



Machine Learning Project

Problem Statement - Spotify Genre Classification

Team Members: Rishika Anil Gupta and Satria Bagus Wicaksono



Table of Contents:

1. Brief Introduction.....	3
1.1. Problem Statement.....	3
1.2. Motivation.....	3
1.3. Dataset Description.....	3
1.4. Column Description.....	3
1.5. Summary of work done.....	4
2. Data Exploration and Pre-processing.....	5
2.1. Exploratory data analysis.....	5
2.1.1. Target column - Genre.....	5
2.1.2. Numerical columns.....	5
2.1.3. Categorical columns.....	7
2.2. Exploring the features further.....	8
2.2.1. 'Unnamed: 21' and 'song_name' columns.....	8
2.2.2. 'Unnamed: 20' column.....	9
2.2.3. 'id', 'uri', 'track_href', 'analysis_url' columns.....	9
2.2.4. 'type' column.....	9
2.3. Splitting the data into train-test.....	9
2.4. Feature Importance, Selection, Extraction & Engineering.....	9
2.4.1. Feature Importance.....	9
2.4.2. Feature Selection.....	9
2.4.3. Feature Extraction.....	10
2.4.4. Feature Engineering.....	10
2.5. Feature Analysis and Visualisation.....	11
3. Model Training and Testing.....	11
3.1. Data Duplication.....	11
3.2. Data Preparation.....	12
3.2.1. Genre column.....	12
3.2.2. Speechiness column.....	12
3.2.3. Transformations: Boxcox and Logarithmic.....	12
3.2. Data Scaling.....	12
3.3. Oversampling (Upsampling).....	12
4. Model Implementations.....	12
5. Comparison of Results.....	13
6. Final Chosen Model.....	13
7. Conclusions.....	13
8. Futurework and limitations.....	14
9. References.....	14



1. Brief Introduction

1.1. Problem Statement

This project aims to classify genres for songs under the spotify dataset^[1]. The problem statement for the same could be defined as: -
“Given a dataset containing song attributes (loudness, tempo, duration, etc), predict the genre of the song.”

1.2. Motivation

Spotify is a popular music streaming service that allows users to access a vast library of songs from various artists and genres. With over 70 million tracks and counting, Spotify's catalog is one of the largest in the world. It offers a wide range of genres, including pop, hip-hop, electronic, classical, jazz, country, and many more.

The motivation is due to the benefits that it can provide to multiple significant parties as described below:-

- Users: To *better music recommendations* by identifying the different genres that a song belongs to, streaming services can recommend songs that are more likely to appeal to the user's musical tastes or *more efficient music organization* by categorizing songs into multiple genres, it becomes easier to find specific types of music when searching through a music library or playlist.
- Researchers: To *better understand the evolution and influence* of different music genres.
- Producers: To *improve the production of music* by analyzing the characteristics of successful songs across different genres, producers can identify the elements that make a song appealing to a wide audience and incorporate those elements into their own work.
- Spotify: To *automate genre tagging* for the new songs added without human intervention. It also helps to *create automated playlists* easily for the users, as Spotify adds 60.000 new tracks every day.

1.3. Dataset Description

The dataset is taken from [Kaggle](#), and from the available set of files we have only used **genre_v2.csv**. A brief summary of the dataset is given below and in Section 1.4. for further analysis of the columns in the data.

Rows	Columns	Target Variable (1)	Categorical Variables (3)	Numerical Variable (11)	Object Variables (7)
42,305	22	genre	Attributes - key, mode and time_signature	Attributes - danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, Unnamed: 20	Attributes - type, id, track_href, song_name, analysis_url, uri, Unnamed: 21

1.4. Column Description^[2]

The features of the dataset can be explained as follows:

- **id**: Spotify ID for the track.
- **uri**: Unique Resource Identifier for any spotify song.
- **track_name**: Title of the track.
- **track_href**: URL for the track.
- **analysis_url**: URL for the track that caters to performing analysis.
- **type**: type of the track, in this dataset - it has only one value of 'audio features'.



- **danceability**: expresses how suitable a track is for dancing, with 0.0 as least danceable and 1.0 as most danceable.
- **energy**: shows a measure of intensity and activity, with measures from 0.0 to 1.0. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.
- **loudness**: gives the overall loudness of a track in decibels (dB).
- **speechiness**: finds the presence of spoken words in a track. For example, recording of poetry, audio books are more speech-like hence the value will be closer to 1.0. Normally, can be discretized into 3 sections:
 - Value below 0.33 - represents music and other non-speech tracks
 - Value between 0.33 and 0.66 - have both music and speech
 - Value above 0.66 - tracks with mostly all spoken words
- **acousticness**: value from 0.0 to 1.0, with 1.0 represents high confidence the track is acoustic.
- **instrumentalness**: represents if the track has vocals. If the value is closer to 1.0, the track has no vocal content. For example, rap/spoken word tracks are vocal.
- **liveness**: identifies if there is any presence of audience in the track. Higher liveness values depict that the tracks were performed live, normally with a value above 0.8.
- **valence**: describes the positiveness conveyed by the track with values from 0.0 to 1.0. Normally, happy or cheerful tracks have high valence and are more positive.
- **tempo**: overall estimated tempo of a track in beats per minute (BPM).
- **duration_ms**: track length in milliseconds.
- **key**: key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C#/D ♭, 2 = D, and so on. If no key was detected, the value is -1.
- **mode**: shows the modality (major - 1 or minor - 0) of a track via which its melodic content is obtained.
- **time_signature**: an estimated time signature. Specifies how many beats are in each bar. Normally, ranges from 3 to 7 indicating time signatures of 3/4 to 7/4, but in this dataset it is from 1 to 5.
- **genre**: genre to which the track belongs.
- **Unnamed 20**: numerical column with random values which is explored further in the section 2.2.2.
- **Unnamed 21**: text column which is explored further in the section 2.2.1.

1.5. Summary of work done

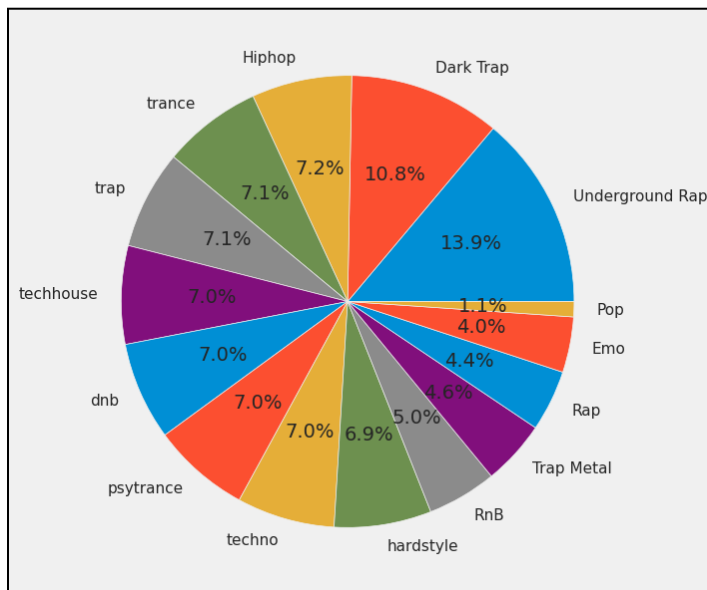
- Through this project, we tried to accomplish several tasks including exploratory data analysis, data preprocessing, feature importance, selection, extraction, and engineering.
- In the exploratory data analysis, the distribution of genres and various numerical and categorical columns is analyzed. Data preprocessing involves handling null values, dropping irrelevant columns, and splitting the data into train and test sets. Feature importance is determined using a Random Forest Classifier, and feature selection is performed based on the importance rankings. Feature extraction is done for the duration column by converting it to minutes. Feature engineering is applied to the speechiness column by discretizing it into three categories.
- After completing the data preparation and feature engineering steps, the dataset is ready for model training and testing. The genre column is encoded into different categories, and speechiness is converted into one-hot vectors. Data scaling may also be applied at this stage.
- The project concludes with model training and testing, where different machine learning algorithms can be applied to classify the genres of songs based on the selected features.
- Overall, this project aims to provide a classification model that can accurately predict the genre of songs based on their attributes, which can be beneficial for music recommendations, research, music production, and automatic genre tagging in streaming services like Spotify.

2. Data Exploration and Pre-processing

2.1. Exploratory data analysis

2.1.1. Target column - Genre

- The total count for each genre in the target class and also the percentage of each class is displayed in the table alongside.
- These values represent that the dataset is nearly fairly balanced in distribution, thus we oversample the classes which are less representative in the later section - 3.3.
- These values represent that the dataset is fairly balanced, since there are 15 different classes with the distribution from a range of 1% to 14%, with an average of 7% which is as expected. It was noticed that the genre pop has 460 songs, while Rap has almost 6k songs.

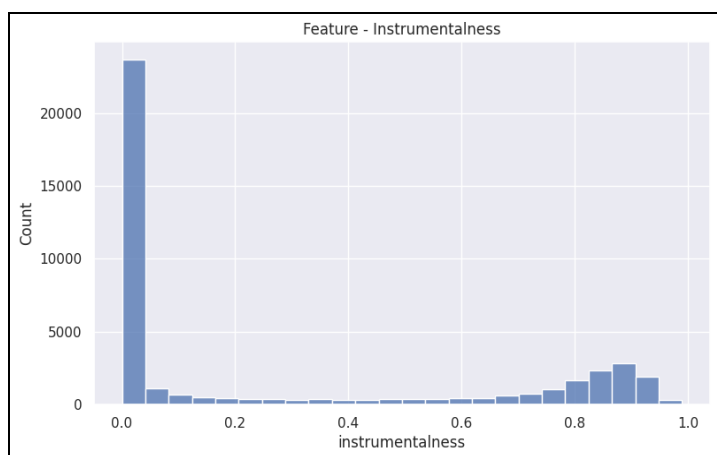
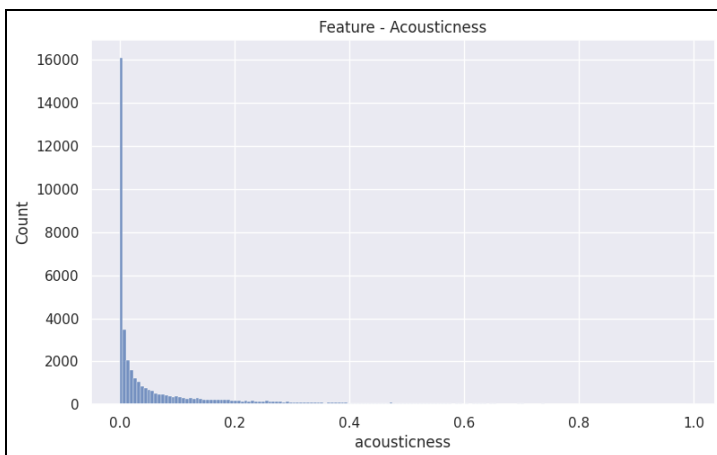
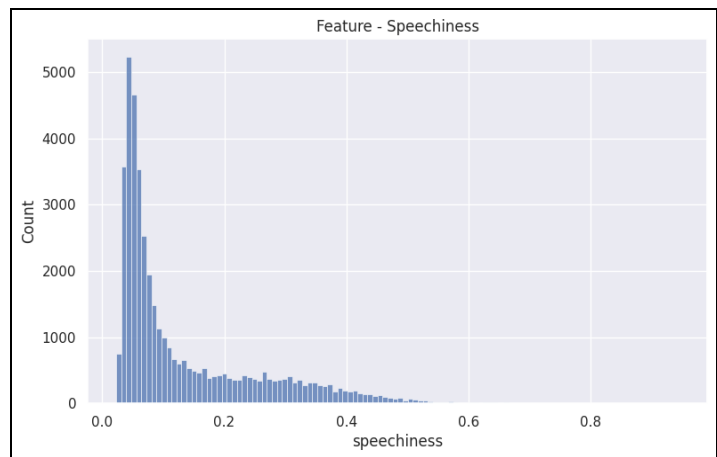
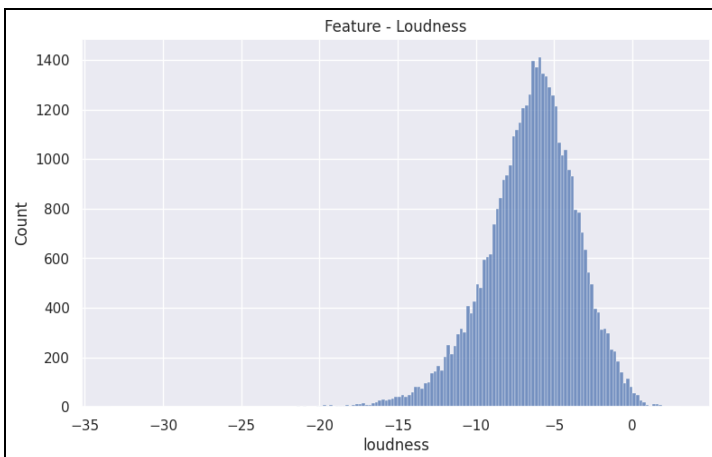
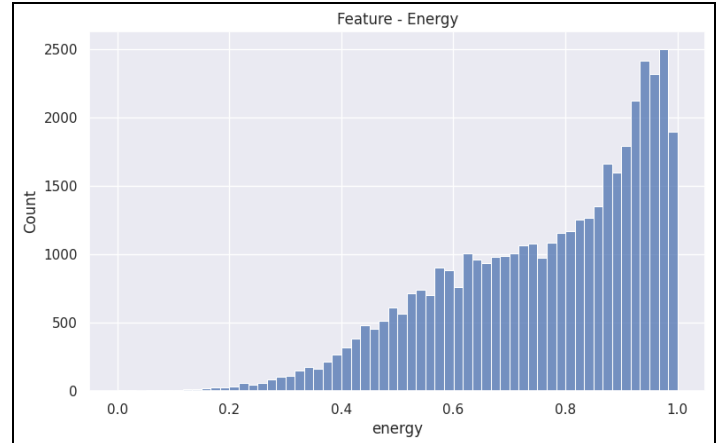
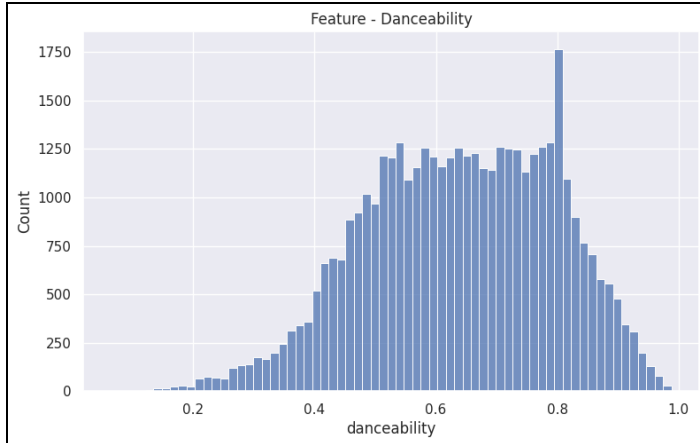


	count	percentage
Underground Rap	5875	13.9%
Dark Trap	4578	10.8%
Hip-hop	3028	7.2%
trance	2999	7.1%
trap	2987	7.1%
techhouse	2975	7.0%
dnb	2966	7.0%
psytrance	2961	7.0%
techno	2956	7.0%
hardstyle	2936	6.9%
RnB	2099	5.0%
Trap Metal	1956	4.6%
Rap	1848	4.4%
Emo	1680	4.0%
Pop	461	1.1%

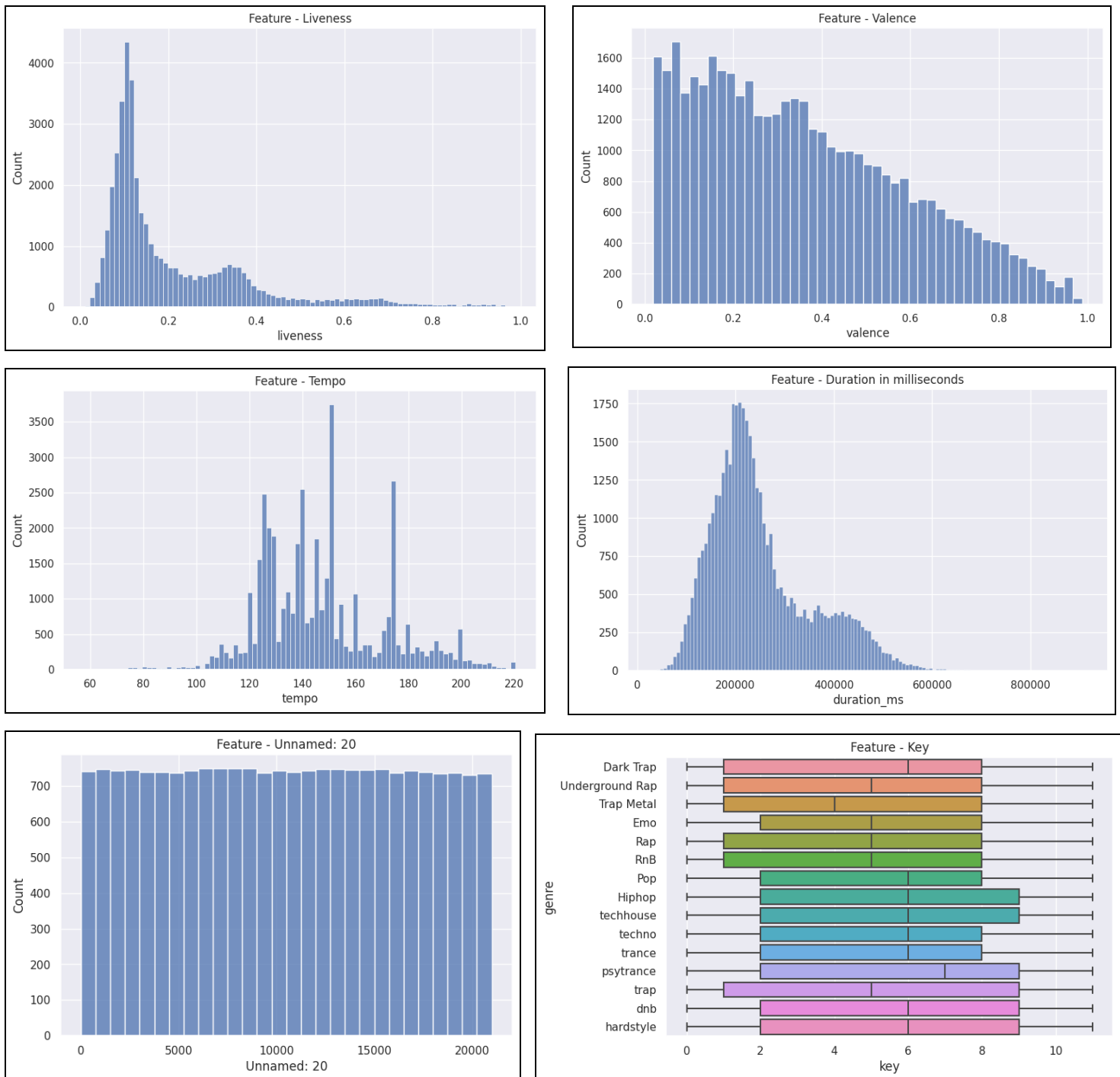
2.1.2. Numerical columns

The columns included in this section are:- 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms'. The following graphs represent the plots of tuple count for each unique value for every numerical column and some observations from these graphs are also noted below:

- Danceability plot seems gaussian but is not exactly a gaussian distribution.
- No specific conclusion can be drawn from the energy plot.
- Loudness distribution seems gaussian.
- Speechiness somewhat represents a chi-square distribution and the same goes for acousticness.
- Instrumentalness has more values as 0, representing more vocal tracks, which is true, as the count of Rap is 5875.

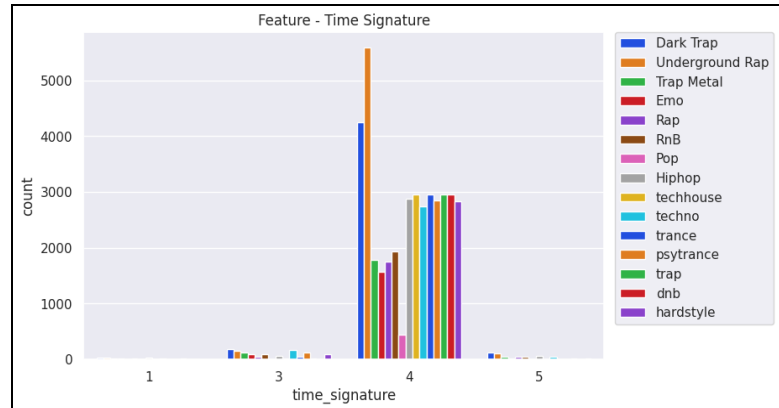
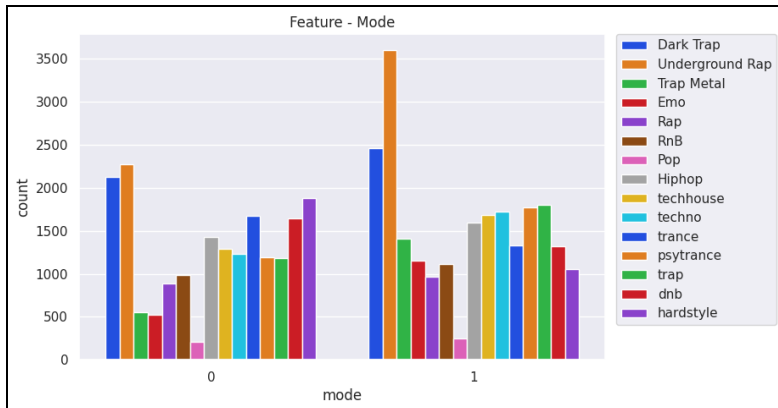


- Valence linearly decreases with most of its value belonging to low, i.e. negative energy - sad, depressed or angry.
- Tempo plot almost seems like a normal distribution.
- Duration_ms has really high values, which shall be catered in the section of feature extraction, i.e. section 2.4.1.
- No specific conclusion can be drawn from the plot of Unnamed: 20, the column is further investigated during pre-processing, i.e. section 2.2.2.



2.1.3. Categorical columns

These include key, mode and time_signature. The following graphs indicate their plots along with some conclusions drawn below:



- There is no significant insight for the columns time_signature, mode and key from the plots. So we are choosing to explore these columns more in the further sections.

2.2. Exploring the features further

2.2.1. 'Unnamed: 21' and 'song_name' columns

- Firstly, for this categorical column of Unnamed: 21, the different categories along with their counts were obtained. For instance: **Euphoric Hardstyle** had a total of **1398** while **HALCYON's 808 BASEMENT** had a count of **12**. On analyzing the data for each category of Unnamed: 21, it was discovered that there was a one-to-one mapping between Unnamed: 21 and genre of the song, i.e. every different category of Unnamed: 21 mapped to one genre, such as '**Euphoric Hardstyle**' mapped to the genre of '**hardstyle**' while '**UKF Drum & Bass - All Uploads**' mapped to the genre of '**dnb**'.
- For the column - song_name, it is an object column (with string values) and identifier of the song.
- Note that both these columns describe the song, but aren't important for model training as such as they are identifiers and will be eventually dropped when preparing the model. But, it is crucial to understand that in our dataset, a song needs a unique identifier (in our case, we have assumed it's the name), to identify that a particular song X, has the tempo value of t1, speechiness of t2 and loudness of t3. This is required to understand what features map to a particular song, which can then further be classified into genre.
- So before starting to inspect these columns further, we identified how many tuples have null in this categorical column of Unnamed: 21.
 - It was found out that out of 42305 tuples, 21525 of Unnamed: 21 have nulls.
 - Also, when checking the column of song_name it was found that out of 42305 rows, 20786 have nulls.
 - Furthermore, it was understood that only 6 tuples have both song_name as well as Unnamed: 21 as null, while for others, either song_name is present or Unnamed: 21.
- Since, any data can be useful to our model, instead of directly dropping it, we decided to impute/drop the values in this column using the following conditions:
 - If both song_name and Unnamed: 21 is null, drop the row. Note that as mentioned in the previous point above only 6 such rows were present.
 - If song_name is null, append Unnamed: 21 with the row number and insert into song_name. E.g. for Unnamed: 21 of **Euphoric Hardstyle** and row number **36411**, we imputed it with a value of **Euphoric Hardstyle 36411**.
 - In all the other cases, the value of song_name is retained.
- After performing the above imputations, the column of 'Unnamed: 21' was dropped.

2.2.2. 'Unnamed: 20' column

Firstly, for this numerical column of Unnamed: 20, it was understood that out of 42305 tuples, 21525 have nulls. The values wherein this column wasn't null were more or less similar to the row numbers. Hence with no relevant information to be extracted from this column, this column was decided to be dropped.

2.2.3. 'id', 'uri', 'track_href', 'analysis_url' columns

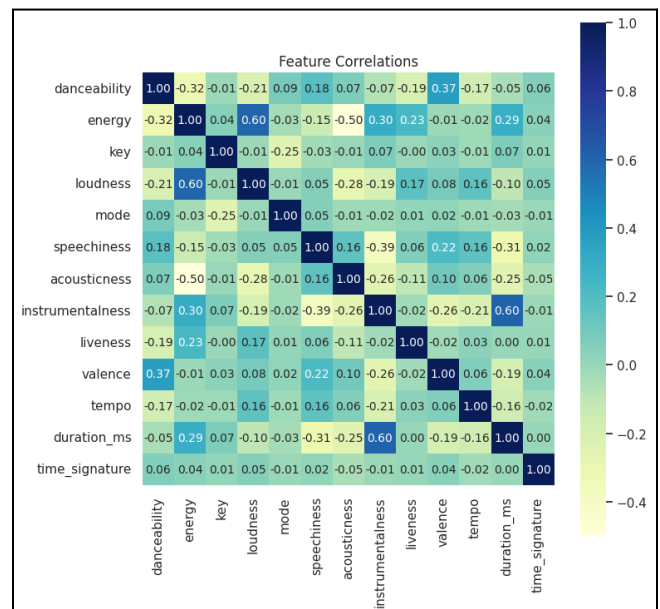
For these object columns, the values were all in strings. Since, we considered only song_name as our unique identifier of the song, we decided to drop these extra identifiers as they won't add any useful information to our model.

2.2.4. 'type' column

For this column "type", all the values belong to only one category - audio_features, i.e. all of 42305 values correspond to the same value of 'audio_features', hence this column is not important for our classification model and was dropped.

2.3. Splitting the data into train-test

Before even splitting the data, we are dropping the column song_name as it won't be useful for training/testing. Thus, after completing the tasks until 2.2, we are left with 42299 tuples and 14 columns. We now split the dataset into train and test, before performing any further features related pre-processing. The split ratio between train and test set is assumed to be 80% and 20%, hence the training dataset now has 33839 rows and 13 features, while the test dataset has 8460 rows and 1 target feature.



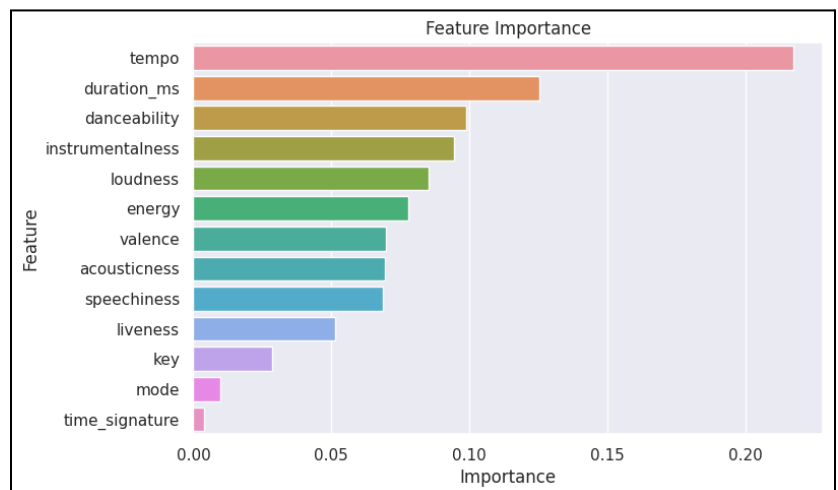
2.4. Feature Importance, Selection, Extraction & Engineering

2.4.1. Feature Importance

Since the train dataset has many features as we can see in the adjoining figure, we chose to fit a standard Random Forest Classifier and use the model to identify the important features as it's very good in doing that with its ensembling (bagging) technique and cross-entropy/gini calculations.

2.4.2. Feature Selection

According to the figure on the right, it was understood that not all features hold much importance. Thus, we decided to run the model for the top 7 features and then top 10 features. Since, there was not much difference between them, we chose to retain the top 10 features, thereby dropping the features - key, mode and time_signature from our train dataset.



2.4.3. Feature Extraction

The column `duration_ms` represents the song duration in milliseconds. As per the figures on the left, when checking the statistics for this column, the minimum duration obtained was 25600 ms while the maximum was 913052 ms. Since all the other columns of the dataset are in fairly lower ranges like between 0 to 1 for speechiness, etc., we decided to reduce the range of this column by converting the duration in milliseconds to duration in minutes, for which we divided the value by 60000 (1 minute = 60 seconds and 1 second = 1000 milliseconds). After extracting the `duration_min` feature, we noticed the minimum value as 0.427 minutes while the maximum as 15.218 minutes.

```
count    33839.000
mean     251207.508
std      103233.156
min       25600.000
25%      180000.000
50%      224800.000
75%      302482.500
max       913052.000
Name: duration_ms, dtype: float64
```

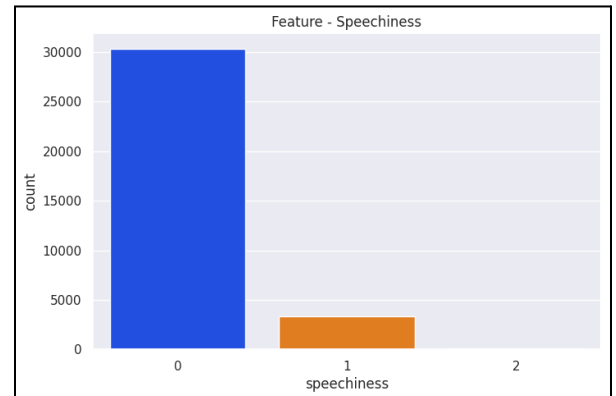
```
count    33839.000
mean         4.187
std          1.721
min          0.427
25%          3.000
50%          3.747
75%          5.041
max          15.218
Name: duration_min, dtype: float64
```

2.4.4. Feature Engineering

We know that speechiness can be discretized into 3 sections:

- Value below 0.33 - represents music and other non-speech tracks (0)
- Value between 0.33 and 0.66 - have both music and speech (1)
- Value above 0.66 - tracks with mostly all spoken words (2)

speechiness	count
0	30369
1	3349
2	121

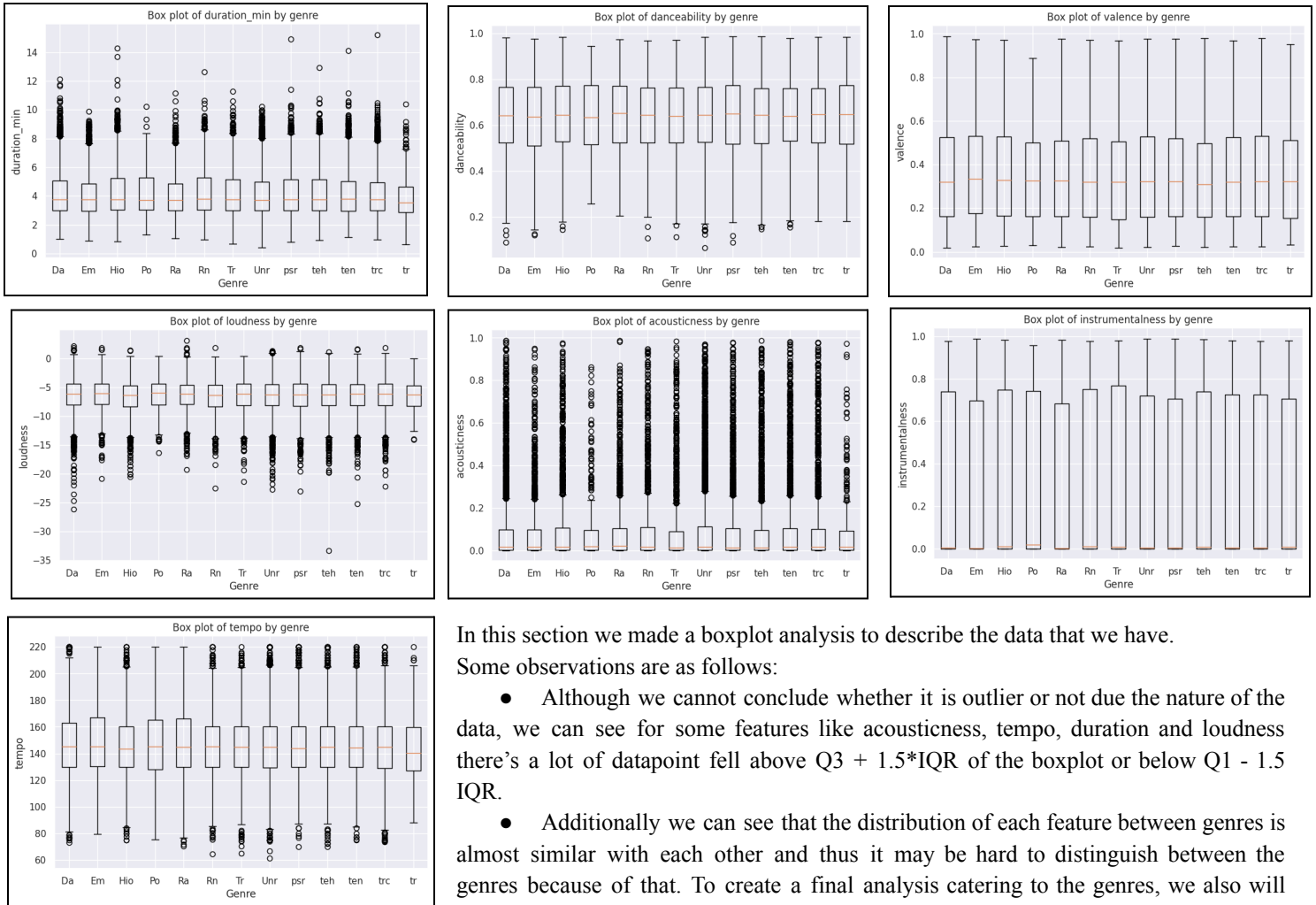


As it can be seen from the figures besides, the values of `speechiness = 0` are the most dominant with a count of approximately 30k in the training data, while `speechiness = 2` with the least dominance of 121 tuples.

Now after performing the above pre-processing steps, our dataset summary looks like below:

Rows	Columns	Target Variable (1)	Categorical Variables (0)	Numerical Variable (11)	Object Variables (0)
42,299	11	genre	-	Attributes - danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_min	-

2.5. Feature Analysis and Visualisation



3. Model Training and Testing

3.1. Data Duplication

Thinking logically, one track can be classified into multiple genres, i.e. one-to-many classification (multi-label) as in the figure below.

	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	genre
425	0.705	0.648	-10.467	0.141	0.005	5.950e-04	0.373	0.398	130.037	155350	Dark Trap
5115	0.705	0.648	-10.467	0.141	0.005	5.950e-04	0.373	0.398	130.037	155350	Underground Rap
10742	0.705	0.648	-10.467	0.141	0.005	5.950e-04	0.373	0.398	130.037	155350	Trap Metal

In this project, we are catering to classifying one track into one genre (one-to-one classification), even though the training has multiple genres. But, for tuples which are truly duplicate (as shown in the figure below), meaning all the features including genre are the same, we are dropping those. The count for such tuples was obtained to be 2850 in the training dataset, thus the size of the training set reduces to 30989 after this step.



	tempo	danceability	instrumentalness	loudness	energy	valence	acousticness	speechiness	liveness	genre	duration_min	is_duplicate
766	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	False
779	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	True
14422	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	True
20382	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	True
23028	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	True
25991	174.046	0.588	0.002	-1.779	0.896	0.112	0.002	0	0.072	dnb	4.171	True

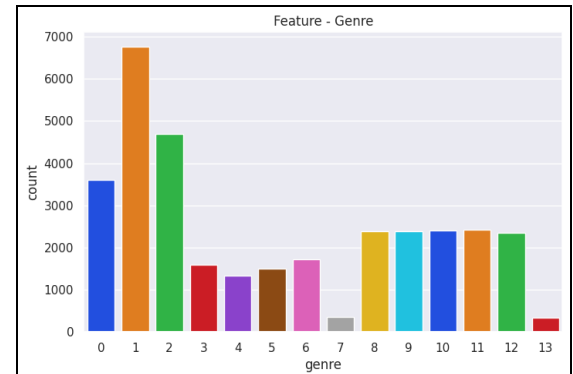
3.2. Data Preparation

3.2.1. Genre column

For this variable, we enumerate the genre into its different categories. Note that, since train-test split was randomly done, only 14 out of the 15 (original df) genre classes are present in the train set.

3.2.2. Speechiness column

Similarly, speechiness is converted to one-hot vectors by encoding it for its different categories. The data frame thus obtained looks like the sample figure alongside.



3.2.3. Transformations: Boxcox and Logarithmic

We tried transformations on the numerical columns which represented almost gaussian distribution (e.g., speechiness, valence, etc.) - including logarithmic and box-cox to handle from skewness of data, but it was not much helpful for the model accuracy, hence it is not included in the final code.

speechiness_0	speechiness_1	speechiness_2
1.0	0.0	0.0
1.0	0.0	0.0
1.0	0.0	0.0
1.0	0.0	0.0
1.0	0.0	0.0

3.2. Data Scaling

As discussed before, our variables are showing different ranges, so we need to scale them so as not to interfere with our models' performance. Note that, the scaling process is implemented differently for training and testing sets to let the testing set treat the training test as a black box and also not use any information from the test set for training out model. Since, for our data, tempo and duration_min show varying ranges (unlike other features) & are gaussian in nature (as discussed before), we apply standard scaler to them.

3.3. Oversampling (Upsampling)

Since, for some classes, the distribution isn't proper, upsampling is performed and all the classes have equal distribution. This helps us to introduce diversity and avoid overfitting while addressing the class imbalance problem. For this project we will be using the technique of SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples that are similar to the existing minority class instances but not exact duplicates.

4. Model Implementations

For this problem statement, we have implemented the following models with a fixed set of hyper-parameters (usually used), instead of fine-tuning them to find optimal values:

- Decision Tree Classifier



- Random Forest Classifier
- XGBoost
- KNeighborsClassifier
- Gaussian Naive Bayes
- Logistic Regression
- Quadratic Linear Discriminant

Furthermore, we have used cross-validation for training and testing our models to know how good our model is. This is done as we cannot use the training data for evaluating the model because it can produce artificially good results. Thus, by introducing optimistic bias, providing fair model comparison and detecting model overfitting (if any), cross validation helps in model evaluation, like in this case. We have used the number of cross-folds as 5.

Even with a very limited computation power on our laptops as well as Google Colaboratory, we tried running hyper-parameter tuning for XGBoost model for the search space of

```
'learning_rate': [0.6, 0.7, 0.8],  
'gamma': [0, 0.01],  
'max_depth': np.arange(2, 10, 2).astype(int)
```

And the optimal model was found at: **'learning_rate': 0.6, 'gamma': 0 and 'max_depth': 8**. But, as compared to the pre-defined hyperparameters used for XGBoost, the model didn't show much improvement for test data. Nonetheless, it was important for us to understand that it is a good approach / method to identify the optimal parameters specific to our dataset. Also, the training error for XGBoost when hyperparameter tuning is 0.97 (as shown in section 5), i.e., it is overfitting. Thus, to avoid overfitting to the training data, we can further tune it by adding regularization parameters of alpha (for L1 regularization) and lambda (for L2 regularization).

5. Comparison of Results

We analyze our model performance by using F1 Macro metrics. The reason for this being since we have an unbalanced dataset, F1 Macro score will be a better reflective measure to evaluate our model performance.

Algorithm	Decision Tree	Random Forests	XGBoost (with optimal hyperparameters)	XGBoost	KNN Classifier	Gaussian Naive Bayes	Logistic Regression	Quadratic Linear Discriminant
Train F1 Macro	0.652	0.714	0.97	0.733	0.718	0.416	0.518	0.612
Test F1 Macro	0.565	0.606	0.619	0.633	0.527	0.383	0.498	0.497

There is not much significant gap between the training and testing f1 macro scores for our models, which indicates that our models are able to generalize better to the training data and they perform better on unseen data, i.e., robust in nature. Also, the models of Gaussian Naive Bayes, Logistic Regression and QDA perform much worse than other models used.

6. Final Chosen Model

Based on the result we got in the previous section, we chose XGBoost as our chosen model. We also noticed that XGBoost outperformed the other models quite significantly, moreover for non-tree based models. After selecting the final model, the results were evaluated using a confusion matrix to produce a deeper insight to our classification result & a proper conclusion is thus derived.

7. Conclusions

- We can see that there are two clusters of genres, one is the genres that are easily distinguishable and the one that gets misclassified to the other genre frequently. Genres such as dnb, hardstyle, psytrance, tech house, techno, trance and trap more or less got classified correctly to their classes. We can see that even though it got misclassified, the amount is quite insignificant.
- On the other hand we can see the rest of the genres like Dark Trap, Emo, Hip-hop, Pop, Rap, RnB and Trap Metal frequently got misclassified as Underground Rap, whereas Underground Rap itself also got misclassified to the other genres too although the amount is not that significant. We can also see that the Pop genre is the hardest genre to classify as it is mostly being misclassified to the other genre, mostly Underground Rap and RnB.

predicted target	Dark Trap	Emo	Hip-hop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Dark Trap	461	8	19	0	5	19	47	293	14	15	3	8	6	43	29
Emo	7	224	13	4	0	37	1	29	5	10	0	0	1	6	4
Hip-hop	39	23	264	3	13	75	7	202	4	1	0	3	0	1	5
Pop	4	14	13	4	0	35	1	29	0	1	0	7	0	2	2
Rap	13	2	22	1	96	29	6	171	1	1	0	3	0	0	2
RnB	19	35	76	3	4	138	1	97	1	1	0	2	0	1	2
Trap Metal	63	13	5	0	0	4	116	152	1	3	0	0	0	3	8
Underground Rap	115	14	120	4	10	50	69	778	2	4	0	7	0	3	9
dnb	3	4	6	0	0	0	0	0	574	0	0	0	0	0	0
hardstyle	3	4	0	0	0	0	1	0	0	546	8	0	0	0	28
psytrance	7	0	1	0	0	0	0	0	0	6	557	0	12	21	4
techhouse	5	0	0	0	0	1	2	2	0	0	0	548	29	8	0
techno	2	0	0	0	0	0	0	0	0	0	13	43	451	34	0
trance	10	2	1	0	0	0	0	0	0	0	22	4	20	523	2
trap	29	6	2	0	0	1	7	5	0	66	3	0	1	5	485


- In conclusion, Underground Rap dominated the classification result and for the other genres especially Dark Trap, Emo, Hip-hop, Pop, Rap, RnB and Trap Metal, it is likely for them to get classified as Underground Rap music. In the future, clustering algorithms might help us to detect which genre is similar to each other and reduce the number of classes we need to classify. Instead of doing classification on the genre we can do it based on the characteristics of the genre itself, for instance we can say Dark Trap and Underground Rap are similar with each other and we can put them in one class using our clustering technique.

8. Futurework and limitations

- In this project, we are classifying one track into one genre (one-to-one or single label classification) instead of multi-label (one-to-many) classification. An extension of our project can be to achieve multi-label classification by choosing probabilistic naive bayes classifiers.
- Due to limited GPU on our systems as well as on Google Collaboratory, we couldn't successfully run the hyperparameter tuning. Hence, we have used the following algorithms with pre-defined hyper parameters, instead of running for a wide range of parameters until we obtain an optimal parameter.
- In addition to this, implementing complex algorithms based on neural networks can better fit and model the data.
- Thus, as a future scope, performing further optimization of algorithmic parameters, along with incorporating multi-label classification can help us achieve better prediction results.



9. References

- [1] A. Samoshyn, “Dataset of songs in Spotify,” Kaggle, <https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify> (accessed Jun. 8, 2023).
- [2] MaharshiPandya, “ spotify tracks dataset,” Kaggle, <https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset> (accessed Jun. 8, 2023).