CS-GY 6923 Machine Learning
Fall 2022

Professor: Dr. Raman Kannan

Exploratory Data Analysis Report
Music Genre Classification

Author: Atmaja Raman
Net ID: ar6871

Table of Contents

Introduction

Exploratory data analysis is a way to investigate data using visualizations. It is helps find trends, patterns and find anomalies in data and check assumptions with the help of statistics and graphs. Data preprocessing and data cleaning are critical components of exploratory data analysis. The data can contain redundancy such as irregular data, missing values or outliers which can cause inaccuracies in model during training. Data preprocessing is an important step.

During the exploratory phase, we formulate questions about data, and try to determine answers for them using visualizations and statistical methods. Several new questions might arise while looking into the obtained results and we try to find explore further to answer these questions.

Data Questions

By performing data analysis, we hope to answer the following data questions.

- What are the dependent and independent variables?
- What are the datatypes of features?
- Does the target variable have class imbalance?
- Are there any missing values?
- Are there faulty erroneous values/ anomalies in data?
- What is the distribution of features?
- What is the spread of data?
- Are the features skewed?
- Are there outliers in the data? Do we need to exclude them?
- Is there a need to perform data imputation? What method to use?
- What are some general insights from the dataset?
- What is the relationship between features and target variable?
- Is there correlation among features and among target classes?
- Is there a need to scale/ standardize/normalize the dataset?
- How to encode categorical features in the data?
- What are the important features in data?
- Do we need all the dimensions in the data?

Overview of the dataset

The Music Genre Prediction dataset used for the exploratory data analysis task was obtained from Kaggle. The dataset consists of songs and their corresponding features such as artist, popularity, genre, etc.

Dataset link: https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre

Number of attributes: **18**
Number of independent attributes: **17**
Number of dependent attributes: **1**
Target variable: **music_genre**
Number of classes in target: **10**
Number of instances: **50005**

Description of Attributes

The table below gives the features and their description to better understand the data attributes .

Feature	Description
instance_id	Unique ID for each song
artist_name	Artist of the song
track_name	Name of the song
popularity	Popularity of the song from 0 to 100. (lower value, higher popularity)
acousticness	Acoustic is a property of sound and the value describes how acoustic the song is. (1-highly acoustic, 0-less acoustic)
danceability	The value describes how suitable a song is for dancing. (1-highly danceable, 0-less danceable)
duration_ms	Duration of the song in milliseconds
energy	Energy is based on the tempo, intensity, loudness of song. (1-high energy, 0-low energy)
instrumentalness	It shows the amount of vocals in a song. (1-less vocal, 0-more vocal)
key	The tonal center of a song.
liveness	It gives the likelihood from 0 to 1 that the song was recorded with a live

	audience. (lower value, less likely to be live)
loudness	The overall loudness of the song track.
mode	Mode indicates modality (major or minor) of a track.
speechiness	The value indicates the amount of words in a song. (0.66>very speechy, 0.33 to 0.66-music & speech, 0.33<no speech)
tempo	The overall pace of the song measured by beats per minute.
obtained_date	Date the song was collected.
valence	A measure of positivity in a track from 0 to 1.(Higher valence, more positive)
music_genre	The genre of the song. Genres are alternative, anime, blues, classical, country, electronic, hiphop, jazz, rap, rock.

Loading the dataset

The dataset was loaded using read.csv command and the top 6 rows were displayed using head().

```
[13] data = read.csv("/content/music_genre.csv")
```

```
head(data)
```

A data.frame: 6 × 18

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo
	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<dbl>	<chr>
1	32894	Röyksopp	Röyksopp's Night Out	27	0.00468	0.652	-1	0.941	0.79200	A#	0.115	-5.201	Minor	0.0748	100.889
2	46652	Thievery Corporation	The Shining Path	31	0.01270	0.622	218293	0.890	0.95000	D	0.124	-7.043	Minor	0.0300	115.00200000000001
3	30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	0.01180	G#	0.534	-4.617	Major	0.0345	127.994
4	62177	Dubloadz	Nitro	34	0.02540	0.774	166875	0.700	0.00253	C#	0.157	-4.498	Major	0.2390	128.014
5	24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	0.90900	F#	0.157	-6.266	Major	0.0413	145.036
6	89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	0.85400	D	0.216	-10.517	Minor	0.0412	?

```
[15] dim(data)
```

50005 · 18

The feature name and datatype of each feature in the dataset is displayed using str() . The datatype information is summarised below in the table.

```
[16] colnames(data)
```

'instance_id' · 'artist_name' · 'track_name' · 'popularity' · 'acousticness' · 'danceability' · 'duration_ms' · 'energy' · 'instrumentalness' · 'key' · 'liveness' · 'loudness' · 'mode' · 'speechiness' · 'tempo' · 'obtained_date' · 'valence' · 'music_genre'

```
str(data)
```

```
'data.frame':   50005 obs. of  18 variables:
 $ instance_id     : num  32894 46652 30097 62177 24907 ...
 $ artist_name     : chr  "Röyksopp" "Thievery Corporation" "Dillon Francis" "Dubloadz" ...
 $ track_name      : chr  "Röyksopp's Night Out" "The Shining Path" "Hurricane" "Nitro" ...
 $ popularity      : num  27 31 28 34 32 47 46 43 39 22 ...
 $ acousticness    : num  0.00468 0.0127 0.00306 0.0254 0.00465 ...
 $ danceability    : num  0.652 0.622 0.62 0.774 0.638 0.755 0.572 0.809 0.509 0.578 ...
 $ duration_ms     : num  -1 218293 215613 166875 222369 ...
 $ energy          : num  0.941 0.89 0.755 0.7 0.587 0.731 0.803 0.706 0.921 0.731 ...
 $ instrumentalness: num  7.92e-01 9.50e-01 1.18e-02 2.53e-03 9.09e-01 8.54e-01 7.74e-06 9.03e-01 2.76e-04 1.12e-02 ...
 $ key             : chr  "A#" "D" "G#" "C#" ...
 $ liveness        : num  0.115 0.124 0.534 0.157 0.157 0.216 0.106 0.0635 0.178 0.111 ...
 $ loudness        : num  -5.2 -7.04 -4.62 -4.5 -6.27 ...
 $ mode            : chr  "Minor" "Minor" "Major" "Major" ...
 $ speechiness     : num  0.0748 0.03 0.0345 0.239 0.0413 0.0412 0.351 0.0484 0.268 0.173 ...
 $ tempo           : chr  "100.889" "115.00200000000001" "127.994" "128.014" ...
 $ obtained_date   : chr  "4-Apr" "4-Apr" "4-Apr" "4-Apr" ...
 $ valence         : num  0.759 0.531 0.333 0.27 0.323 0.614 0.23 0.761 0.273 0.203 ...
 $ music_genre     : chr  "Electronic" "Electronic" "Electronic" "Electronic" ...
```

Datatype of Attributes

Feature	Datatype
instance_id	`<dbl> - numeric double`
artist_name	`<chr> - character (nominal)`
track_name	`<chr> - character (nominal)`
popularity	`<dbl> - numeric double`
acousticness	`<dbl> - numeric double`
danceability	`<dbl> - numeric double`
duration_ms	`<dbl> - numeric double`
energy	`<dbl> - numeric double`
instrumentalness	`<dbl> - numeric double`
key	`<chr> - character (ordinal)`
liveness	`<dbl> - numeric double`
loudness	`<dbl> - numeric double`
mode	`<chr> - character (ordinal)`
speechiness	`<dbl> - numeric double`
tempo	`<chr> - character (nominal)`
obtained_date	`<chr> - character (ordinal)`
valence	`<dbl> - numeric double`
music_genre	`<chr> - character (ordinal)`

The summary statistics of each feature is obtained by using summary() function. From the output we know the following:

- Some features have 5 missing or null values, probably from the same set of rows.
- Duration_ms feature has -1 as minimum and a very high maximum pointing to outliers.

- While most columns have values in range 0 to 1 some columns like duration_ms, loudness have values in a different range which indicates the need to scale.
- Tempo (beats per minute) has character datatype which needs to be converted to numeric.
- Date is also of character type and not date type.

```
summary(data)
```

```
 instance_id    artist_name         track_name          popularity
Min.   :20002   Length:50005       Length:50005       Min.   : 0.00
1st Qu.:37974   Class :character   Class :character   1st Qu.:34.00
Median :55914   Mode  :character   Mode  :character   Median :45.00
Mean   :55888                                         Mean   :44.22
3rd Qu.:73863                                         3rd Qu.:56.00
Max.   :91759                                         Max.   :99.00
NA's   :5                                             NA's   :5
 acousticness     danceability     duration_ms          energy
Min.   :0.0000   Min.   :0.0596   Min.   :     -1   Min.   :0.000792
1st Qu.:0.0200   1st Qu.:0.4420   1st Qu.: 174800   1st Qu.:0.433000
Median :0.1440   Median :0.5680   Median : 219281   Median :0.643000
Mean   :0.3064   Mean   :0.5582   Mean   : 221253   Mean   :0.599755
3rd Qu.:0.5520   3rd Qu.:0.6870   3rd Qu.: 268612   3rd Qu.:0.815000
Max.   :0.9960   Max.   :0.9860   Max.   :4830606   Max.   :0.999000
NA's   :5        NA's   :5        NA's   :5         NA's   :5
instrumentalness      key              liveness          loudness
Min.   :0.000000   Length:50005       Min.   :0.00967   Min.   :-47.046
1st Qu.:0.000000   Class :character   1st Qu.:0.09690   1st Qu.:-10.860
Median :0.000158   Mode  :character   Median :0.12600   Median : -7.277
Mean   :0.181601                      Mean   :0.19390   Mean   : -9.134
3rd Qu.:0.155000                      3rd Qu.:0.24400   3rd Qu.: -5.173
Max.   :0.996000                      Max.   :1.00000   Max.   :  3.744
NA's   :5                             NA's   :5         NA's   :5
    mode            speechiness         tempo           obtained_date
Length:50005       Min.   :0.02230   Length:50005       Length:50005
Class :character   1st Qu.:0.03610   Class :character   Class :character
Mode  :character   Median :0.04890   Mode  :character   Mode  :character
                   Mean   :0.09359
                   3rd Qu.:0.09853
                   Max.   :0.94200
                   NA's   :5
   valence       music_genre
Min.   :0.0000   Length:50005
1st Qu.:0.2570   Class :character
Median :0.4480   Mode  :character
Mean   :0.4563
3rd Qu.:0.6480
Max.   :0.9920
NA's   :5
```

Cleaning up the data

Cleaning the data corrects errors in the dataset that may negatively impact a predictive model and also helps make further analysis more efficient.Cleaning the dataset involves the following steps:

- Removing duplicate rows of data
- Handling null values
- Handling string inconsistencies/ anomaly values
- Handling faulty datatypes

First the dataset was searched for duplicates and four rows were found to be redundant, so these rows were removed from the dataset.

Removing the duplicates

```
[19] which(duplicated(data))
```

10002 · 10003 · 10004 · 10005

```
[20] data = data %>% distinct()
```

```
dim(data)
```

50001 · 18

Checking for null values

```
[22] data %>%
       summarise_all(~sum(is.na(.)))
```

A data.frame: 1 × 18

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	obtained_date	valence	music_genre
<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	0	0	1	1	1	1	1	1	0	1	1	0	1	0	0	1	0

```
[23] data = data %>% drop_na()    #Dropping the rows with null values as there few
```

```
[24] data %>%
       summarise_all(~sum(is.na(.)))
```

A data.frame: 1 × 18

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	obtained_date	valence	music_genre
<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
[25] dim(data)
```

50000 · 18

```
[26] n_distinct(data$music_genre)
```

10

The data was further checked for null values and as there were very few null values,so the row with null values was dropped. The dataset after removing null and duplicates has 50000 rows and 18 columns with 10 classes in the target variable.

The dataset was also checked for inconsistencies in the character columns like mode and key. It could be that the same strings are represented differently for example, mode can have different values like “Minor”, “minor”, “min” etc. for representing Minor mode. There were no such inconsistencies found for mode, key, music_genre columns.

Handling string inconsistencies

```
unique(data$mode)   #No inconsistencies found
```

'Minor' · 'Major'

```
[28] unique(data$key)
```

'A#' · 'D' · 'G#' · 'C#' · 'F#' · 'B' · 'G' · 'F' · 'A' · 'C' · 'E' · 'D#'

```
[29] unique(data$music_genre)
```

'Electronic' · 'Anime' · 'Jazz' · 'Alternative' · 'Country' · 'Rap' · 'Blues' · 'Rock' · 'Classical' · 'Hip-Hop'

```
n_distinct(data$artist_name)
```

6863

```
[30] unique(data$artist_name)
```

'Röyksopp' · 'Thievery Corporation' · 'Dillon Francis' · 'Dubloadz' · 'What So Not' · 'Axel Boman' · 'Jordan Comolli' · 'Hraach' · 'K
'Alison Wonderland' · 'RJD2' · 'Massive Attack' · 'The Prodigy' · 'Fatboy Slim' · 'Barely Alive' · 'Daniel Avery' · 'Boombox Cartel' ·
'!!!' · 'Boogie T' · 'JOYRYDE' · 'Ottawan' · 'Bonobo' · 'Tangerine Dream' · 'Tomáš Dvořák' · 'Justice' · 'GusGus' · 'Dion Timmer' · '
'Jauz' · 'The Chemical Brothers' · 'Virtual Riot' · 'Tiga' · 'QUIX' · 'Apashe' · 'Claes Rosen' · 'Minnesota' · '3LAU' · 'Primal Scream'

The next step is to check for faulty datatypes. The tempo column is a character datatype but it has numerical values, so the column needs to be converted to a numeric. The tempo column is found to have 4980 values that are not numerical i.e not containing 0-9 and a decimal point. This condition was checked with regular expressions ***[^0-9.]***. These values will get converted to null values (NA) when we typecast the column from character to numeric datatype. The 4980 nulls would have to be imputed as such a large number of rows cannot be dropped.

Handling faulty data types

```
count = sum(stri_count(data$tempo,regex='[^0-9.]'))  #count of non-numeric rows in tempo
count
```

```
4980
```

```
[33] data$tempo <- as.numeric(as.character(data$tempo)) #converting char to numeric
```

```
Warning message in eval(expr, envir, enclos):
"NAs introduced by coercion"
```

```
[34] data %>%
       summarise_all(~sum(is.na(.)))     #The non-numeric values converted to NA
```

A data.frame: 1 × 18

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	c
<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4980	

```
[35] sum(is.na(data$tempo))
```

```
4980
```

Checking for class imbalance

Imbalanced classification is challenging because of the skewed class distribution. If class distribution is not balanced, machine learning algorithms will perform poorly. Class imbalance happens when some classes have large number of samples as compared to others. In such situations the model will end up predicting the class with large number of samples most of the time and fail to predict the minority classes properly. Machine learning models and evaluation metrics perform poorly as they assume balanced class distribution. If there is class imbalance then undersampling or oversampling techniques should be used to fix the class imbalance.

Checking for class imbalance

```
[ ]  target = table(data$music_genre)
     target
```

```
Alternative       Anime       Blues   Classical     Country  Electronic
       5000        5000        5000        5000        5000        5000
    Hip-Hop        Jazz         Rap        Rock
       5000        5000        5000        5000
```

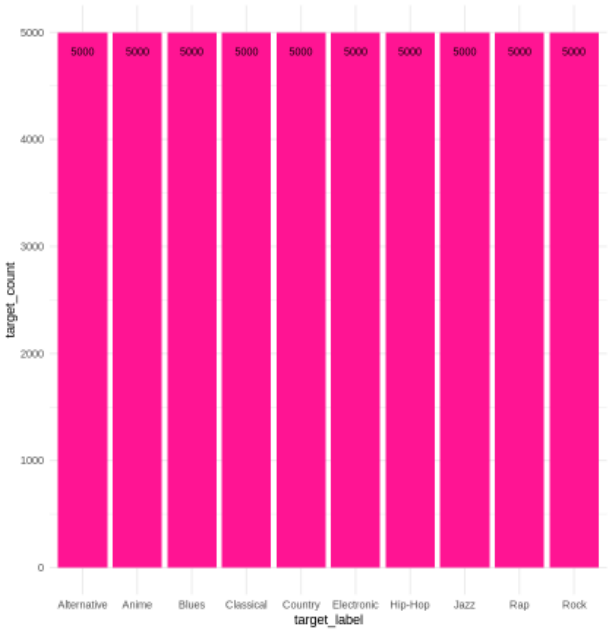
```
[ ]  target_label = names(target)
     target_count = as.numeric(target)
     target_label
     target_count
```

```
'Alternative' · 'Anime' · 'Blues' · 'Classical' · 'Country' · 'Electronic' · 'Hip-Hop' · 'Jazz' · 'Rap' · 'Rock'
5000 · 5000 · 5000 · 5000 · 5000 · 5000 · 5000 · 5000 · 5000 · 5000
```

```
[ ]  trgt_df = data.frame(label= c(target_label), count= c(target_count))
```

The target variable music_genre has 10 genre classes Alternative, Anime, Blues, Classical, Country, Electronic, Hip-Hop, Jazz, Rap, Rock. All of them have the same number of samples. They all have 5000 data points each, so no class imbalance is present in the dataset. No sampling techniques need to be applied in this case.

```
ggplot(data=trgt_df, aes(x=target_label, y=target_count)) +
  geom_bar(stat="identity", fill="deeppink")+
  geom_text(aes(label=target_count), vjust=3, color="black", size=3)+
  theme_minimal()                                                    #no class imbalance
```



After performing tasks like data imputation, outlier removal, the target variable was checked again for class imbalance. Here even though the datapoints under each genre are not exactly equal they are still very close to each other. As a result, there won't be much class imbalance. So no need to perform sampling.

```
target_check = table(no_outliers$music_genre)
target_check
```

```
Alternative       Anime       Blues   Classical     Country  Electronic
       4507        4517        4476        4212        4508        4496
    Hip-Hop        Jazz         Rap        Rock
       4509        4468        4488        4487
```

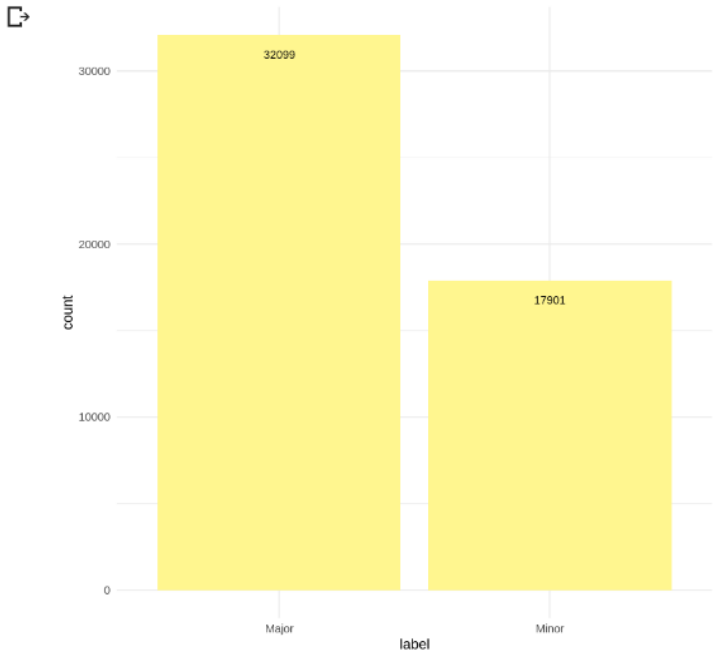
Data exploration

Barcharts

In data exploration, the data is further explored with plots like bargraphs, line charts, and also by using functions such as group.by(), order(), summarise(), to find some interesting insights. Exploratory plots help highlight any broad features, patterns, trends present in the data.

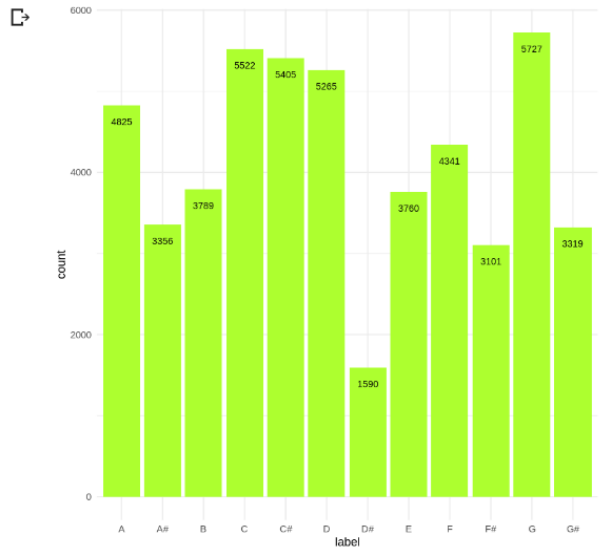
Barplots are useful for visualizing categorical data, with the number of entries for each category being proportional to the height of the bar. The first plot shows feature mode and the count of datapoints in major and minor. There are 32099 datapoints with mode as major and 17901 with mode as minor. This shows that more songs have mode as major.

```
ggplot(data=mode_df, aes(x=label, y=count)) +
  geom_bar(stat="identity", fill="khaki1")+
  geom_text(aes(label=count), vjust=3, color="black", size=3)+
  theme_minimal()
```



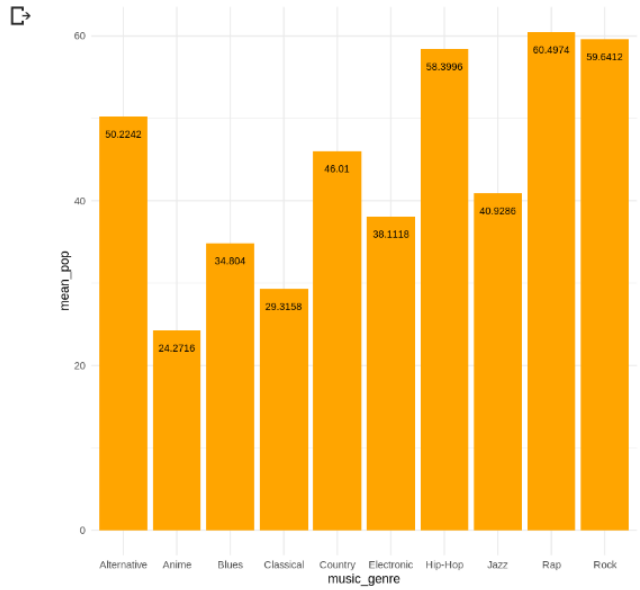
The second plot shows feature key and the count of datapoints in different keys (A, A#, B, C, C#, D, D#, E, F, F#, G, G#). There are 5727 datapoints with key G making it the most popular key, followed by C, C#, D and A in order. Key D# is the least popular key with only 1590 songs in that key.

```
ggplot(data=key_df, aes(x=label, y=count)) +
  geom_bar(stat="identity", fill="greenyellow")+
  geom_text(aes(label=count), vjust=3, color="black", size=3)+
  theme_minimal()
```



The third plot shows mean popularity of each genre. This gives the overall popularity of the genre in the time period. The most popular genre was found to be rap with a mean popularity of 60 followed by rock with mean popularity of 59 and Hip-Hop with 58. The least popular genre was anime with mean popularity of 24.

```
ggplot(data=pop_df, aes(x=music_genre, y=mean_pop)) +
  geom_bar(stat="identity", fill="orange")+
  geom_text(aes(label=mean_pop), vjust=3, color="black", size=3)+
  theme_minimal()
```



```
artist_df = data %>%
  group_by(artist_name) %>%
  summarise(mean_pop = mean(popularity))

artist_df = artist_df[order(-artist_df$mean_pop),]

head(artist_df,10)
```

A tibble: 10 × 2

artist_name	mean_pop
<chr>	<dbl>
Duki	82.00000
Heuss L'enfoiré	81.00000
NSG	81.00000
Coolio	80.00000
Danny Ocean	80.00000
Ben E. King	79.00000
Snow Patrol	78.00000
Post Malone	77.23333
Cadet	77.00000
Jet	77.00000

```
track_df = data %>%
  group_by(track_name) %>%
  summarise(mean_pop = mean(popularity))

track_df = track_df[order(-track_df$mean_pop),]

head(track_df,10)
```

A tibble: 10 × 2

track_name	mean_pop
<chr>	<dbl>
Wow.	99
Sunflower - Spider-Man: Into the Spider-Verse	97
MIDDLE CHILD	96
Taki Taki (with Selena Gomez, Ozuna & Cardi B)	96
Adan y Eva	95
Going Bad (feat. Drake)	95
SICKO MODE	94
a lot	93
SAD!	92
Pure Water (with Migos)	91

The above two tibbles show the top ten artists and top ten tracks (based on mean popularity). Artists Duki topped with mean popularity of 82, followed by Heuss Lenfoire and NSG topped with mean popularity of 81. Wow and Sunflower from Spiderman topped in track list with mean popularity of 99 and 97.

The dataframe below displays the top 2 songs in each genre with the highest mean popularity of songs within that genre. This was found by grouping the data by genre and computing the mean popularity of each track. A track can have multiple versions of it like slow, reverbed, remix etc. with different popularity rating. So finding mean popularity for track name takes into account the different versions of the song. After finding mean popularity, slice() function is used to obtain the top 2 tracks in each genre.

```
top2 = genre_track[order(-genre_track$mean_pop),] %>% slice(1:2)
top2
```

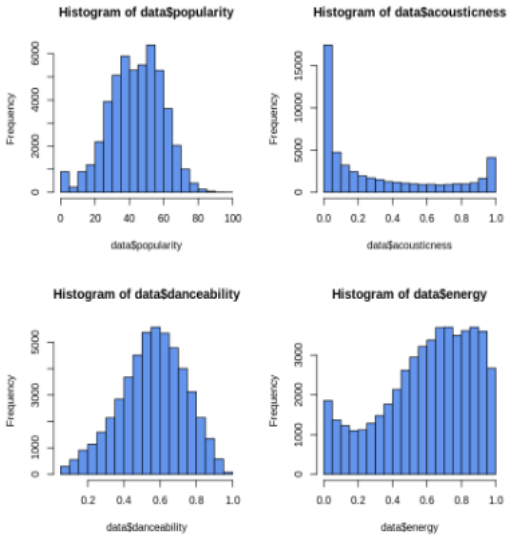
A grouped_df: 20 × 3

music_genre	track_name	mean_pop
<chr>	<chr>	<dbl>
Alternative	Sanctuary	83
Alternative	SLOW DANCING IN THE DARK	81
Anime	Silhouette	65
Anime	Unravel	65
Blues	The Joker	74
Blues	Ain't No Rest for the Wicked	72
Classical	Piano Concerto No. 21 in C Major, K. 467 "Elvira Madigan": II. Andante	68
Classical	Sonata No. 14 "Moonlight" in C-Sharp Minor", Op. 27 No. 2: I. Adagio sostenuto	68
Country	Beautiful Crazy	82
Country	Sweet Home Alabama	82
Electronic	Taki Taki (with Selena Gomez, Ozuna & Cardi B)	96
Electronic	Bola Rebola	84
Hip-Hop	MIDDLE CHILD	96
Hip-Hop	Adan y Eva	95
Jazz	September	79
Jazz	Miss Me More	74
Rap	Wow.	99
Rap	Sunflower - Spider-Man: Into the Spider-Verse	97
Rock	Bad Liar	90
Rock	Believer	89

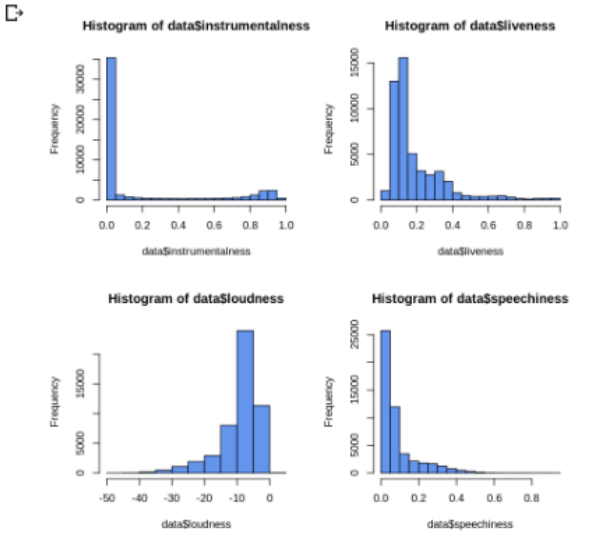
Histograms

Histograms are used to summarise continuous data by dividing the range of possible values in the dataset into groups. It helps in determining the mean, mode and distribution of data.

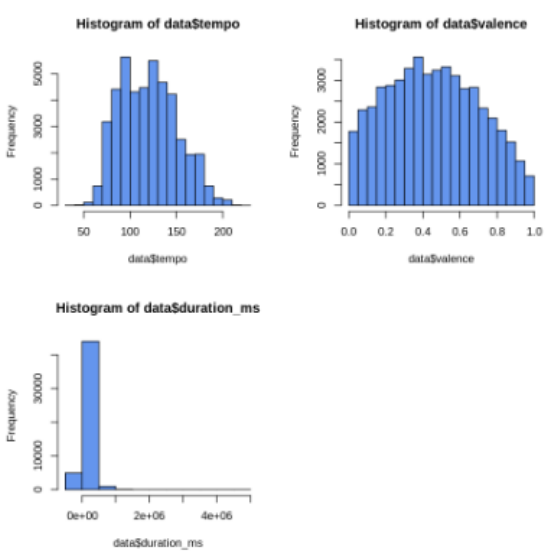
```
par(mfrow = c(2,2))
hist(data$popularity,col = "cornflowerblue")
hist(data$acousticness, col = "cornflowerblue")
hist(data$danceability, col = "cornflowerblue")
hist(data$energy, col = "cornflowerblue")
```




```
par(mfrow = c(2,2))
hist(data$instrumentalness, col = "cornflowerblue")
hist(data$liveness,col = "cornflowerblue")
hist(data$loudness, col = "cornflowerblue")
hist(data$speechiness, col = "cornflowerblue")
```




```
par(mfrow = c(2,2))
hist(data$tempo, col = "cornflowerblue")
hist(data$valence,col = "cornflowerblue")
hist(data$duration_ms, col = "cornflowerblue")
```

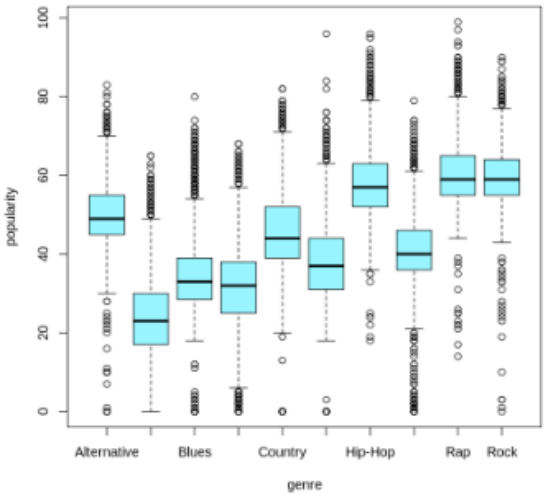


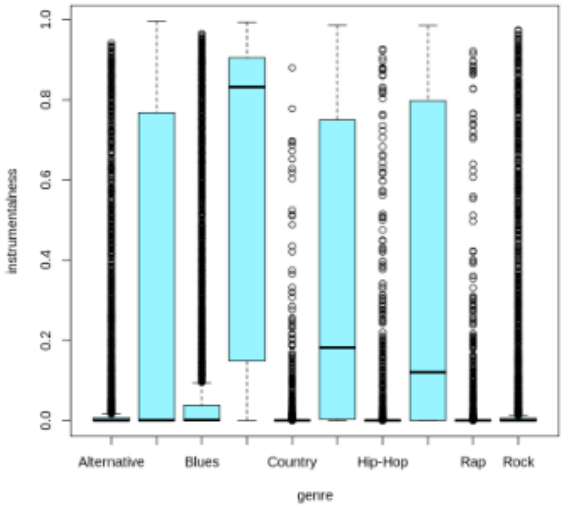
Summary of insights from the histograms

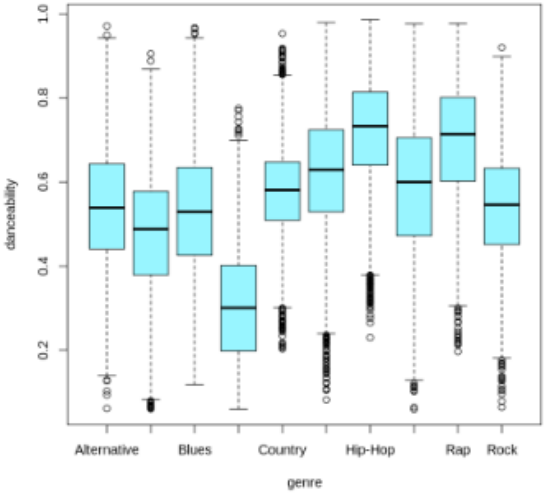
- Danceability and Valence seem to have a normal distribution and unimodal.
- Liveness, instrumentalness, acousticness, speechiness, duration_ms all seem to have skewness towards the right.
- Loudness, energy is skewed towards the left.
- Tempo and popularity are bimodal distributions.

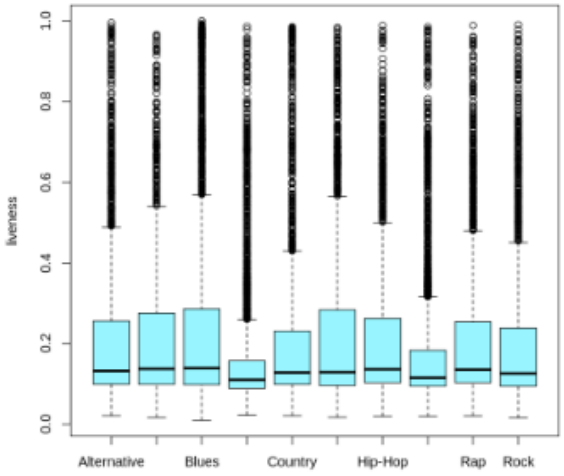
Boxplots

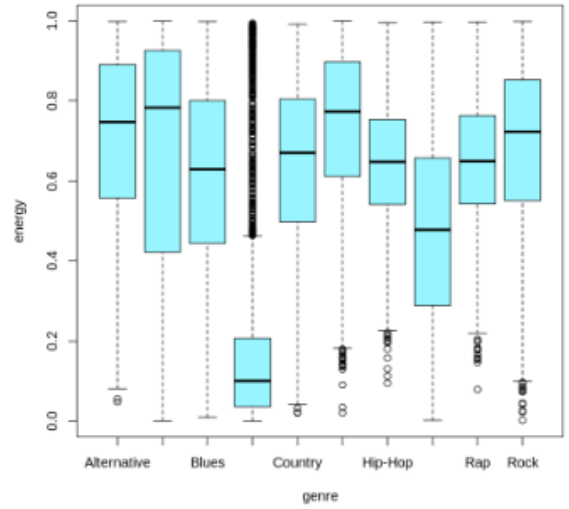
Boxplots give a visual summary of data and help in identifying the median, dispersion of data, skewness, outliers with ease. They give information about the minimum, maximum, first quartile, third quartile and the median.

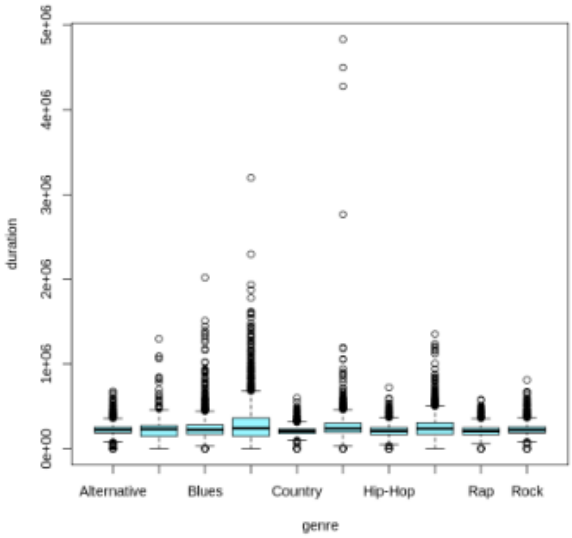


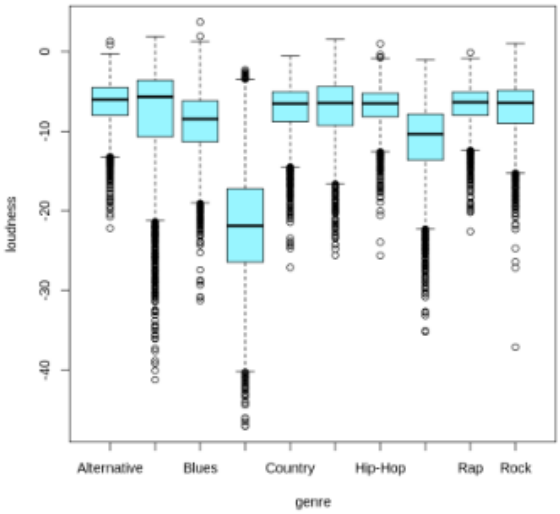


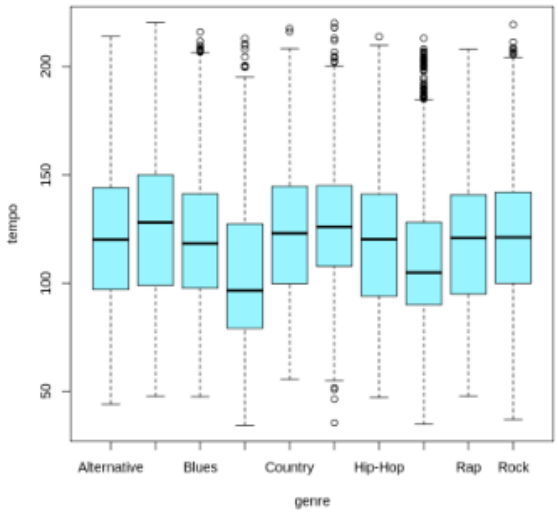


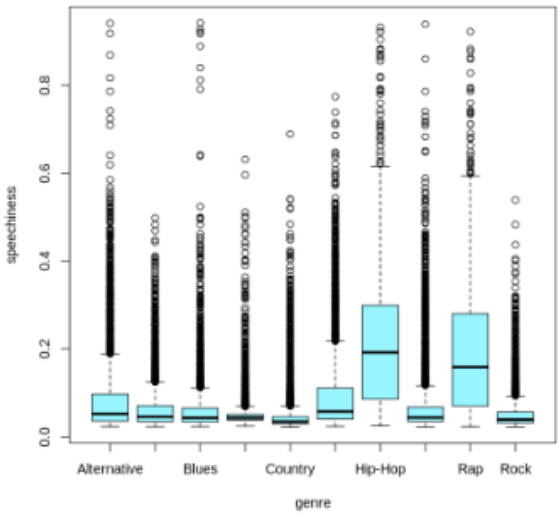


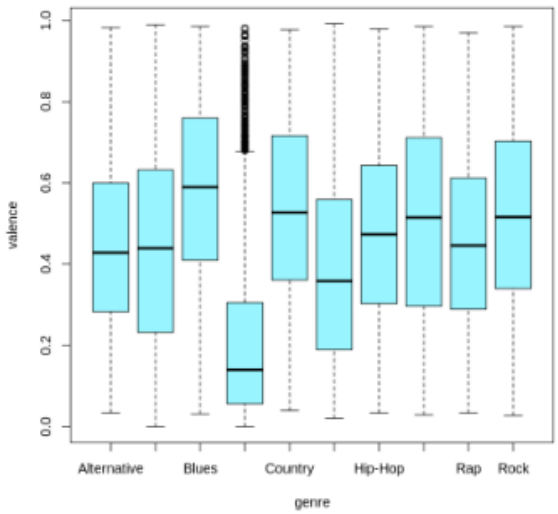

energy
0.0
0.2
0.4
0.6
0.8
1.0
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre


duration
0e+00
1e+06
2e+06
3e+06
4e+06
5e+06
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre


loudness
-40
-30
-20
-10
0
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre


tempo
50
100
150
200
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre


speechiness
0.0
0.2
0.4
0.6
0.8
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre


valence
0.0
0.2
0.4
0.6
0.8
1.0
Alternative
Blues
Country
Hip-Hop
Rap
Rock
genre

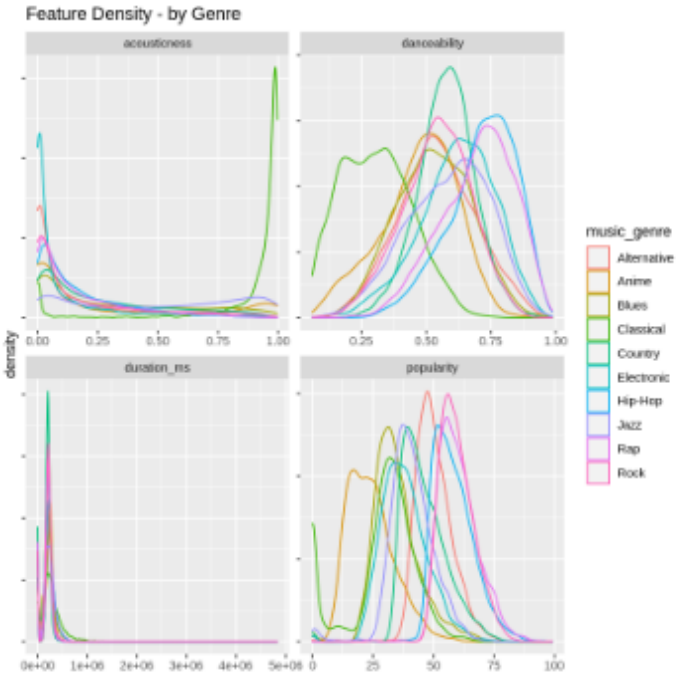
In the boxplots above, the circular points represent the outliers, the rectangular blue box represent the 25th to 75th percentile of data (interquartile range), the bold black line within the blue box represents the median and the whiskers at the end represent the minimum and maximum values.

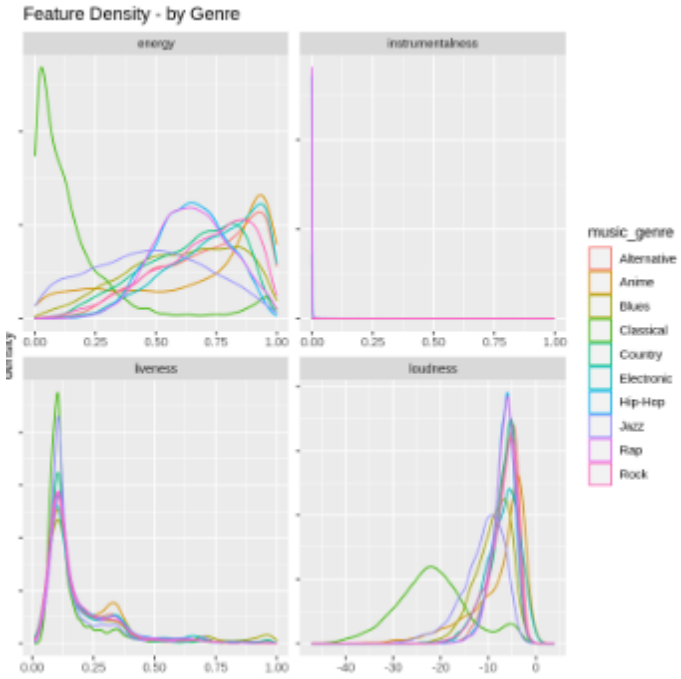
Summary of insights from the boxplots

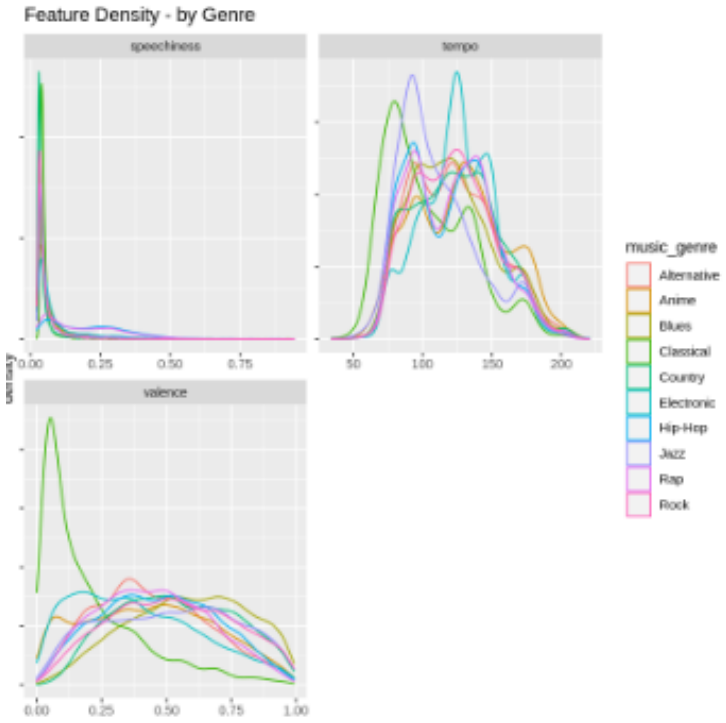
- Instrumentalness, liveness, popularity, speechiness, duration all seem to have outliers.
- Liveness, duration_ms, speechiness have the boxplots towards the bottom indicating a right skewed distribution.
- Loudness and energy have the boxplots towards the top indicating a left skewed distribution.
- In liveness and tempo the medians of all the genres seem to be similar.

Density Plots

Density plots represent the distribution of a numeric variable by using kernel density estimate to give the probability density function of the variable. They are better than histograms too find the distribution because they are not affected by bins.







Summary of insights from the density plots

- In acousticness, classical genre's curve is left skewed while all other genres are right skewed. This points to the major use of string instruments in classical.
- In danceability, classical genre's curve is right skewed which implies classical music is less danceable.
- In instrumentalness, rap's curve is almost towards zero, showing that rap has large amount of vocals and less instrument.
- In energy, classical genre's curve is right skewed implying its less fast and less energetic.
- In loudness, classical's curve follows edge peak distribution.

Data Imputation

Data imputation is the substitution of missing or inconsistent values with meaningful values. When dealing with missing data, if percentage of missing data is very low then we can resort to dropping the rows, but if the percentage of missing values is high, we should impute the missing values.

In the tempo column, there are 4980 null or missing values present which is a large number to drop, so we need to do imputation. By looking at the boxplot and histogram of the tempo variable we know it is not a skewed distribution, so mean imputation can be done. In mean imputation, the missing values are replaced by the mean of the column, but this might not be very accurate. So rather than imputing the missing values with the mean of entire tempo column, they were imputed with the mean tempo of the particular genre. The data was grouped by music_genre using the group_by() function and each genre group's missing values were imputed with the mean tempo of that genre group. The code and output are shown below.

```
data %>%
  group_by(music_genre) %>%
  summarise(mean_tempo = mean(tempo,na.rm=TRUE))
```

A tibble: 10 × 2

music_genre	mean_tempo
<chr>	<dbl>
Alternative	122.5472
Anime	126.8000
Blues	121.3800
Classical	104.0532
Country	123.7843
Electronic	125.9306
Hip-Hop	120.1541
Jazz	111.6994
Rap	120.5855
Rock	122.6696

```
data %>%                    #count of NA by category
    group_by(music_genre) %>%
    summarise((across(tempo,  ~sum(is.na(.)))))
```

A tibble: 10 × 2

music_genre	tempo
<chr>	<int>
Alternative	505
Anime	503
Blues	530
Classical	500
Country	514
Electronic	534
Hip-Hop	480
Jazz	479
Rap	496
Rock	439

Each genre group's mean tempo is calculated. Each genre group has around 450-500 missing values and these values are imputed by the mean tempo for that genre. Imputing values based on music_genre would be more accurate way to impute the tempo values.

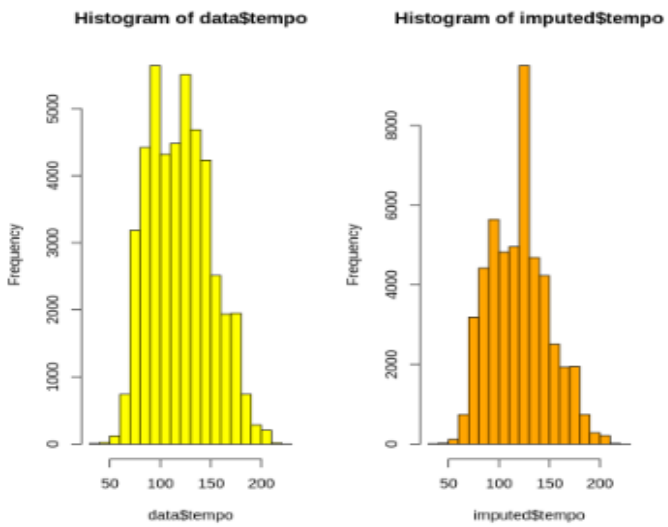
```
imputed = data %>%
            group_by(music_genre) %>%
            mutate(tempo= ifelse(is.na(tempo),mean(tempo,na.rm = TRUE),tempo))

head(imputed)
```

A grouped_df: 6 × 18

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	obtained_date	valence	music_genre
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<dbl>	<chr>
32894	Röyksopp	Röyksopp's Night Out	27	0.00468	0.652	-1	0.941	0.79200	A#	0.115	-5.201	Minor	0.0748	100.8890	4-Apr	0.759	Electronic
46652	Thievery Corporation	The Shining Path	31	0.01270	0.622	218293	0.890	0.95000	D	0.124	-7.043	Minor	0.0300	115.0020	4-Apr	0.531	Electronic
30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	0.01180	G#	0.534	-4.617	Major	0.0345	127.9940	4-Apr	0.333	Electronic
62177	Dubloadz	Nitro	34	0.02540	0.774	166875	0.700	0.00253	C#	0.157	-4.498	Major	0.2390	128.0140	4-Apr	0.270	Electronic
24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	0.90900	F#	0.157	-6.266	Major	0.0413	145.0360	4-Apr	0.323	Electronic
89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	0.85400	D	0.216	-10.517	Minor	0.0412	125.9306	4-Apr	0.614	Electronic

The histogram below shows the distribution of the tempo feature before and after data imputation was performed. The one peak is because of mean imputation, because alot of genres have means in 120-125 range.

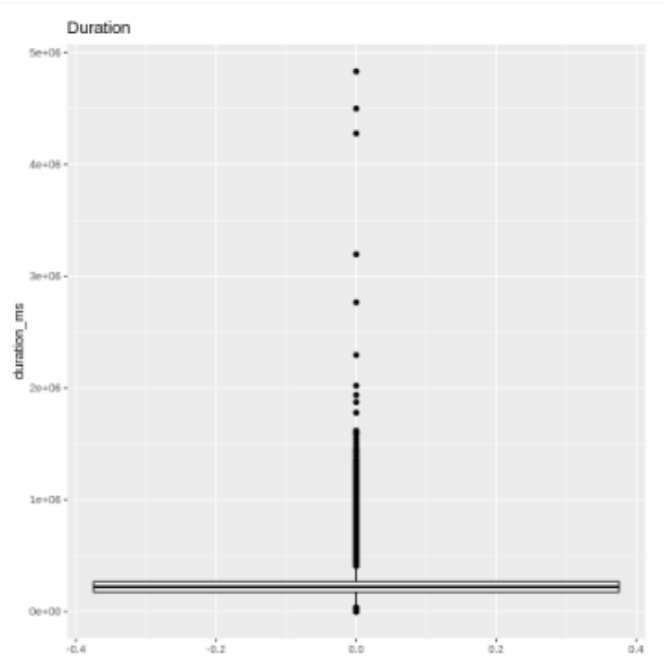
```
par(mfrow=c(1,2))
hist(data$tempo, col="yellow")
hist(imputed$tempo, col="orange")
```

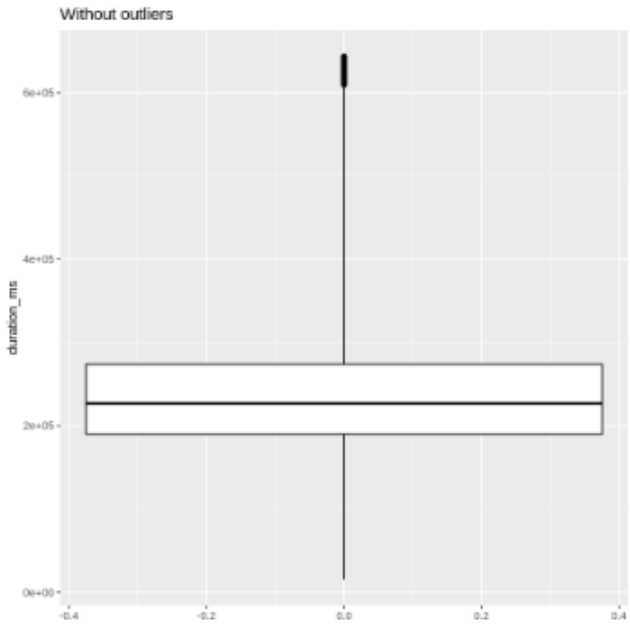


Removing Outliers from data

An outlier is a datapoint that is significantly different from the other observations in a dataset. Some outliers represent natural variations in the data, and they should not be removed from the dataset and are called true outliers. Some outliers might have to be removed because they represent errors, anomalies or poor sampling.

With the help of boxplots, we identified outliers in variables like speechiness, liveness, loudness, duration. The outliers in features like speechiness, liveness, instrumentalness are all within 0 to 1 values i.e they are within the prescribed range of values of their measurement. These outliers have not been removed as they represent natural variation in the data. Including these outliers would help the model perform better on data it has not seen before and minority data. Duration_ms feature has erroneous values like -1 and 4830606 which don't make sense and need to be removed. Anything over 4 times the interquartile range and values below 2.5th percentile have been removed. The boxplot before and after removing outliers are shown below.





Encoding Categorical Features

Many machine learning models only accept numerical feature values as input and cannot handle categorical data as input. These categorical features need to be converted to numeric values by a process called encoding where each category is assigned an integer values.

In the dataset, mode, key, music_genre are all categorical. Music_genre is the target variable and mode and key are independent variables. One hot encoding was performed for mode and key columns. In one hot encoding n new columns are created (where n is the number of categories of that variable) and 1 is added to its corresponding column and 0 to all other n-1 columns. In label encoding, each category is assigned an integer value and label encoding was performed on target variable music_genre. Each genre was assigned an integer value. The output data after encoding is shown below.

```
encoded_data = one_hot(as.data.table(no_outliers))
head(encoded_data)
```

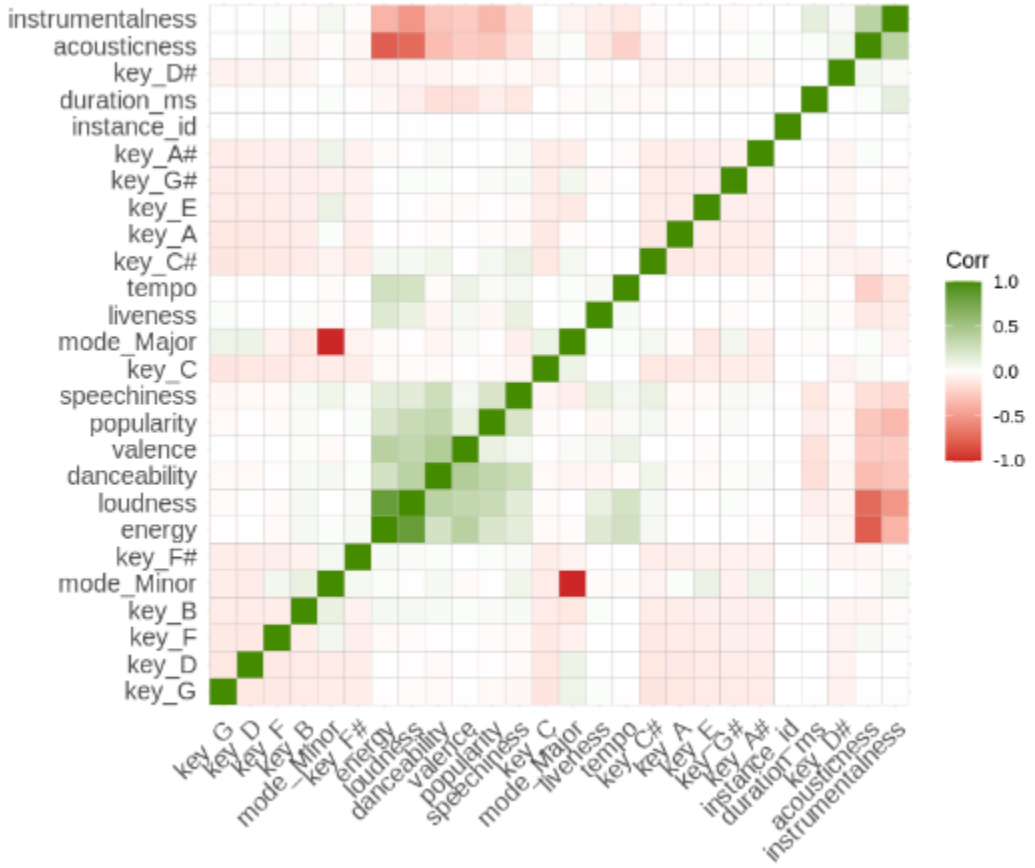
A data.table: 6 × 30

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key_A	⋯	key_G#	liveness	loudness	mode_Major	mode_Minor	speechiness	tempo	obtained_date	valence	music_genre
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	⋯	<int>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl>	<chr>	<dbl>	<chr>
46652	Thievery Corporation	The Shining Path	31	0.01270	0.622	218293	0.890	9.50e-01	0	⋯	0	0.124	-7.043	0	1	0.0300	115.0020	4-Apr	0.531	Electronic
30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	1.18e-02	0	⋯	1	0.534	-4.617	1	0	0.0345	127.9940	4-Apr	0.333	Electronic
62177	Dubloadz	Nitro	34	0.02540	0.774	166875	0.700	2.53e-03	0	⋯	0	0.157	-4.498	1	0	0.2390	128.0140	4-Apr	0.270	Electronic
24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	9.09e-01	0	⋯	0	0.157	-6.266	1	0	0.0413	145.0360	4-Apr	0.323	Electronic
89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	8.54e-01	0	⋯	0	0.216	-10.517	0	1	0.0412	125.9306	4-Apr	0.614	Electronic
43760	Jordan Comolli	Clash	46	0.02890	0.572	214408	0.803	7.74e-06	0	⋯	0	0.106	-4.294	1	0	0.3510	149.9950	4-Apr	0.230	Electronic

```
map = c('Alternative'=1,'Anime'=2,'Blues'=3,'Classical'=4,'Country'=5,'Electronic'=6,'Hip-Hop'=7,'Jazz'=8,'Rap'=9,'Rock'=10)
```

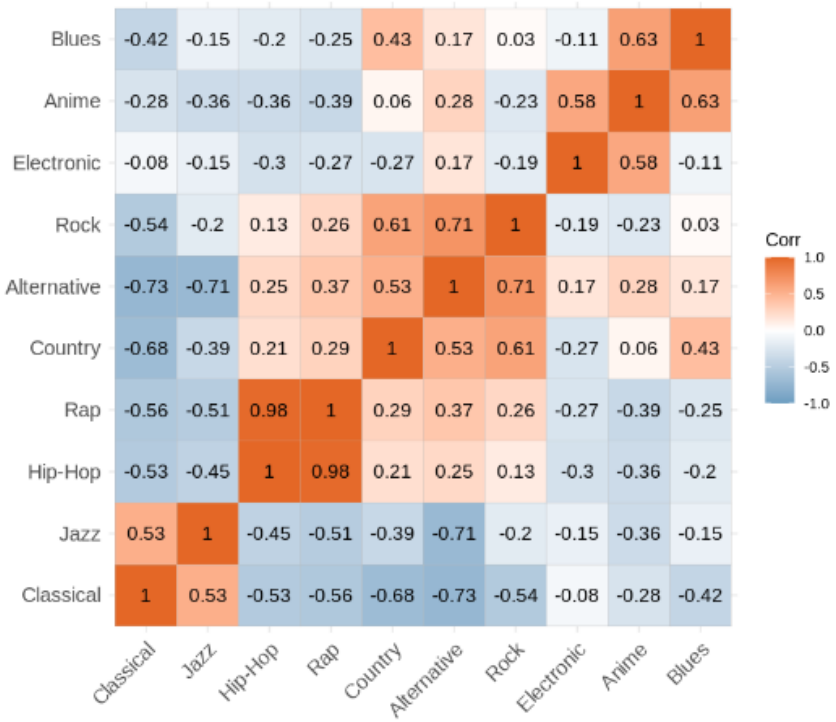
Correlation

Correlation is the dependence between two random variables. It expresses the exent to which two variables are linearly related. Correlation matrix displays this correlation between different variables in a matrix. The diagonals have a value of 1 showing that each variable is highly correlated to itself. The green cells (0 to 1 values) represent positive correlation while red cells (0 to -1) represent negative correlation.



Summary of insights from feature correlation matrix

- Loudness and energy are highly positively correlated. This means that one of the variables either loudness or energy can be dropped before modelling as they have the same impact on it.
- Loudness and energy are also positively correlated with danceability, valence, popularity and speechiness.
- Mode major and minor are highly negatively correlated
- Energy and acousticness are negatively correlated
- Loudness and acousticness are negatively correlated
- Instrumentallness and loudness are negatively correlated



Summary of insights from genre correlation matrix

- Each genre is highly correlated with itself.
- Rap and Hip-Hop are highly positively correlated.
- Rock and alternative are positively correlated.
- Rock and Country are positively correlated.
- Anime and Electronic are positively correlated.
- Anime and Blues are positively correlated.
- Alternative and classical are negatively correlated.
- Alternative and Jazz are negatively correlated.
- Country and classical are negatively correlated.
- Rap and classical are negatively correlated
- Hip-Hop and classical are negatively correlated.

Scaling the data

Standardizing or scaling data makes the attributes to have a zero mean and variance of 1. Scaling must be done when the range of values for different attributes vary. Variables that are measured at different scales do not contribute equally to the analysis and might cause bias. The variables with large values will contribute more or have more influence on the model. Scaling the makes the range uniform. In the dataset, features like loudness have negative values and duration (in milliseconds) has large positive values. The data was scaled using scale() function. The output of data after scaling is shown below.

```
scaled_data = encoded_data %>%
   mutate_at(c(4:9,22:23,26:27,29), list(~c(scale(.))))

head(scaled_data)
```

A data.table: 6 × 30

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key_A	⋯	key_G#	liveness	loudness	mode_Major	mode_Minor	speechiness	tempo	obtained_date	valence
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	⋯	<int>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl>	<chr>	<dbl>
46652	Thievery Corporation	The Shining Path	-0.8620880	-0.8545075	0.34696392	-0.2575814	1.09107118	2.3928336	0	⋯	0	-0.4333897	0.3287244	0	1	-0.6285765	-0.1719359	4-Apr	0.2911733
30097	Dillon Francis	Hurricane	-1.0556701	-0.8829633	0.33565847	-0.2901090	0.57717063	-0.5137700	0	⋯	1	2.1164068	0.7267707	1	0	-0.5843629	0.2738172	4-Apr	-0.5134954
62177	Dubloadz	Nitro	-0.6685058	-0.8170190	1.20617823	-0.8816497	0.36780374	-0.5424890	0	⋯	0	-0.2281622	0.7462956	1	0	1.4249000	0.2745034	4-Apr	-0.7695263
24907	What So Not	Divide & Conquer	-0.7975606	-0.8782699	0.43740753	-0.2081104	-0.06235005	2.2658130	0	⋯	0	-0.2281622	0.4562108	1	0	-0.5175512	0.8585250	4-Apr	-0.5541352
89064	Axel Boman	Hello	0.1703501	-0.8765578	1.09877644	3.3978266	0.48581053	2.0954195	0	⋯	0	0.1387598	-0.2412726	0	1	-0.5185337	0.2030213	4-Apr	0.6284839
43760	Jordan Comolli	Clash	0.1058227	-0.8066875	0.06432763	-0.3047343	0.75989083	-0.5503031	0	⋯	0	-0.5453320	0.7797669	1	0	2.5253277	1.0286674	4-Apr	-0.9320857

Chi-Square Test

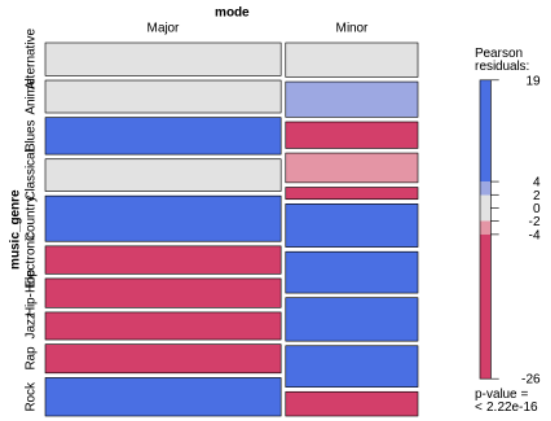
Chi-squared test is used to test the independence between two categorical variables. The null hypothesis assumes the variables are independent. The Chi-square test works by comparing the frequencies observed to the expected frequencies. If the p-value is less than 0.05 it is not statistically significant and we can reject null hypothesis. Below the p-value is less than 0.05, so we can reject the null hypothesis, which means that the variables mode, genre , and key, genre are dependent on each other.

The mosaic plot shows where the observed frequencies deviated from the expected value. The pink cells show where observed was smaller than expected frequency and the blue cells show where observed was greater than expected frequency.

```
chisq.test(table(no_outliers$mode, no_outliers$music_genre))
```

```
        Pearson's Chi-squared test

data:  table(no_outliers$mode, no_outliers$music_genre)
X-squared = 2090.3, df = 9, p-value < 2.2e-16
```

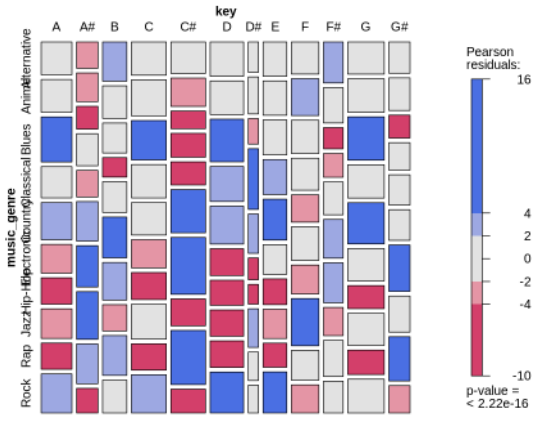
```
mosaic(~ mode + music_genre,
  direction = c("v", "h"),
  data = no_outliers,
  shade = TRUE
)
```




```
chisq.test(table(no_outliers$key, no_outliers$music_genre))
```

```
        Pearson's Chi-squared test

data:  table(no_outliers$key, no_outliers$music_genre)
X-squared = 2192, df = 99, p-value < 2.2e-16
```

```
mosaic(~ key + music_genre,
  direction = c("v", "h"),
  data = no_outliers,
  shade = TRUE
)
```



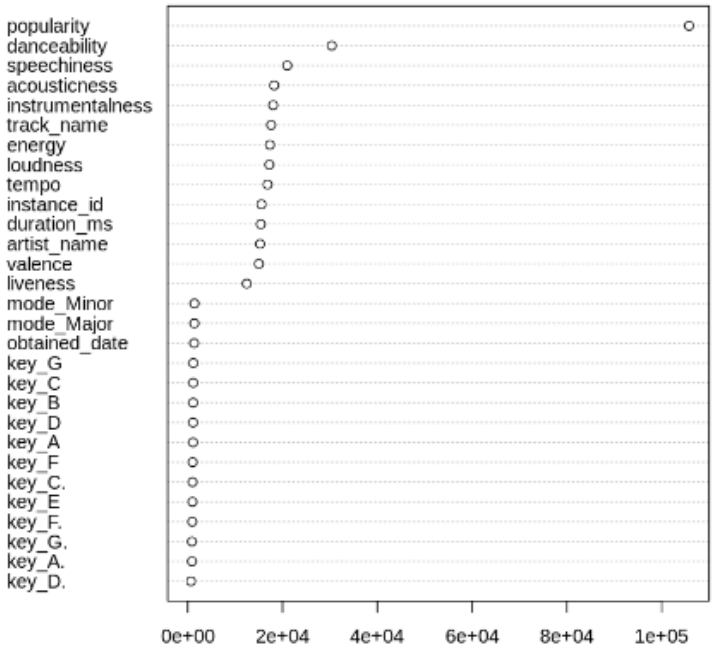
Feature Selection

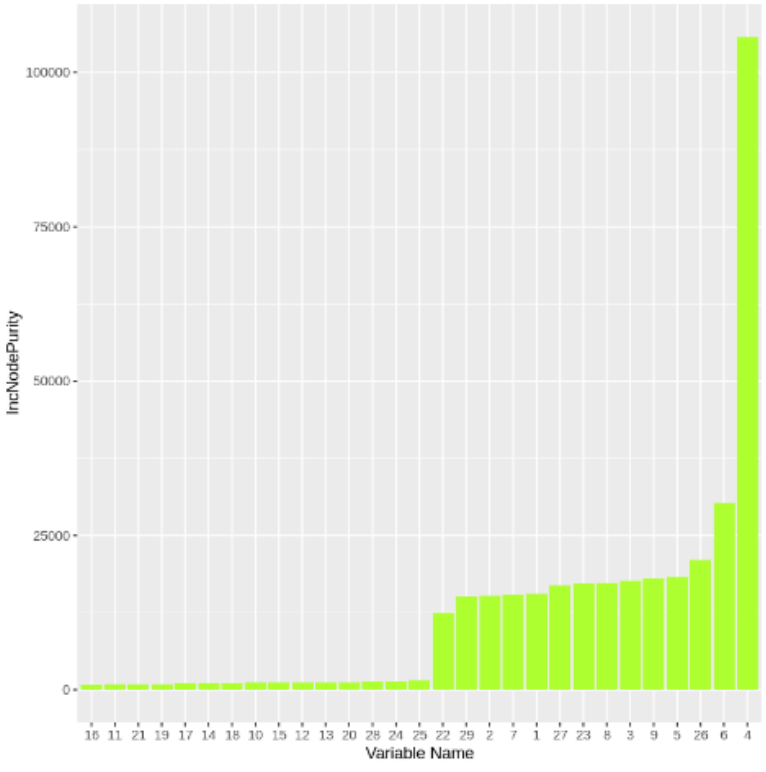
Feature selection is the process of selecting features that would influence the model. It reduces the number of input variables to the model. The selected features are a subset of the independent variables. Random forest algorithm is used for feature selection because it is accurate, less overfitting and performs well if features are correlated. The output gives the list of features with node purity values. Higher the value, more important the feature.

```
rf = randomForest(target_genre ~ ., data = var_data)
imp = importance(rf)
imp
```

	IncNodePurity		
instance_id	15548.4819	key_D	1128.6476
artist_name	15268.0101	key_D.	675.9868
track_name	17611.9646	key_E	1000.5953
popularity	105769.0377	key_F	1042.0368
acousticness	18247.8024	key_F.	955.2619
danceability	30374.5863	key_G	1177.5795
duration_ms	15422.7991	key_G.	897.1462
energy	17340.8155	liveness	12462.9604
instrumentalness	18028.8324	loudness	17214.9652
key_A	1121.6760	mode_Major	1412.8868
key_A.	896.1923	mode_Minor	1422.6969
key_B	1143.5396	speechiness	20989.2732
key_C	1163.6198	tempo	16859.1334
key_C.	1018.9071	obtained_date	1363.2145

The first graph below plots the feature name and the node purity value on Y and X axis respectively. The Y-axis displays the features in order of importance. The bargraph has the features on the X axis and the node purity values on the Y-axis. The X axis shows the index value of the features. From these charts we can see that popularity, danceability, speechiness, acousticness, instrumentalness, energy are the top five important features.


popularity
danceability
speechiness
acousticness
instrumentalness
track_name
energy
loudness
tempo
instance_id
duration_ms
artist_name
valence
liveness
mode_Minor
mode_Major
obtained_date
key_G
key_C
key_B
key_D
key_A
key_F
key_C.
key_E
key_F.
key_G.
key_A.
key_D.
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
IncNodePurity


100000
75000
50000
25000
0
IncNodePurity
16 11 21 19 17 14 18 10 15 12 13 20 28 24 25 22 29 2 7 1 27 23 8 3 9 5 26 6 4
Variable Name

References

1. https://bookdown.org/dli/rguide/
2. https://rcompanion.org/handbook/C_01.html
3. https://www.rdocumentation.org/
4. http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf
5. https://bookdown.org/ndphillips/YaRrr/
6. https://www.tmwr.org/
7. https://benwhalley.github.io/just-enough-r/
8. https://rdrr.io/
9. https://www.statmethods.net/index.html
10. https://www.r-bloggers.com/2020/02/a-guide-to-encoding-categorical-features-using-r/
11. https://koalatea.io/r-one-hot-encoding/
12. https://www.statology.org/scale-function-in-r/
13. https://cran.r-project.org/web/packages/ggcorrplot/readme/README.html
14. https://statsandr.com/blog/outliers-detection-in-r/
15. https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/
16. https://www.statology.org/variance-inflation-factor-r/
17. https://www.linkedin.com/pulse/how-find-most-important-variables-r-amit-jain/
18. https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/
19. https://data-flair.training/blogs/chi-square-test-in-r/
20. https://www.jmp.com/en_us/statistics-knowledge-portal/exploratory-data-analysis/mosaic-plot.html

21. https://towardsdatascience.com/mosaic-plot-and-chi-square-test-c41b1a527ce4
22. https://www.linkedin.com/pulse/how-find-most-important-variables-r-amit-jain/
23. https://www.r-bloggers.com/2015/06/variable-importance-plot-and-variable-selection/
24. https://www.kaylinpavlik.com/classifying-songs-genres/
25. https://rpubs.com/jsl0516/spotify_genres