Data Mining, Big Data, and Analytics - CMPS451

Spring 2024

Team 2
⟨Key, Value⟩Knights

Ahmed Mohamed Saad	1190184
Hazem Montasser	2200003
Ali Hassan Khalaf	2200011

Table of Contents

1. Introduction

In the ever-expanding landscape of digital music, the ability to accurately classify songs into their respective genres is not only a fundamental task but also a challenging endeavor. With the exponential growth of music streaming platforms and digital libraries, there is an increasing demand for robust and efficient methods to organize and categorize the vast amounts of available music.

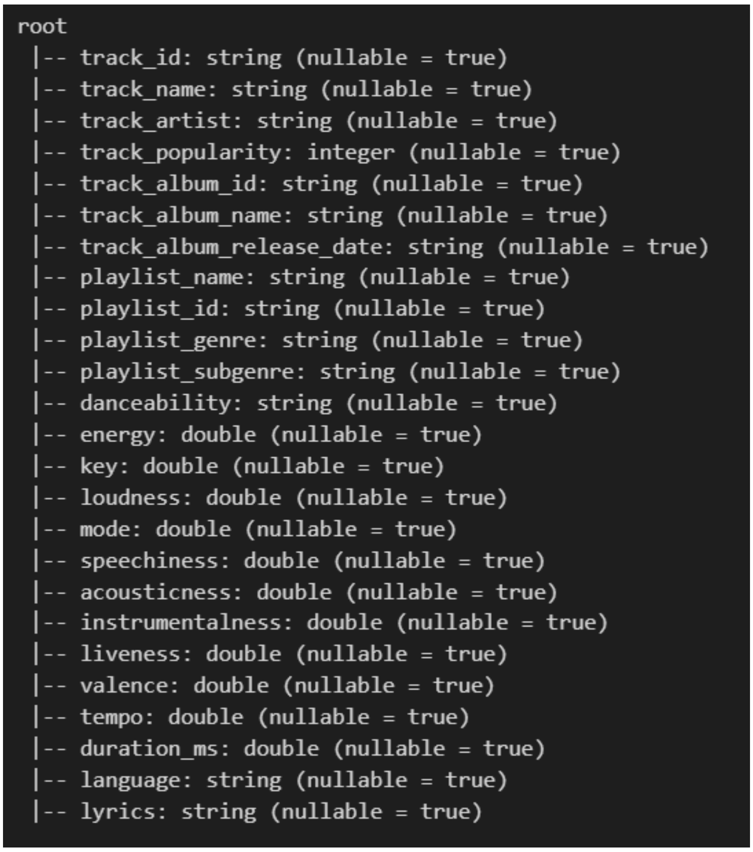
This project aims to leverage the power of big data analytics to tackle the complex task of song genre classification. Unlike traditional approaches that rely solely on manual tagging or simplistic rule-based systems, our approach harnesses the wealth of information embedded within the audio features and lyrical content of songs.

2. Datasets used

- ○ Audio features and lyrics of Spotify songs (kaggle.com)
- ○ 10+ M. Beatport Tracks / Spotify Audio Features (kaggle.com)

3. Data Preprocessing

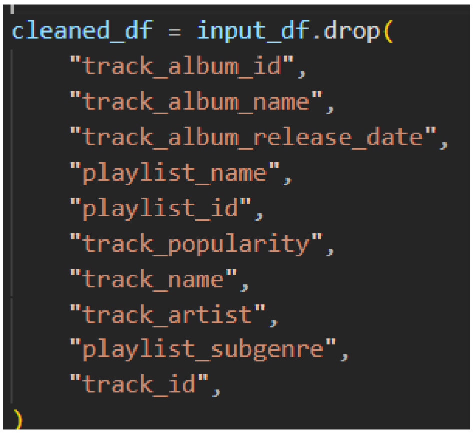
3.1 Dataset Original Schema



```
root
 |-- track_id: string (nullable = true)
 |-- track_name: string (nullable = true)
 |-- track_artist: string (nullable = true)
 |-- track_popularity: integer (nullable = true)
 |-- track_album_id: string (nullable = true)
 |-- track_album_name: string (nullable = true)
 |-- track_album_release_date: string (nullable = true)
 |-- playlist_name: string (nullable = true)
 |-- playlist_id: string (nullable = true)
 |-- playlist_genre: string (nullable = true)
 |-- playlist_subgenre: string (nullable = true)
 |-- danceability: string (nullable = true)
 |-- energy: double (nullable = true)
 |-- key: double (nullable = true)
 |-- loudness: double (nullable = true)
 |-- mode: double (nullable = true)
 |-- speechiness: double (nullable = true)
 |-- acousticness: double (nullable = true)
 |-- instrumentalness: double (nullable = true)
 |-- liveness: double (nullable = true)
 |-- valence: double (nullable = true)
 |-- tempo: double (nullable = true)
 |-- duration_ms: double (nullable = true)
 |-- language: string (nullable = true)
 |-- lyrics: string (nullable = true)
```

:

3.2 Remove Unnecessary Columns



```
cleaned_df = input_df.drop(
    "track_album_id",
    "track_album_name",
    "track_album_release_date",
    "playlist_name",
    "playlist_id",
    "track_popularity",
    "track_name",
    "track_artist",
    "playlist_subgenre",
    "track_id",
)
```

3.3 Remove records with null values

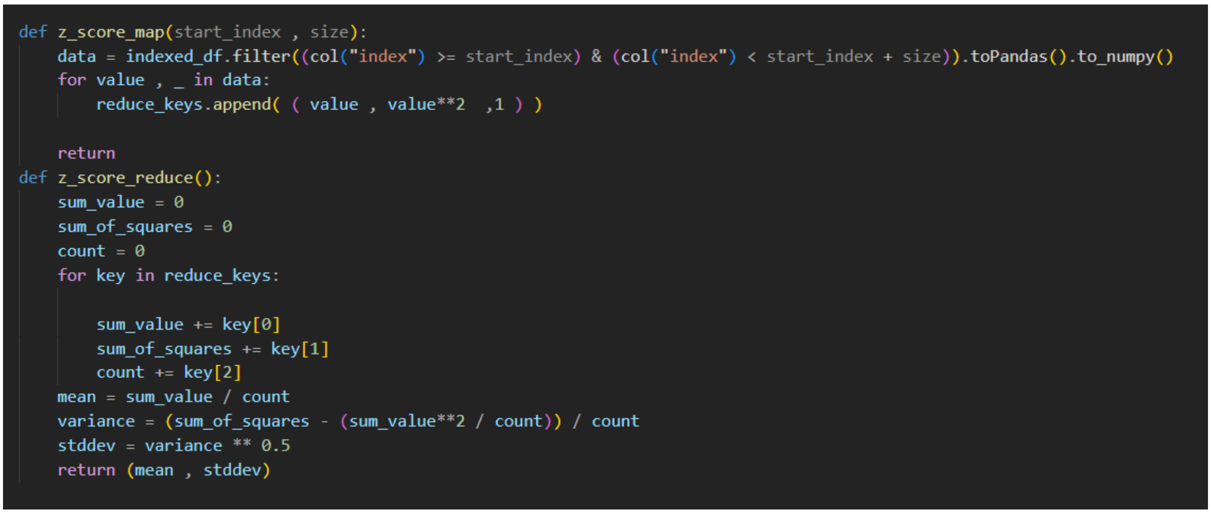
3.4 make sure that all features except genre , lyrics have numeric values

3.5 Normalize numeric Features using Map-Reduce

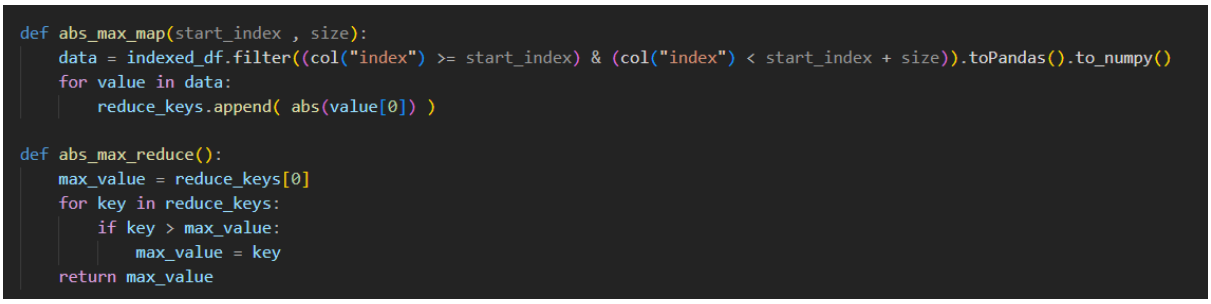
We Have 2 map-Reduce Stages:

First One takes a feature values and compute sum , sum of squares and count Then in the reduce stage these values are used to get mean , std-deviation of each feature to perform z-score normalization

Second one Computes the absolute maximum value for each feature after applying z-score normaliztion then this value is used to make the range of our values from -1 to 1 [-1 , 1]



```
def z_score_map(start_index , size):
    data = indexed_df.filter((col("index") >= start_index) & (col("index") < start_index + size)).toPandas().to_numpy()
    for value , _ in data:
        reduce_keys.append( ( value , value**2  ,1 ) )

    return
def z_score_reduce():
    sum_value = 0
    sum_of_squares = 0
    count = 0
    for key in reduce_keys:

        sum_value += key[0]
        sum_of_squares += key[1]
        count += key[2]
    mean = sum_value / count
    variance = (sum_of_squares - (sum_value**2 / count)) / count
    stddev = variance ** 0.5
    return (mean , stddev)
```



```
def abs_max_map(start_index , size):
    data = indexed_df.filter((col("index") >= start_index) & (col("index") < start_index + size)).toPandas().to_numpy()
    for value in data:
        reduce_keys.append( abs(value[0]) )

def abs_max_reduce():
    max_value = reduce_keys[0]
    for key in reduce_keys:
        if key > max_value:
            max_value = key
    return max_value
```

3.6 Lyrics preprocessing

The preparation of the lyrics for the model involves several steps to transform raw text data into a format suitable for training and embedding in the model. Here's a brief description:

1. **Tokenization:**

 - The lyrics are first tokenized using `RegexTokenizer` from PySpark. This process involves splitting the raw text into individual words or tokens based on a specified pattern (`pattern="\\W"`), which typically separates words based on non-word characters like spaces, punctuation, etc.

2. **Stopwords Removal:**
   - Stopwords (commonly occurring words like "the", "is", "and", etc.) are removed from the tokenized lyrics to focus on more meaningful words.
3. **Text Cleaning:**
   - Text cleaning steps like lowercase conversion, removing special characters, or handling irregularities in the text data can be performed depending on the specific preprocessing needs of the dataset.

4. **Sequence Padding:**

- After tokenization, the lyrics are converted into sequences of integers (word indices) using Keras `Tokenizer`. This involves fitting the `Tokenizer` on the tokenized lyrics to create a vocabulary and then converting the tokenized lyrics into sequences of integers based on this vocabulary.

5. **Padding Sequences:**

- The sequences of integers (representing tokenized lyrics) are padded to ensure uniform length (`max_len`) across all sequences. This is achieved using `pad_sequences` from Keras, which pads sequences shorter than `max_len` and truncates sequences longer than `max_len`.

6. **Embedding Initialization:**

- An embedding matrix (`embedding_matrix`) is created based on pre-trained word embeddings (GloVe embeddings). Each word in the vocabulary is associated with a pre-trained embedding vector, which initializes the embedding layer in the model. This step ensures that the model starts with meaningful word representations learned from large text corpora.

7. **Model Input Preparation:**

- Finally, the tokenized and padded lyrics (`X_train_lyrics_padded`, `X_val_lyrics_padded`) are used as inputs to the model along with the corresponding audio features (`X_train_audio`, `X_val_audio`). These inputs are fed into the model during training (`model.fit`) to jointly learn from both text and audio modalities for the classification task.

In summary, the preparation of the lyrics involves tokenization, sequence padding, embedding initialization with pre-trained embeddings, and integration into the model.

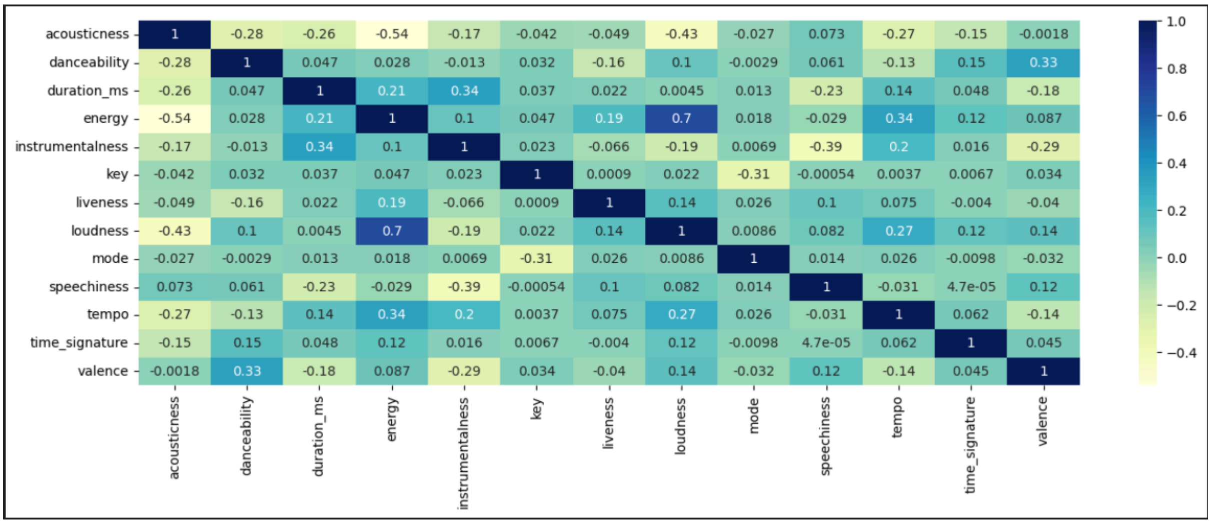
(Note: we used 2 datasets the first one was only 18k songs and it had bad distribution of genres one genre had 1500 songs while others had 4000) so it didn’t yield good results at classification so we decided to use another dataset with the same schema but larger in size ,
The new dataset is 120k songs with 6 genres 20k song each)

The following EDA is done on the new Dataset

4. EDA

4.1 Cross-Correlation Matrix

We tried to draw a cross-correlation matrix between all the numeric features to understand how well they correlate and check if we can remove any oth them if they have strong correlation with each other

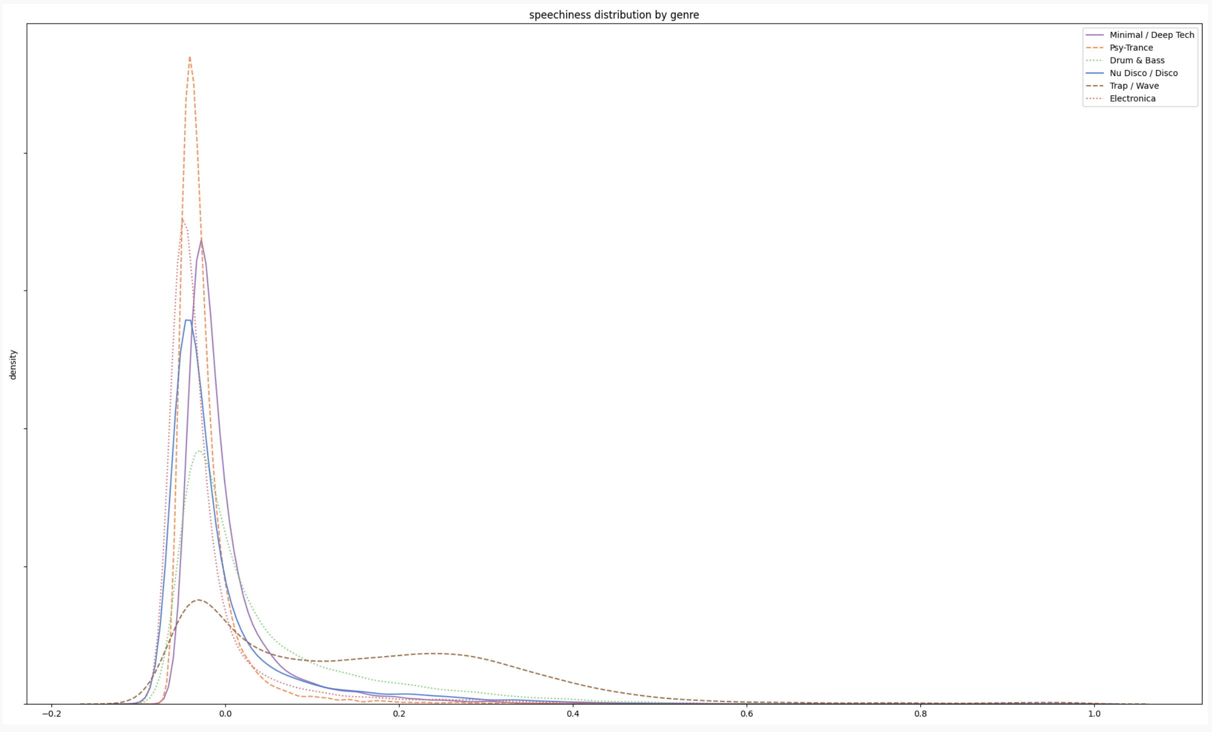


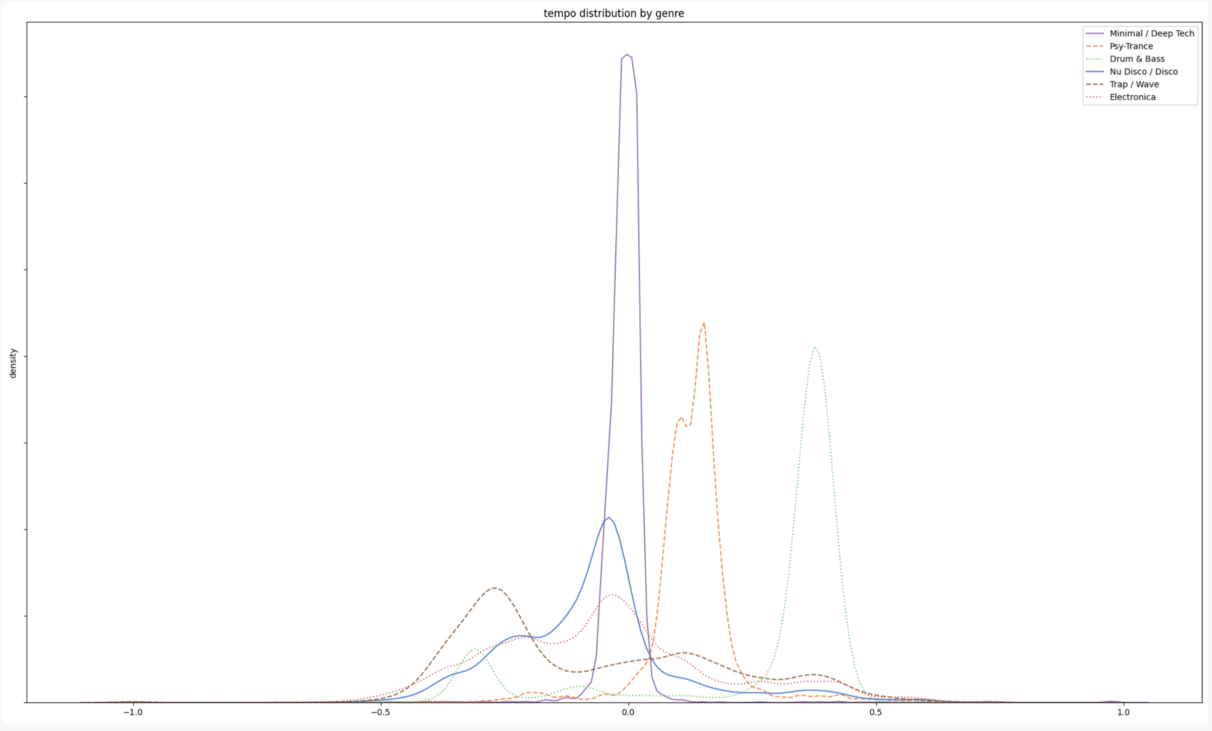
	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
acousticness	1	-0.28	-0.26	-0.54	-0.17	-0.042	-0.049	-0.43	-0.027	0.073	-0.27	-0.15	-0.0018
danceability	-0.28	1	0.047	0.028	-0.013	0.032	-0.16	0.1	-0.0029	0.061	-0.13	0.15	0.33
duration_ms	-0.26	0.047	1	0.21	0.34	0.037	0.022	0.0045	0.013	-0.23	0.14	0.048	-0.18
energy	-0.54	0.028	0.21	1	0.1	0.047	0.19	0.7	0.018	-0.029	0.34	0.12	0.087
instrumentalness	-0.17	-0.013	0.34	0.1	1	0.023	-0.066	-0.19	0.0069	-0.39	0.2	0.016	-0.29
key	-0.042	0.032	0.037	0.047	0.023	1	0.0009	0.022	-0.31	-0.00054	0.0037	0.0067	0.034
liveness	-0.049	-0.16	0.022	0.19	-0.066	0.0009	1	0.14	0.026	0.1	0.075	-0.004	-0.04
loudness	-0.43	0.1	0.0045	0.7	-0.19	0.022	0.14	1	0.0086	0.082	0.27	0.12	0.14
mode	-0.027	-0.0029	0.013	0.018	0.0069	-0.31	0.026	0.0086	1	0.014	0.026	-0.0098	-0.032
speechiness	0.073	0.061	-0.23	-0.029	-0.39	-0.00054	0.1	0.082	0.014	1	-0.031	4.7e-05	0.12
tempo	-0.27	-0.13	0.14	0.34	0.2	0.0037	0.075	0.27	0.026	-0.031	1	0.062	-0.14
time_signature	-0.15	0.15	0.048	0.12	0.016	0.0067	-0.004	0.12	-0.0098	4.7e-05	0.062	1	0.045
valence	-0.0018	0.33	-0.18	0.087	-0.29	0.034	-0.04	0.14	-0.032	0.12	-0.14	0.045	1

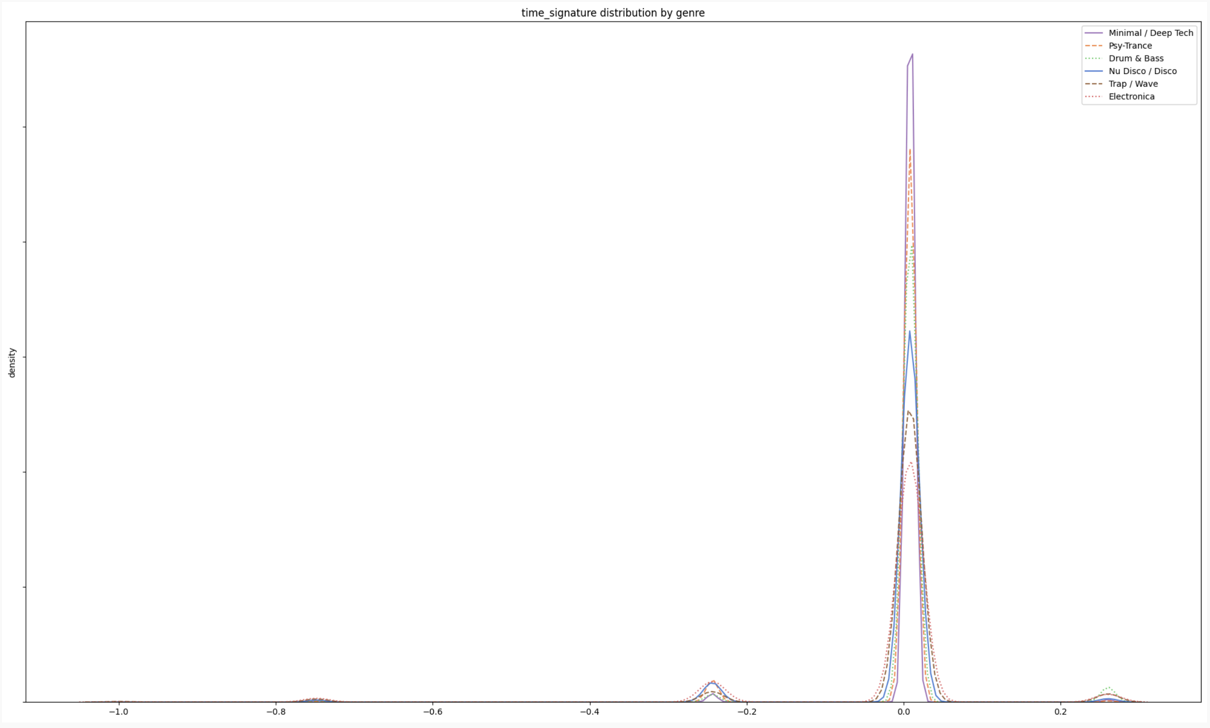
The only two features that had a kind of correlation where loudness and energy with positive correlation of **0.7** But we decided to leave the two features as the they aren't so strong correlated **(>0.8).**

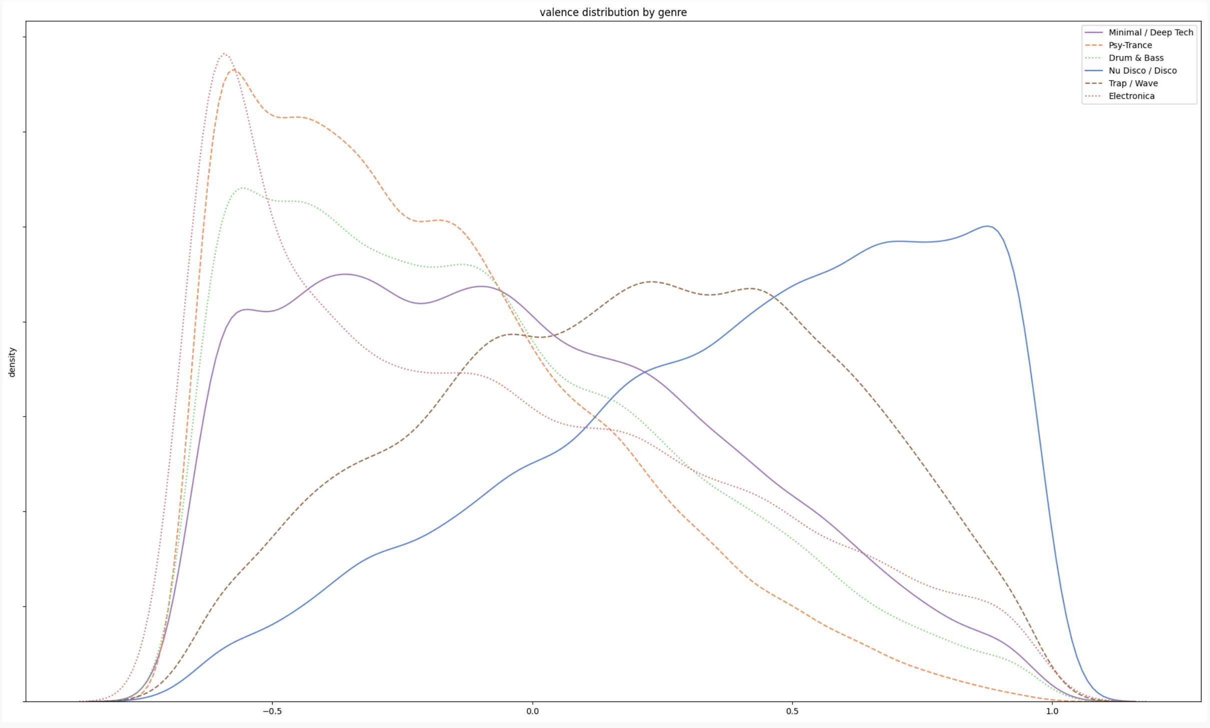
4.2 KDE (Kernel Density Estimate) plot per feature

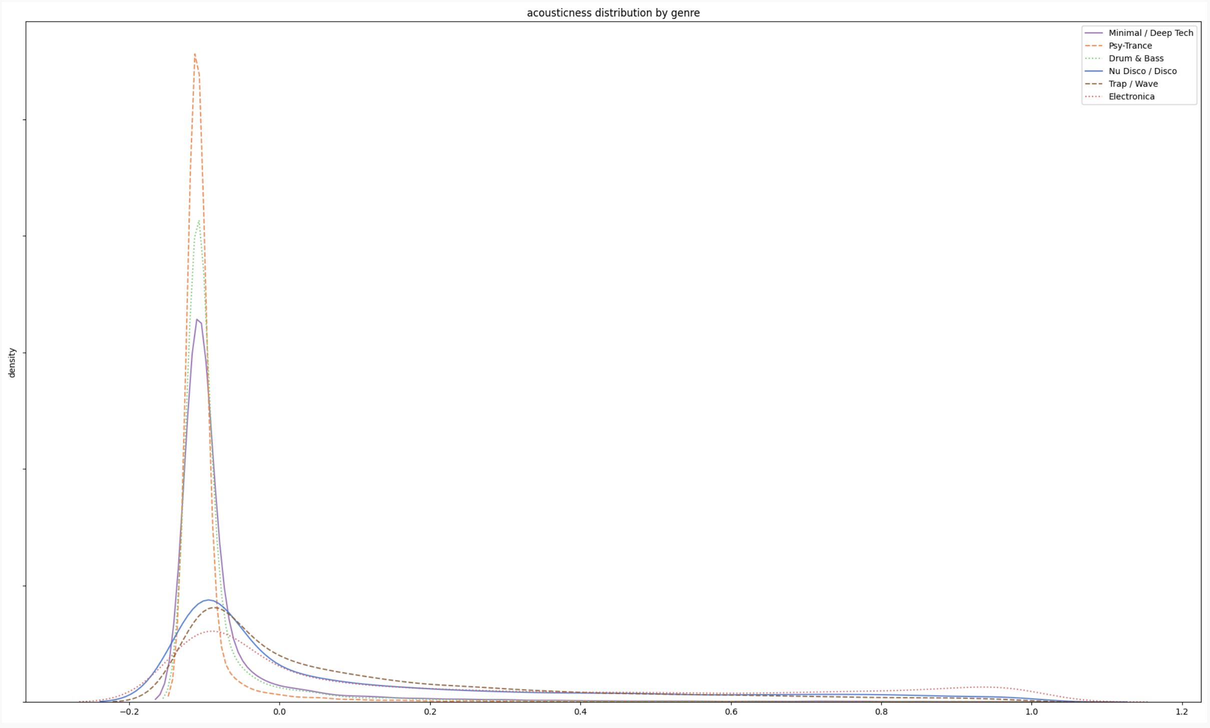
To further understand our numeric features according to each genre we plotted the Distribution of genres according to each feature

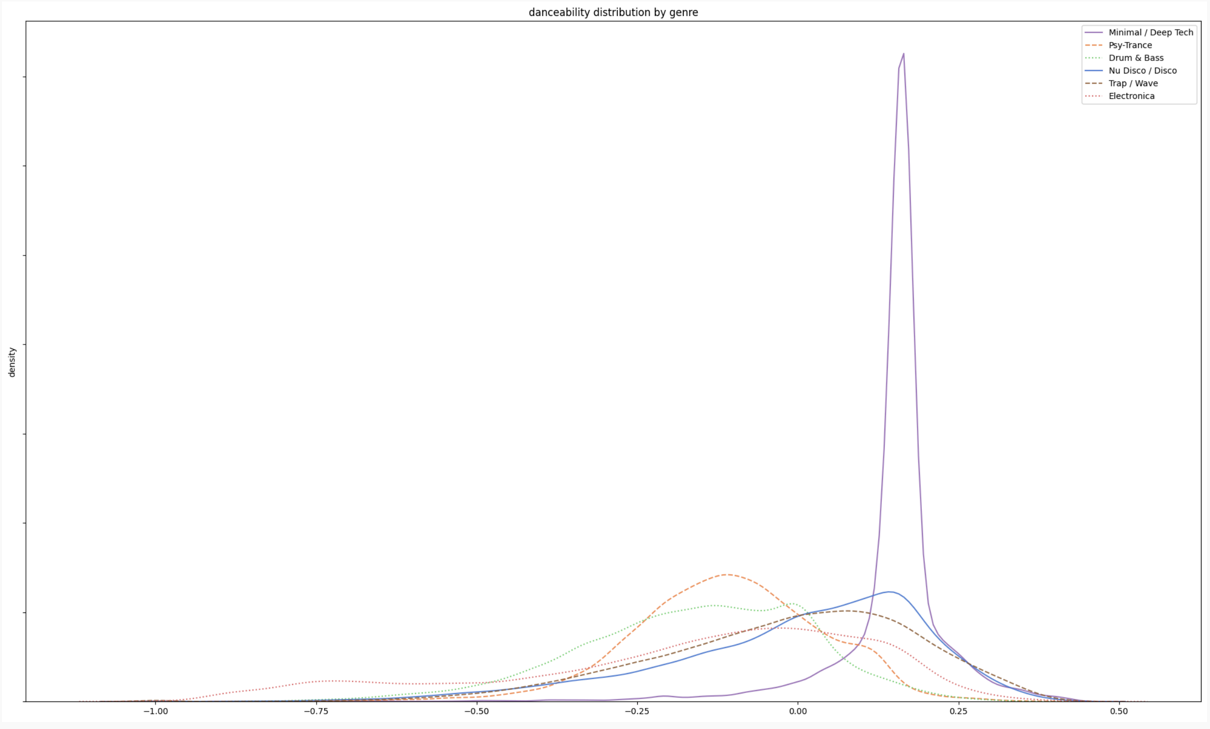

speechiness distribution by genre
density
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

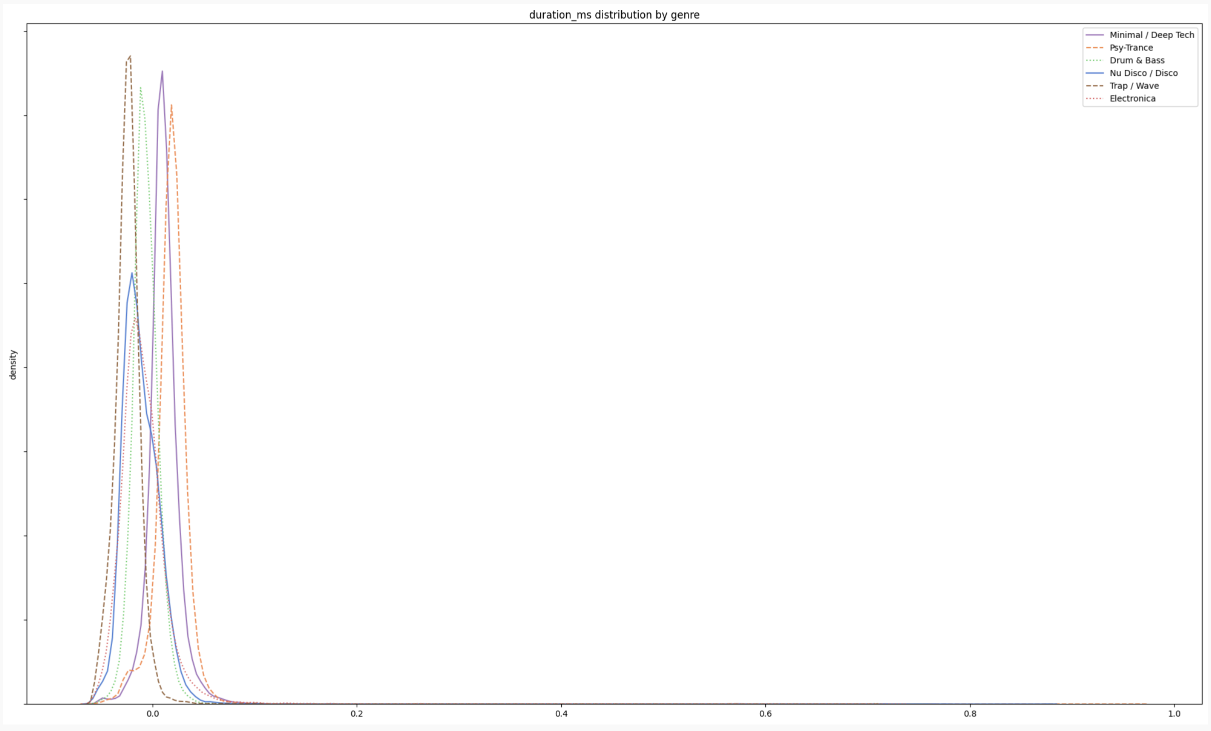

tempo distribution by genre
density
−1.0
−0.5
0.0
0.5
1.0
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

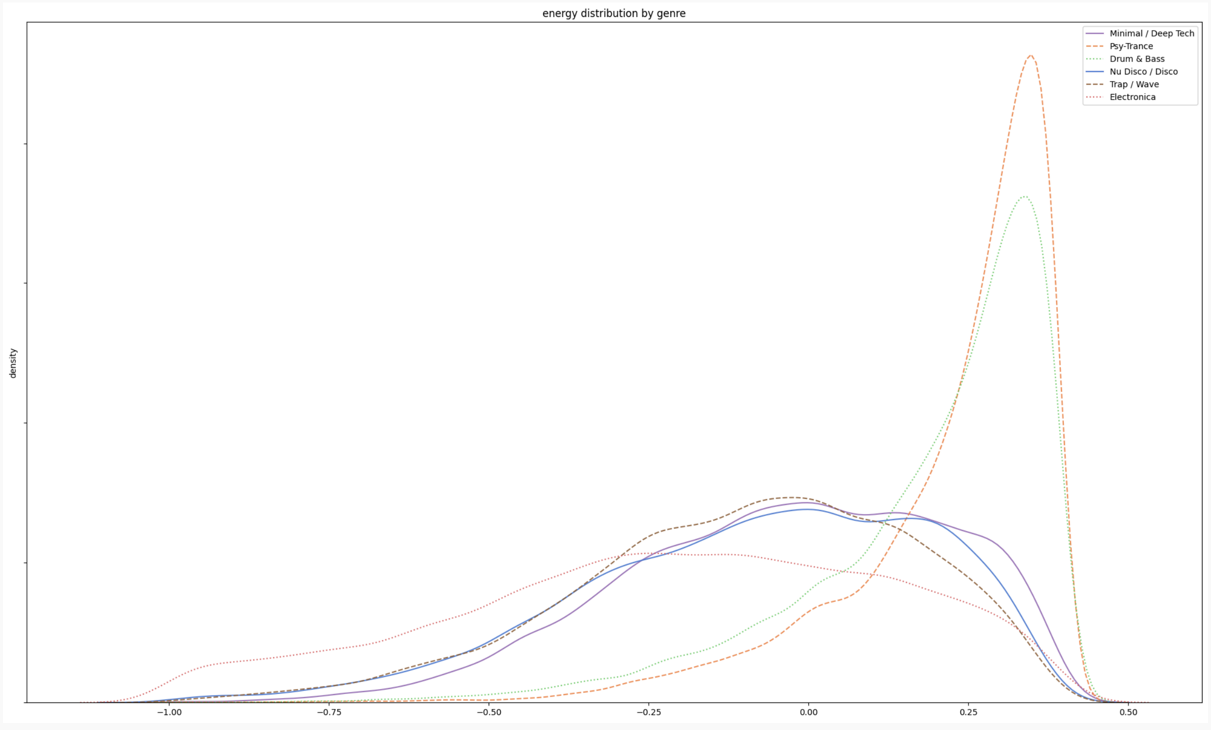

time_signature distribution by genre
density
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

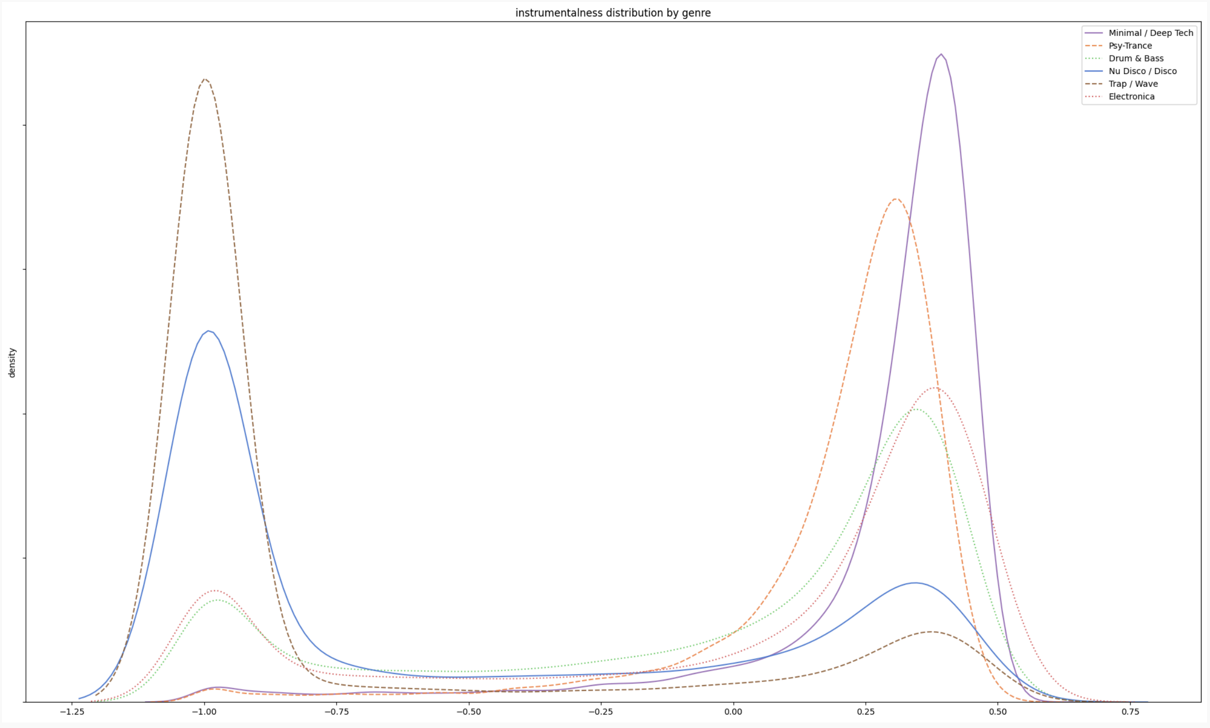

valence distribution by genre
density
−0.5
0.0
0.5
1.0
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

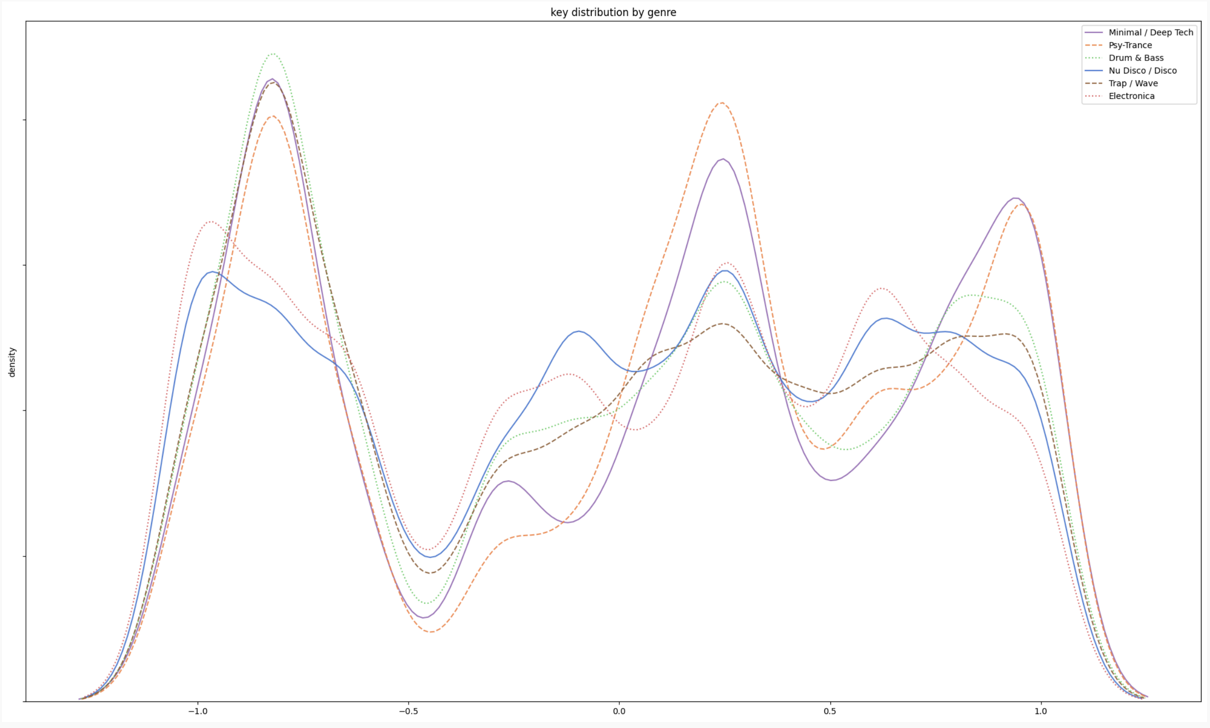

acousticness distribution by genre
density
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

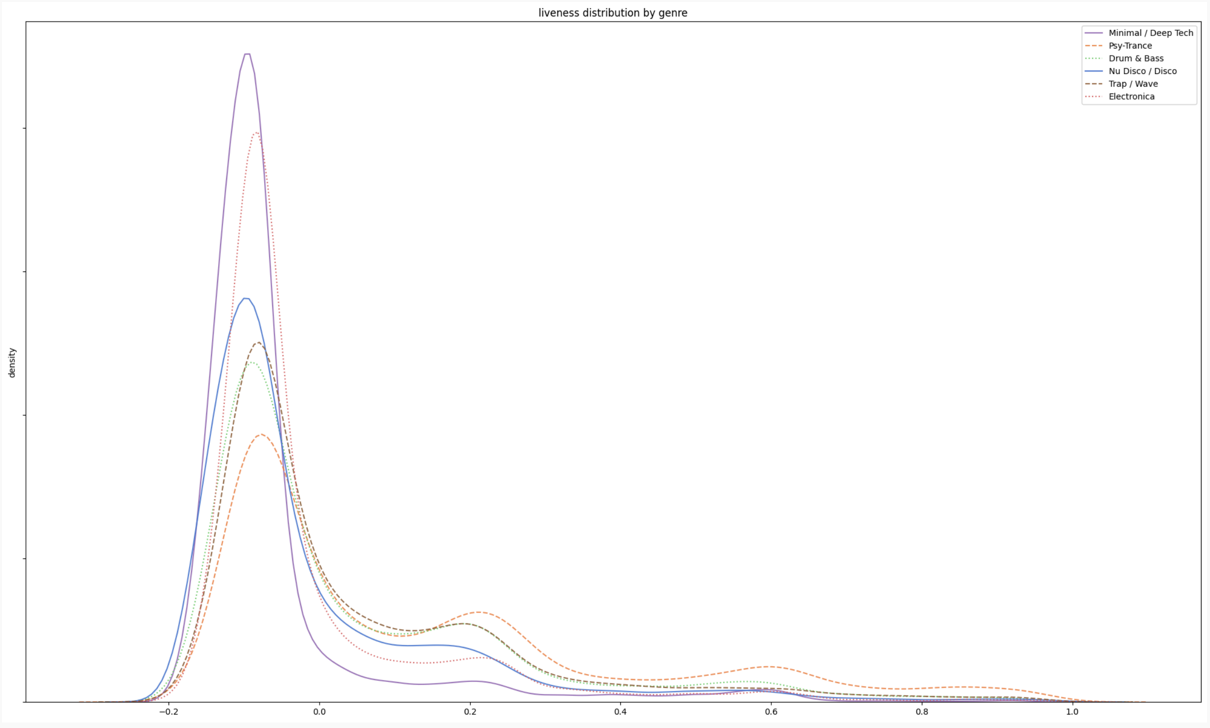

danceability distribution by genre
density
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica

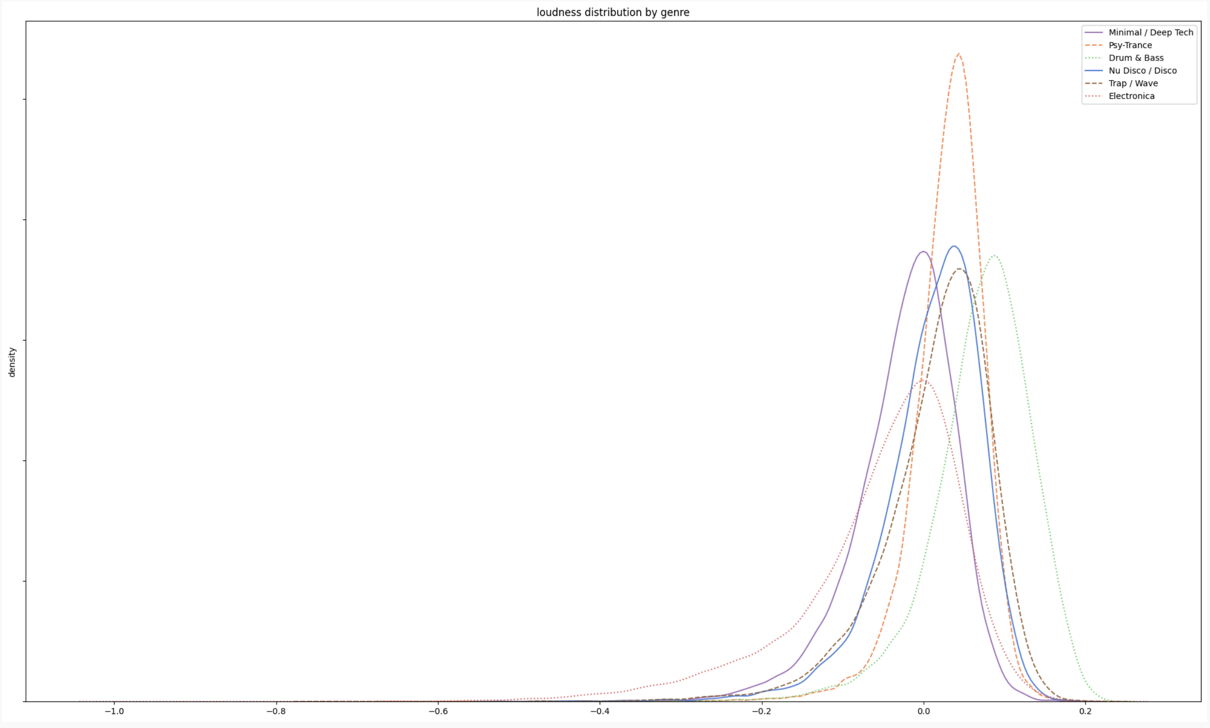

duration_ms distribution by genre
density
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica
0.0
0.2
0.4
0.6
0.8
1.0

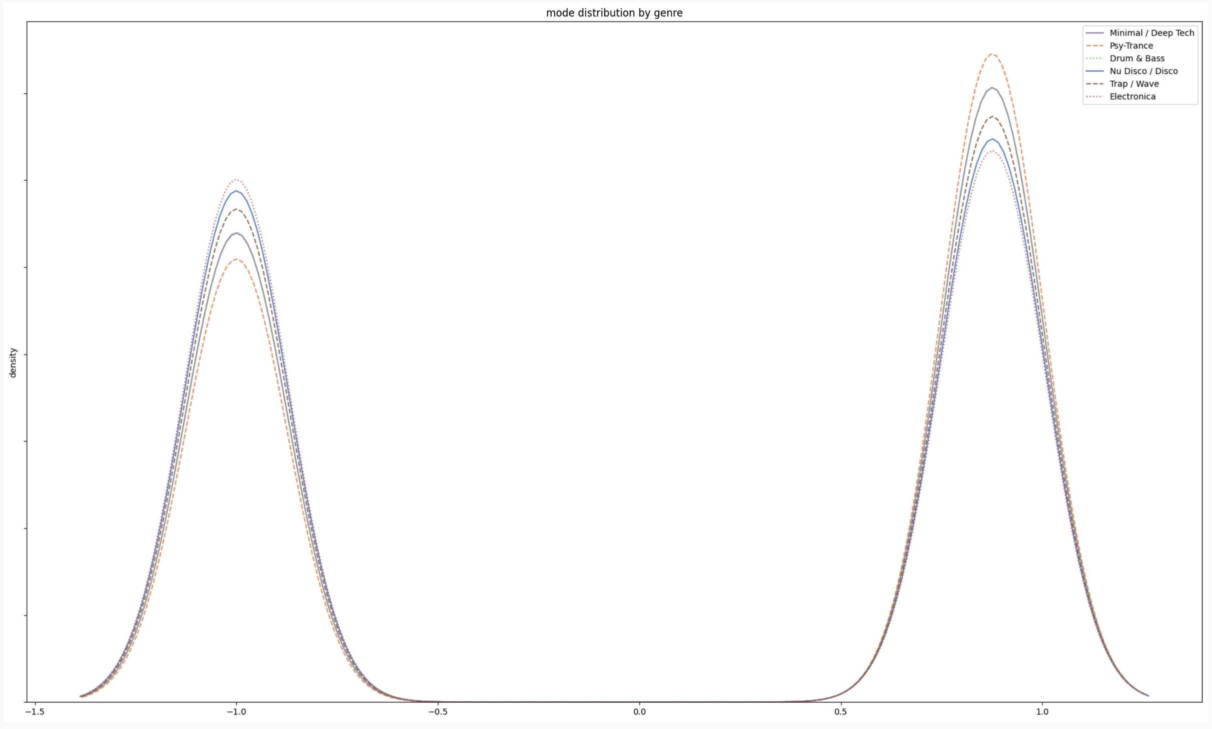

energy distribution by genre
density
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50


instrumentalness distribution by genre
density
−1.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica


key distribution by genre
density
−1.0
−0.5
0.0
0.5
1.0
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica


liveness distribution by genre
density
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


loudness distribution by genre
density
Minimal / Deep Tech
Psy-Trance
Drum & Bass
Nu Disco / Disco
Trap / Wave
Electronica
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2



In Summary we found that some features have nearly no effect on the classification process because the distribution of genres in these features is nearly identical:

These Genres Are : Duration , time_signature and mode
When trying to remove these genres from our classification process they affected only the Random forest classifier by 2%

5. Methodology

5.1 Overview of Model Approaches

This section outlines two approaches designed to classify song genres, each leveraging different sets of features and architectures to achieve the task.

5.2 Neural Networks Based Classifiers

5.2.1 Model 1: Hybrid Text and Audio Model

Architecture:

- Textual Input Branch: This branch handles the textual lyrics, which are converted into sequences of integer tokens. These tokens are then passed through an embedding layer initialized with pre-trained word embeddings. This layer remains static during training to capitalize on existing semantic knowledge. Following this, a bidirectional LSTM layer processes the sequences to capture contextual information from the lyrics in both forward and backward directions.
- Audio Input Branch: Simultaneously, audio features extracted from the songs are fed directly into the model. These features do not undergo preliminary transformations, maintaining the raw audio data integrity.
- Concatenation and Processing: Outputs from both branches are merged and processed through several dense layers with ReLU activation functions, which facilitate learning of intricate relationships between the combined features.
- Output: A softmax layer concludes the architecture, delivering a probability distribution over predefined song genres.

Training:

- The model uses the Adam optimizer and categorical cross-entropy loss, with a focus on maximizing accuracy.

5.2.2 Model 2: Audio-Only Model

Architecture:

- Single Input Branch: This model solely processes audio features, which are represented as vectors of floating-point values capturing various acoustic properties of the songs.

- Dense Layers: The audio vectors pass through a series of dense layers, beginning with 128 units followed by 64 units, both employing ReLU activation. These layers aim to extract and learn hierarchical patterns that are indicative of different music genres.
- Output: The architecture culminates in a dense softmax layer that predicts genre categories based on the processed audio features.

Training:

- Similar to the LSTM model, this model utilizes the Adam optimizer and categorical cross-entropy loss, aiming for high accuracy in genre classification.

5.3 Classic Classification Classifiers

Classification Models Used Only for audio features:

We used 4 different classification models to try to classify our songs using only audio features , **all this models were trained on 80% of the data and testing was done on the other 20%**

5.3.1 Random Forest Classifier

parameters used**(numTrees=200, maxDepth=20)**

5.3.2 DecisionTreeClassifier

parameters used**(maxDepth=30 , maxBins=128)**

5.3.3 Logistic Regression classifier

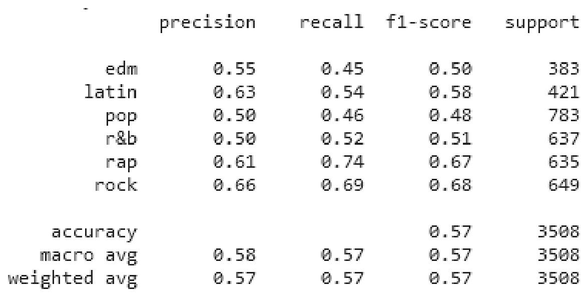
parameters used**(maxIter=100000, regParam=0, elasticNetParam=0)**

5.3.4 SVM (Support Vector Machine)

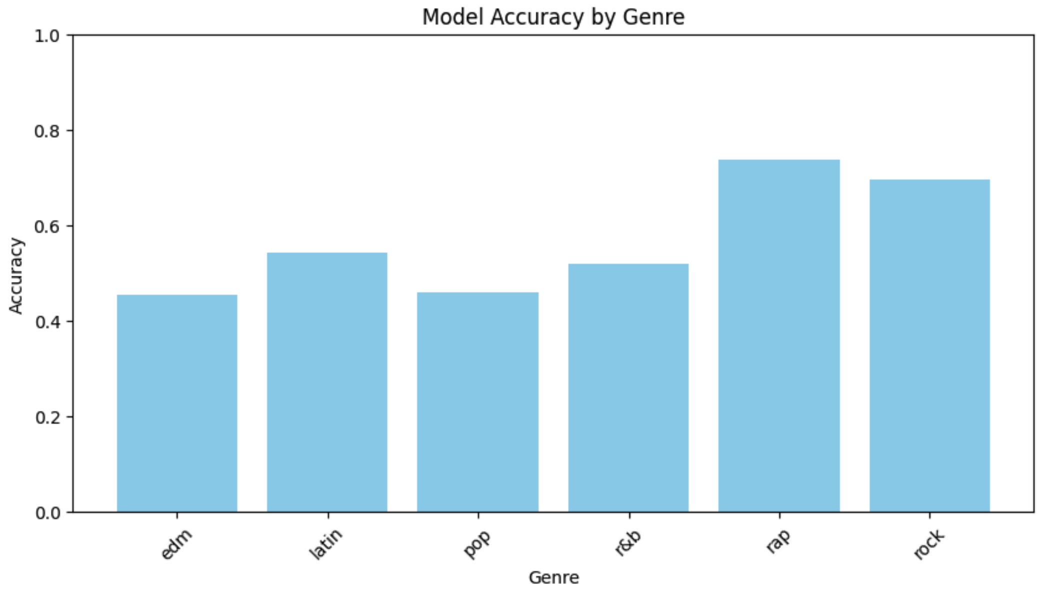
parameters used**(kernel="rbf" , break_ties=True)**

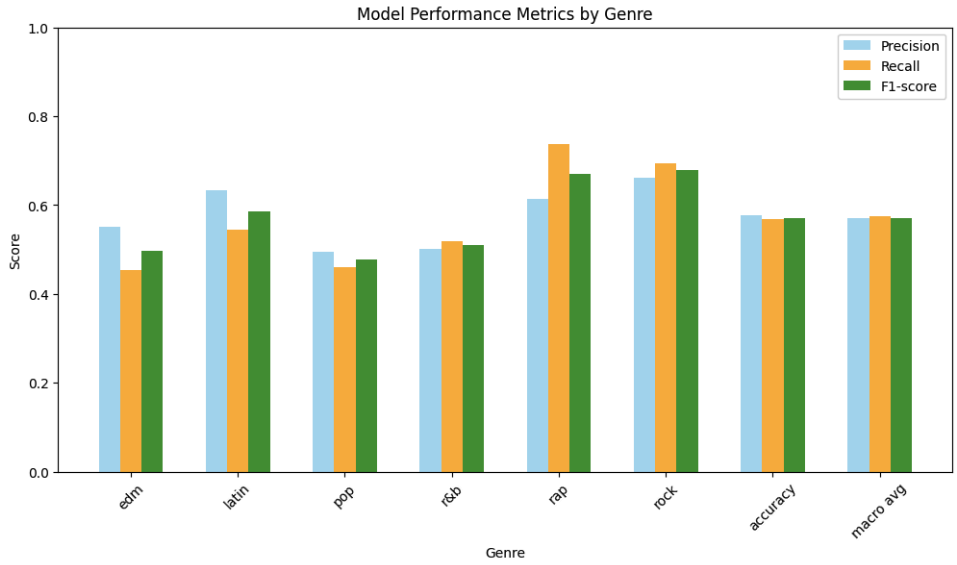
6. Results

Model 1: Hybrid Text and Audio Model



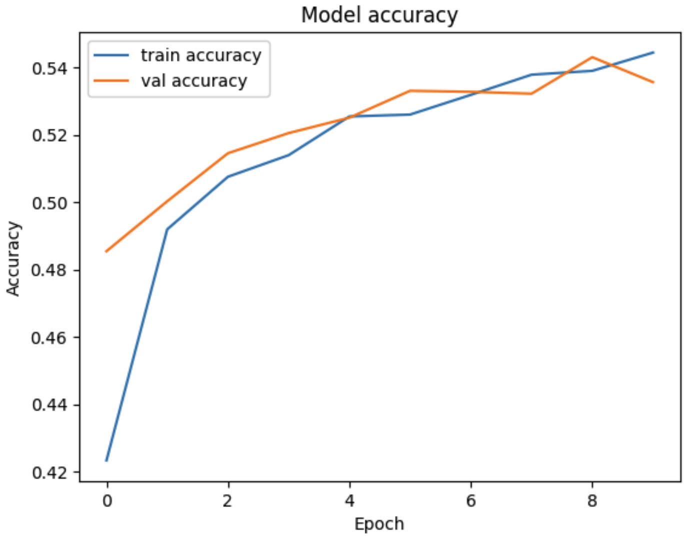
```
              precision    recall  f1-score   support

         edm       0.55      0.45      0.50       383
       latin       0.63      0.54      0.58       421
         pop       0.50      0.46      0.48       783
         r&b       0.50      0.52      0.51       637
         rap       0.61      0.74      0.67       635
        rock       0.66      0.69      0.68       649

    accuracy                           0.57      3508
   macro avg       0.58      0.57      0.57      3508
weighted avg       0.57      0.57      0.57      3508
```

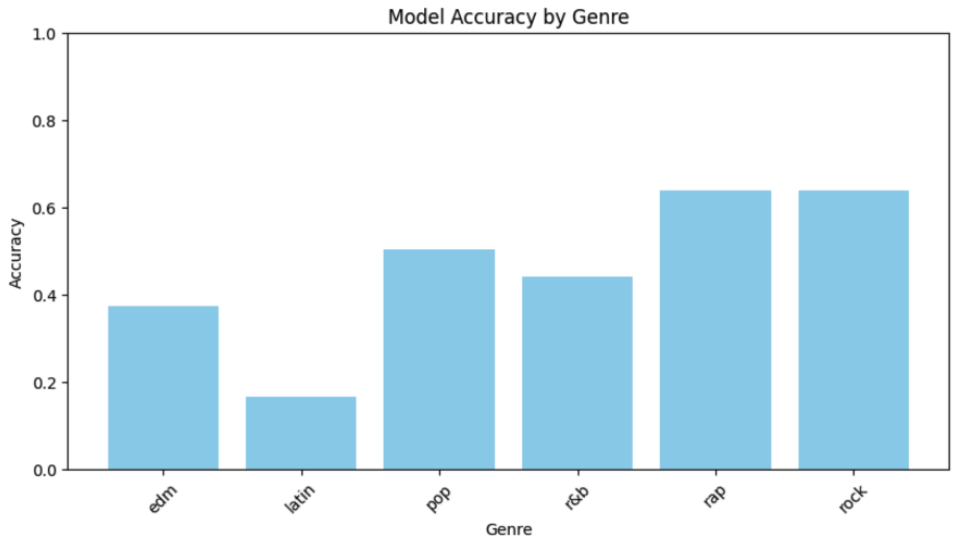


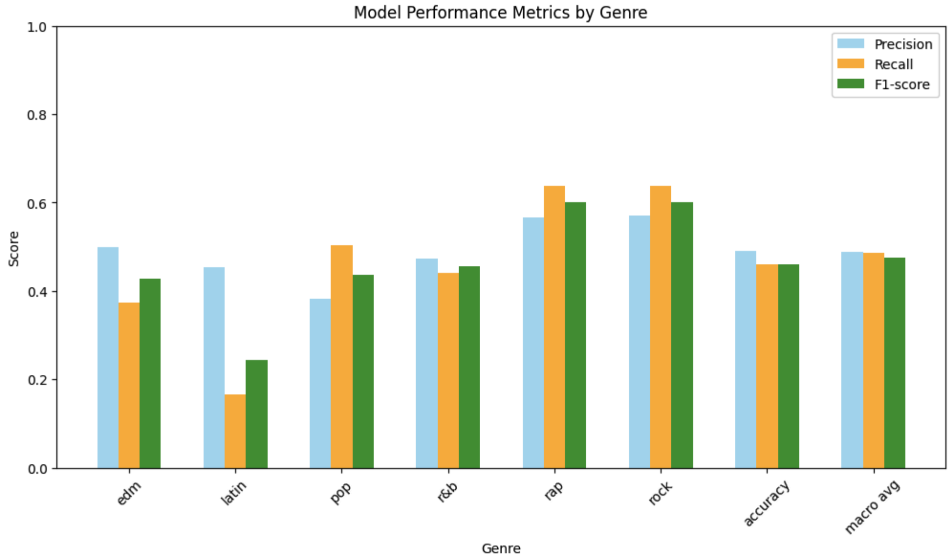


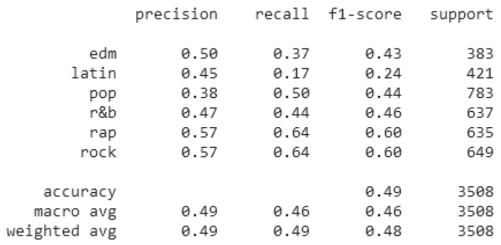
AUC Score: 0.8519

Model 2: Audio-Only Model





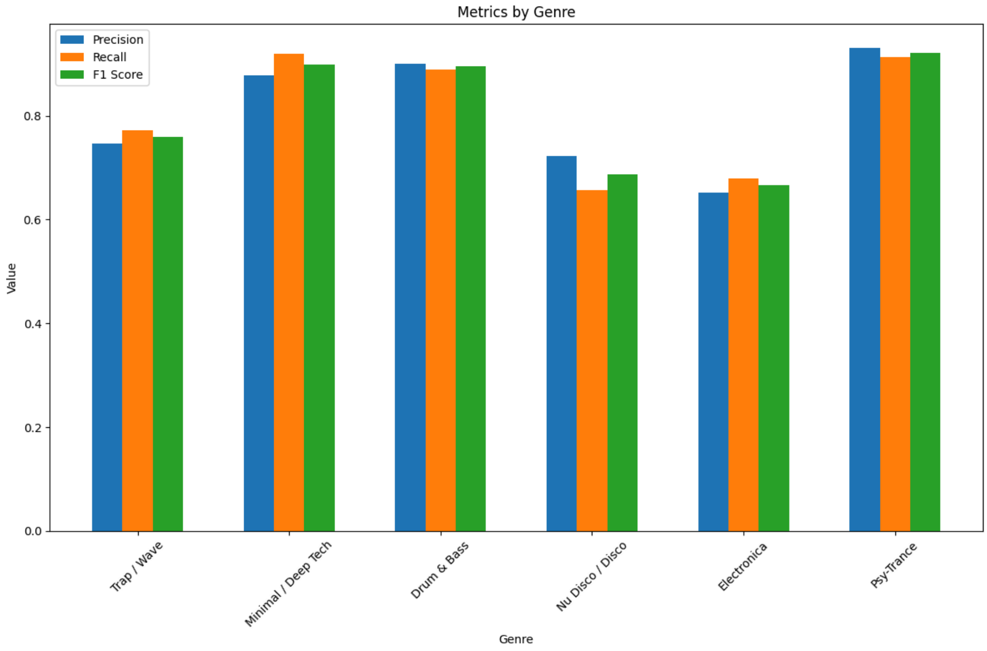


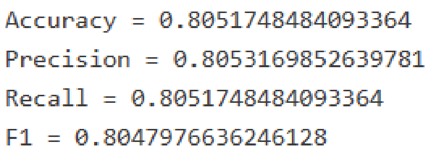


```
              precision    recall  f1-score   support

         edm       0.50      0.37      0.43       383
       latin       0.45      0.17      0.24       421
         pop       0.38      0.50      0.44       783
         r&b       0.47      0.44      0.46       637
         rap       0.57      0.64      0.60       635
        rock       0.57      0.64      0.60       649

    accuracy                           0.49      3508
   macro avg       0.49      0.46      0.46      3508
weighted avg       0.49      0.49      0.48      3508
```

AUC Score: 0.5656

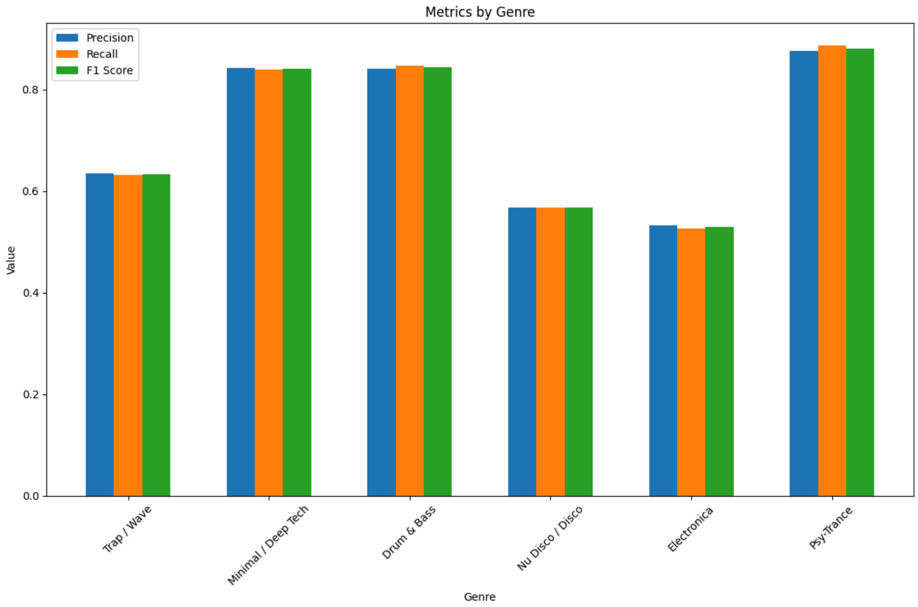
3- Random Forest

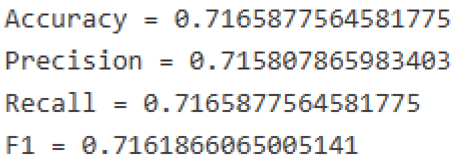




```
Accuracy = 0.8051748484093364
Precision = 0.8053169852639781
Recall = 0.8051748484093364
F1 = 0.8047976636246128
```

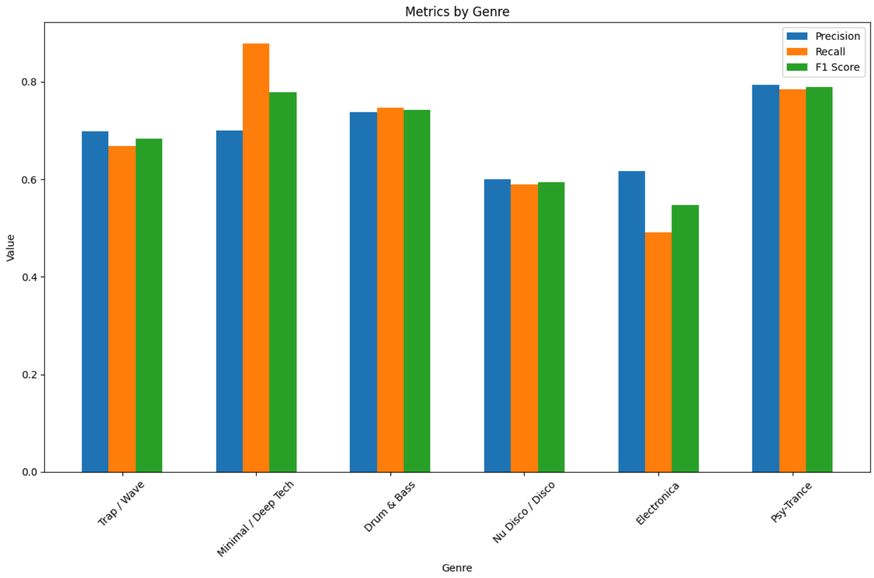
4- Decision Tree Classifier

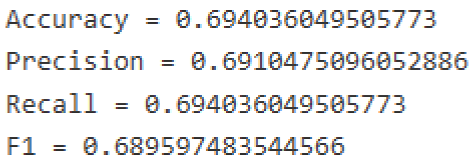




```
Accuracy = 0.7165877564581775
Precision = 0.715807865983403
Recall = 0.7165877564581775
F1 = 0.7161866065005141
```

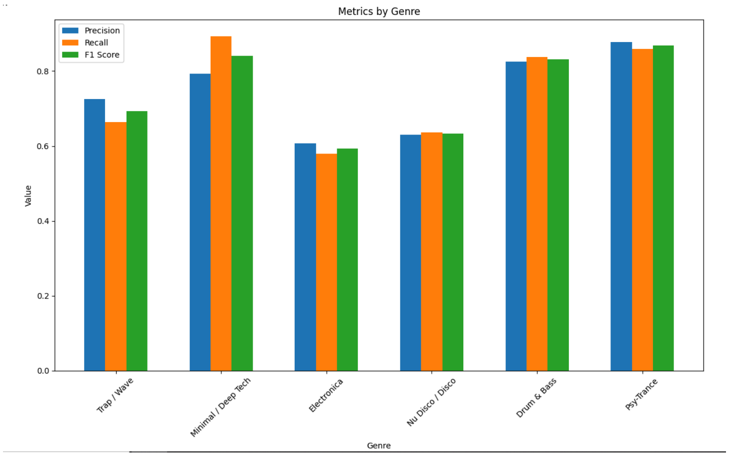
5- Linear Regression

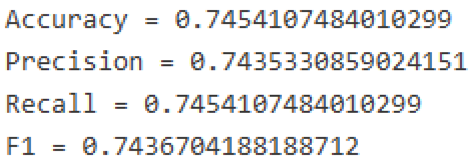




```
Accuracy = 0.694036049505773
Precision = 0.6910475096052886
Recall = 0.694036049505773
F1 = 0.689597483544566
```

6- SVM





```
Accuracy = 0.7454107484010299
Precision = 0.7435330859024151
Recall = 0.7454107484010299
F1 = 0.7436704188188712
```

In Summary Easy to recognize features where Minimal/Deep Tech and psy-trance While electronica and Disco where harder to classify which can be explained by that they don’t have mush significanc in terms of distributions in different genres