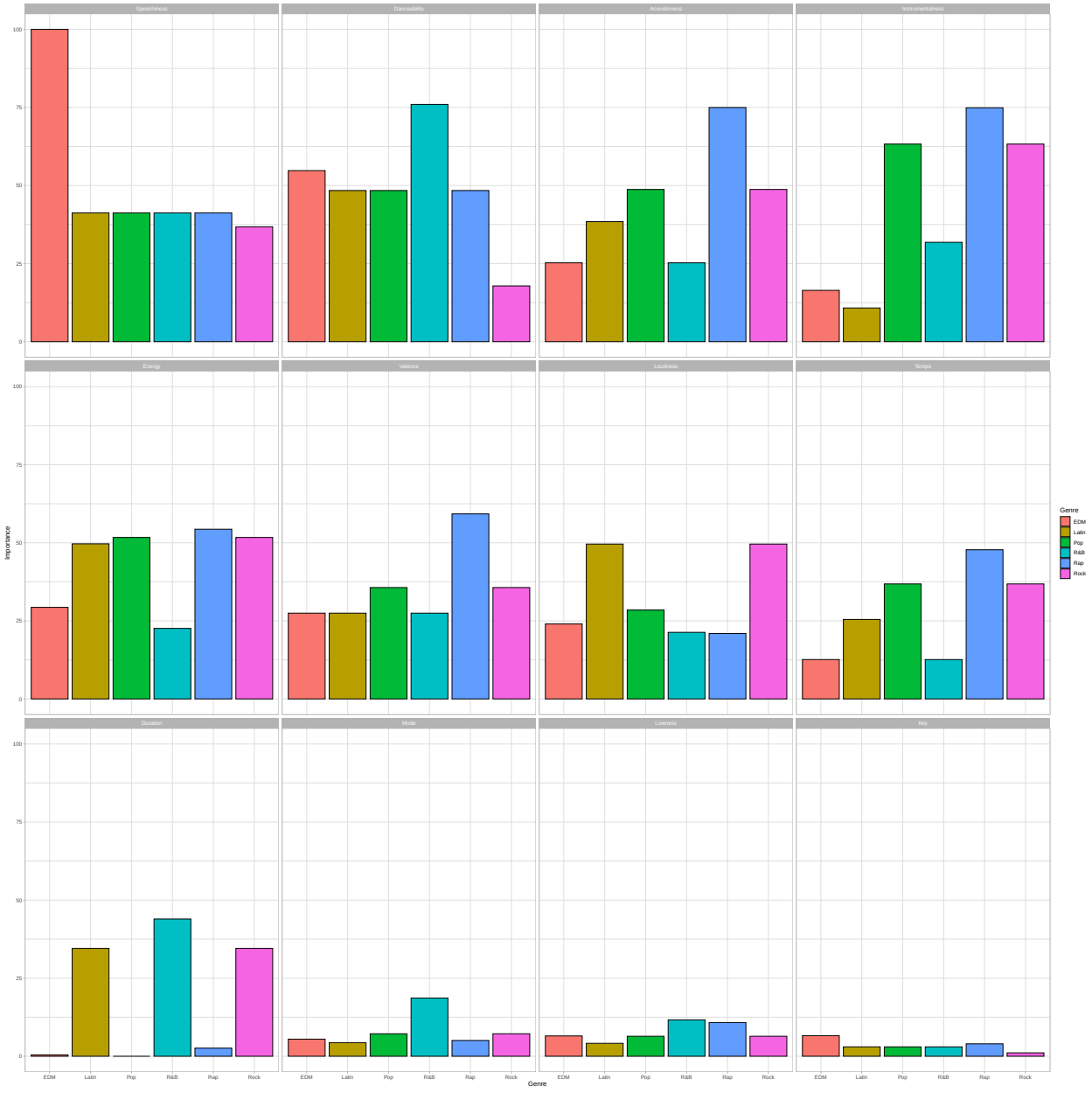

Figure 22: Bar Charts Displaying Generalised Importance for Each Genre per Predictor

Discussion and Conclusion

The various *Genres* can be described and defined by audio features (i.e. *Predictors*) as follows: EDM is considered high in *Energy*; *Rap* is high in *Speechiness*; *Rock* has low *Danceability* and longer song *Duration*; and Latin is moderate to high in *Danceability*. R&B and Pop are much harder to define, but further analysis may help in defining them.

This exercise in successfully classifying music into *Genres* failed. Even though accuracy appeared high, overall accuracy was poor. However, misclassification was not as high as expected. Certain *Genres* - Rock, EDM, and Rap - were more successfully classified into the correct *Genres*, whereas R&B, Latin, and Pop were not. It is not unsurprising that classification failed for R&B and Pop because they were not easily defined,

whereas Rap was more successfully classified. However, Latin music should have been more successfully classified.

The first possibility relates to the *Genres* selected. There is a high degree of similarity between R&B, Latin, Pop, and Rap (though Rap was more separable than the other *Genres*). Because of these similarities, the models find it difficult to correctly classify music from these *Genres*. There are stylistic similarities between these *Genres*, due to their shared histories, common musical influences, and their influences on each other (Evolution of Latin Hip Hop – Recording Arts Canada, n.d.). As previously mentioned, this is easily discernible in the *SubGenres* of each *Genre*. For instance, Latin Hip-Hop and Hip-Hop are sub-genres of Latin and Rap, but Latin Hip-Hop could easily be placed in Rap (Evolution of Latin Hip Hop – Recording Arts Canada, n.d.). If these highly similar *Genres* were replaced by less similar *Genres*, greater separability could be visible, and model performance could be improved. This would require using a different data set or creating a novel data set using the `spotifyr` package. Alternate options for highly separable *Genres* may include Country, Classical, Jazz, Death Metal, or Nursery Rhymes.

A related factor is that there may be too many target variables - the more classes a classification model has to use for categorisation, the greater the chance of reduced model performance. *SubGenres* were not used precisely for this reason, but it is possible that using six *Genres* was also too many. However, while reducing the number of target variables makes classification easier, it reduces model applicability as well. Categories are often not binary.

Another related possibility is an issue with the methods employed. The models categorised the songs in the data set into a single *Genre*. Practically speaking, songs can have elements of, and indeed, be of many different *Genres*. This is the reason why *SubGenres* exist. Classifying them into one *Genre* may be overly reductive. A better approach may be to classify songs into the three most likely *Genres*. If such a method were used, many songs will probably be classified as a mixture of R&B, Latin, Rap, or Pop, and may reduce errors.

Finally, related to the methodology employed, the 'curse of dimensionality' may also play a role (Yiu, 2019). If too many *Predictor* variables are included in a machine learning model, model efficacy may be lowered. In fact, *k-Nearest Neighbour* models are particularly susceptible to this effect, which is why they are commonly employed in data sets featuring ten or less *Predictors* (Yıldırım, 2020). This exercise used twelve *Predictors*. The *kNN* model relies on the Eucliean distance between points and known groups to classify unknown points (Yıldırım, 2020). With an increase in the number of *Predictors* (i.e. dimensions), there is an exponential increase in Euclidean distance between points and groups due to the increase in dimensions, resulting in lowered model performance (Yıldırım, 2020).

There are two primary methods of dealing with the curse of dimensionality. The first is to use a much larger data set by increasing the number of observations. To avoid the curse, the size of the data set needs to increase exponentially with the number of *Predictors* used (Yiu, 2019). The exact number of observations required for optimal performance is unknown, but it is likely well above the 32,833 observations in this data set. Additionally, the data set had to be downsampled because the *SVM* model took well over an hour to run (and it never completed). Unfortunately, in downsampling the data set, the data set was cursed: the models were able to complete, but their overall efficacy was reduced.

The second way of dealing with the curse of dimensionality is to reduce the number of *Predictors* used, i.e. dimension reduction, especially in the case of *kNN* models (Yiu, 2019). From the plots of the decision tree and *Variable Importance* (**Figures 20 to 22**), *Key*, *Mode*, and *Liveness* were found to not be important in predicting *Genre*. Removing these *Predictors* may also improve model efficacy. However, using a combination of a larger data set and a smaller number of *Predictors* is likely needed if this exercise were to be run again.

Overall, this was a useful introduction to classification. While high accuracy was not expected, there were several areas where clear improvements could be made. If provided the opportunity to repeat this exercise, these improvements could be applied, hopefully increasing the efficacy of these models.

List of References

Bambrick, N. (2016). Support Vector Machines: A Simple Explanation. Kdnuggets.com.

https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

Behrens, J. T. (1997). Principles and procedures of exploratory data analysis. Psychological Methods, 2(2), 131–160.

https://doi.org/10.1037/1082-989x.2.2.131

Curtis, M. E., & Bharucha, J. J. (2010). The minor third communicates sadness in speech, mirroring its use in music. Emotion, 10(3), 335–348.

https://doi.org/10.1037/a0017928

Dash, S. K. (August 5, 2022). A brief introduction to linear discriminant analysis. AnalyticsVidhya.com.

https://www.analyticsvidhya.com/blog/2021/08/a-brief-introduction-to-linear-discriminant-analysis/

Davies, S. (2012). On Defining Music. Monist, 95(4), 535–555.

https://doi.org/10.5840/monist201295427

Dictionary.com. (2022). Music definition & meaning. Dictionary.com. Retrieved December 1, 2022, from https://www.dictionary.com/browse/music

Evolution of Latin Hip Hop – Recording Arts Canada. (n.d.). Recordingarts.com.

https://recordingarts.com/record/evolution-of-hip-hop/latin/

Glossary of Music Terms: Streaming – Spotify for Artists. (n.d.). Artists.spotify.com. Retrieved December 15, 2022, from

https://artists.spotify.com/en/blog/glossary-of-music-terms-streaming

Harmon, J., & Grantham, N. (2022, March 8). DataSets. GitHub.

https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-01-21/readme.md

Kuhn, M. (2019). 15 Variable Importance | The caret Package. In topepo.github.io.

https://topepo.github.io/caret/variable-importance.html

Malloch, S., & Trevarthen, C. (2018). The Human Nature of Music. Frontiers in Psychology, 9.

https://doi.org/10.3389/fpsyg.2018.01680

Parikh, R., Mathai, A., Parikh, S., Chandra Sekhar, G., & Thomas, R. (2008). Understanding and using sensitivity, specificity and predictive values. Indian Journal of Ophthalmology, 56(1), 45.

https://doi.org/10.4103/0301-4738.37595

Pavlik, K. (2019, December 20). Classifying genres in R using Spotify data. Kaylin Pavlik.

https://www.kaylinpavlik.com/classifying-songs-genres/

Pavlik, K., Harmon, J., & Grantham, N. (2019). Githubusercontent.com.

https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-21/spotify_songs.csv

Spotify. (2022). About Spotify. Spotify; Spotify Newsroom.

https://newsroom.spotify.com/company-info/

Thompson, C. (n.d.). R Wrapper for the “Spotify” Web API. Www.rcharlie.com.

https://www.rcharlie.com/spotifyr/

Trehub, S. E., Becker, J., & Morley, I. (2015). Cross-cultural perspectives on music and musicality. Philosophical transactions of the Royal Society of London. Series B, Biological sciences, 370(1664), 20140096.

https://doi.org/10.1098/rstb.2014.0096

Web API Reference | Spotify for Developers. (n.d.). Developer.spotify.com.

https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

Xiaozhou, Y. (2020, October 7). Linear Discriminant Analysis, Explained. Medium.

https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b

Yıldırım, S. (2020, March 1). K-Nearest Neighbors (kNN) — Explained. Medium.

https://towardsdatascience.com/k-nearest-neighbors-knn-explained-cbc31849a7e3

Yiu, T. (2019, July 20). The Curse of Dimensionality. Medium; Towards Data Science.

https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e