Nishant Mishra
23178030
MADE
January 18, 2024

Songs Analysis: Using Billboard and Spotify Dataset

Introduction

The purpose of this project is to analyze a given song and extract relevant information from it. In the music industry, understanding the factors that contribute to a song's popularity is crucial for artists, producers, and record labels. This project aims to analyze the influence of lyrics and musical features on the popularity of songs. By combining information from the Spotify Dataset, and the Billboard Hot 100 Dataset, and trying the proposed methods as below: Sentiment Analysis: Analyze song lyrics to determine sentiment and emotional content. Correlation Analysis: Examine correlations between audio features (from Spotify data) and chart performance data (from Billboard data).Further analysis is on report.ipynb or final-report.pdf (https://htmlpreview.github.io/?https://github.com/nish-nm/made-template/blob/main/project/final-report.html)

To replicate project please create an account on data.world and kaggle, you would need api keys for downloading datasets, otherwise you are free to use repective data files in "/data/" folder More instructions to setup data.world and kaggle api keys are at the respective urls: data.world and kaggle. Make sure to use setup Python v3.9.13 or above, use requirements.txt to install all dependencies alternatively you can also use pipenv to setup your Python enviournment.

Data Preparation

About Billboard dataset:

* Metadata URL:
Billboard hot-100 with lyrics (https://data.world/tazwar2700/billboard-hot-100-with-lyrics-and-emotion-mined-scores)
* Data URL: https://data.world/tazwar2700/billboard-hot-100-with-lyrics-and-emotion-mined-scores/workspace/file?filename=Final+processed+dataset.xlsx
* Data Type: XLSX
Excel file provided on data.world (https://data.world/), contains lyrics and songs in Billboard's top 100 from 1958 to 2020.

About Spotify dataset:

* Metadata URL: Spotify Dataset 1921- 2020 (https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks)
* Data URL: https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks/download?datasetVersionNumber=1
    * Data Type1: CSV (artists.csv)
    * Data Type2: CSV (tracks.csv)
    * Data Type3: JSON (dict_artists.json)

The Important file we will be exploring will be tracks.csv which contains information about songs and their musical features, while other files provide some good information this is currently not relevant to our analysis.
The CSV file provided on kaggle(https://www.kaggle.com/datasets), contains 600k+ songs on the popular streaming platform Spotify and contains information about artists, their tracks, and Musical Features.

The fetch_data.py script fetches data from the two websites. The '.env' file contains all the credentials for Kaggle API and DataWorld API.

About load_dataworld function:

```
"""
    Downloads the dataset from DataWorld and saves it as an XLSX file.
    If there are any null values in the DataFrame, it drops them and saves the DataFrame as a CSV file.

    Parameters:
    api_key (str): The API key for DataWorld.
    output_folder (str): The folder where the XLSX file will be saved. Default: "/Users/sash/projects/MADE/made-template/data/dataworld"
    output_filename (str): The name of the XLSX file. Default: "Spotify_billboard.xlsx"
    csv_filename (str): The name of the CSV file. Default: "Spotify_billboard.csv"
```

```
 Returns:
    pandas.DataFrame: The DataFrame containing the dataset.
"""
```

About kaggle_download function:

```
"""
    Download all files from a Kaggle dataset.

    Parameters:
    - dataset_name (str): Name of the Kaggle dataset (e.g., 'yamaerenay/spotify-dataset-
19212020-600k-tracks').
    - destination_folder (str): Destination folder for downloading files.

    Returns:
    - None
"""
```

About split_csv function:

```
"""
    Split a large CSV file into smaller files based on size.

    Parameters:
    - input_file (str): Path to the input CSV file.
    - output_folder (str): Folder where the smaller CSV files will be saved.
    - file_size_limit_mb (int): Maximum size (in MB) for each smaller CSV file. Default
is 25 MB.

    Returns:
    - None
"""
```

We need a split_csv function as file 'tracks.csv' is a large file and GIT does not not let large files more than 100 mb, so to takle it the function creates 5 csv files each of size around 25 mb's to make it easier to push the files. The fetch_data.py script also checks for null values and drops the null rows making it easier for analysis.

Data Preprocessing:

Billboard data:

```
['url', 'WeekID', 'Week Position', 'Song', 'Performer', 'SongID',
       'Instance', 'Previous Week Position', 'Peak Position', 'Weeks on Chart',
       'Lyrics', 'Artist', 'words', 'wordCount', 'languages', 'all_words',
       'allWordCount', 'year', 'decade', 'MTLD', 'TTR', 'CTTR',
       'sentimentScore', 'sentimentScore_pos', 'sentimentScore_neg',
       'emo_words', 'emoWordCount', 'joy', 'joy_normalized', 'sadness',
       'sadness_normalized', 'anger', 'anger_normalized', 'disgust',
       'disgust_normalized', 'trust', 'trust_normalized', 'anticipation',
       'anticipation_normalized', 'fear', 'fear_normalized', 'surprise',
       'surprise_normalized', 'emo_score', 'happy', 'happy_normalized',
       'sorrow', 'sorrow_normalized']
```

url: song/track url
WeekID: The date when song was published
Song: Song Name
Performer: Music Artist/Band
Lyrics: Words contained in lyrics scraped from genius.com
allWordCount: Word Counter for lyrics
words: a list of words seperated by a ‘,’.
emo_words: words contributing to emotion.
joy: joyous/happy words.
sadness: words with sad emotions.
disgust: disgustful words
trust: words invoking trust as emotion
fear: fear invoking words in lyrics
happy: happiness invoking words
sorrow: words that can crush your heart

Spotify Data

Tracks.csv
id: id of track
name: name of track
popularity: popularity of track in range 0 to 100
duration_ms: duration of songs in ms
explicit: whether it contains explicit content or not
artists: artists who created the track
id_artists: id of artists who created the track
release_date: date of release
danceability: how danceable a song is in range 0 to 1
energy: how energized a song is in range 0 to 1

Methods:

Emotion Analysis

Sentiment analysis to analyze positive and negative emotions in song lyrics. Emotion scores were aggregated by decade. To achieve this it is important to load csv file containing Billboard data. A create_decade function in report.ipynb does the job creating a new column called decade in dataframe, 60s represent all the songs released in 1950 to 1960(including 1950 but excluding 1960), and so on. Next step in analysis goes into creating a histogram with x-axis as years and y-axis number of songs.
Next step was to group words of lyrics according to decade.

	decade	all_words
0	00s	['produce', 'scott', 'storch', 'juelz', 'santa...
1	10s	['say', 'oh', 'god', 'see', 'way', 'shine', 't...
2	20s	['im', 'like', 'water', 'ship', 'roll', 'night...
3	50s	['girl', 'change', 'bobby', 'sox', 'stocking',...
4	60s	['still', 'though', 'broke', 'heart', 'still',...
5	70s	['saturday', 'night', 'saturday', 'night', 'sa...
6	80s	['joey', 'tempest', 'light', 'go', 'see', 'rea...
7	90s	['puffy', 'like', 'right', 'yeah', 'bad', 'boy...

Table 1

Next step was to analyze positive and negative emotions in song lyrics.
The positive emotions and negative mapped using following features.

```
# Select the relevant columns for emotion scores
positive_emotions = ['joy_normalized', 'trust_normalized', 'surprise_normalized']
negative_emotions = ['sadness_normalized', 'anger_normalized', 'disgust_normalized',
'fear_normalized']
```

The results obtained by plotting positive and negative emotions will be discussed in further section.
To analyse the words prominent in songs lyrics during 1970s and 2010s. This would highlight the emotions that are prominent withh audience and why everyone is listening to and relating to which sentiment and emotion in respective years. Word clouds were used for this usecase.

```
# Function to create and display a minimalistic word cloud
def generate_minimal_wordcloud(words, title):
    wordcloud = WordCloud(width=800, height=400, background_color='white',
max_words=50,
                          colormap='viridis', contour_width=1,
contour_color='black').generate(words)

    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis('off')
    plt.show()

# Generate minimalistic word clouds for the 70s and the 2010s
generate_minimal_wordcloud(words_70s, 'Word Cloud for All Words in Songs in the 1970s')
generate_minimal_wordcloud(words_2010s, 'Word Cloud for All Words in Songs in the
2010s')
```

Further to analyze the relevance of song emotions on popularity of a song, spotify data was loaded to a dataframe, since we split the tracks.csv into smaller chunks to make it feasible for uploading data to github. This section of code reads csv masked as 'part_*.csv' and loads every file into a dataframe 'spotify_data'. It is now key to merge spotify data an billboard data on song name as this is a key common in both the dataframes. Use of inner join and method called pd.merge() was used to achieve this.
Feature selection on merged data is important for further analysis.

```
# Select relevant features
selected_features = ['duration_ms', 'explicit', 'danceability', 'energy', 'key',
'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
'valence', 'tempo', 'time_signature', 'anticipation_normalized', 'fear_normalized',
'surprise_normalized', 'happy_normalized', 'sorrow_normalized']

# Keep only selected features
analysis_data = merged_data[selected_features + ['popularity']]
```

Correlation matrix and box plots were used to explore relationship between selected audio features and song popularity.

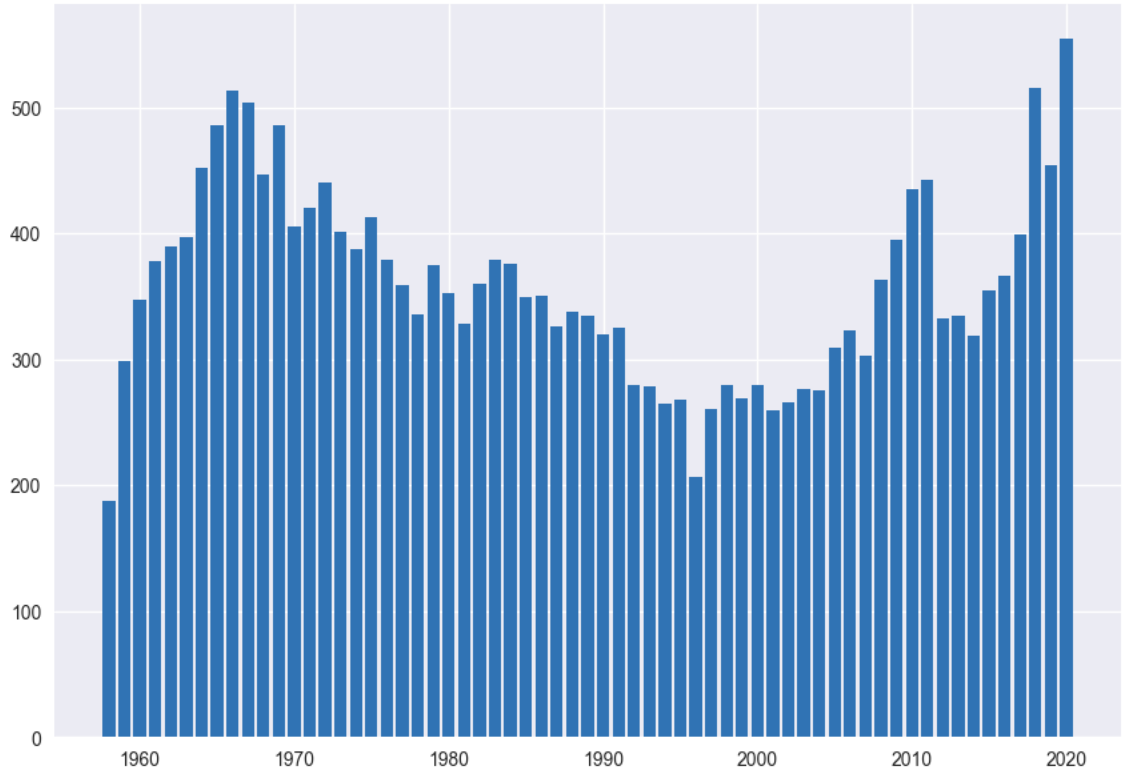
Models:

To further explore whether machine can predict popularity of songs on selected features, I employed models like Linear Regression, Ridge Regression and Autoencoders to predict popularity. The Mean Square Error was used to highlight if predictions were close to oberved andd true valuesin test-set.
The models performed badly while predicting the popularity. The lowest Mean Square Error was for Gradient Boosting Algorithm
Best Hyperparameters: {'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 150, 'subsample': 1.0}
Best Gradient Boosting Mean Squared Error: 242.30755117280606.
Since the bad results by the models, it is essential to highlight Grading Boosting Feature Importance. So that not so relevant and correlated features may be dropped in future works for better results.

Results:

Decade-wise Analysis

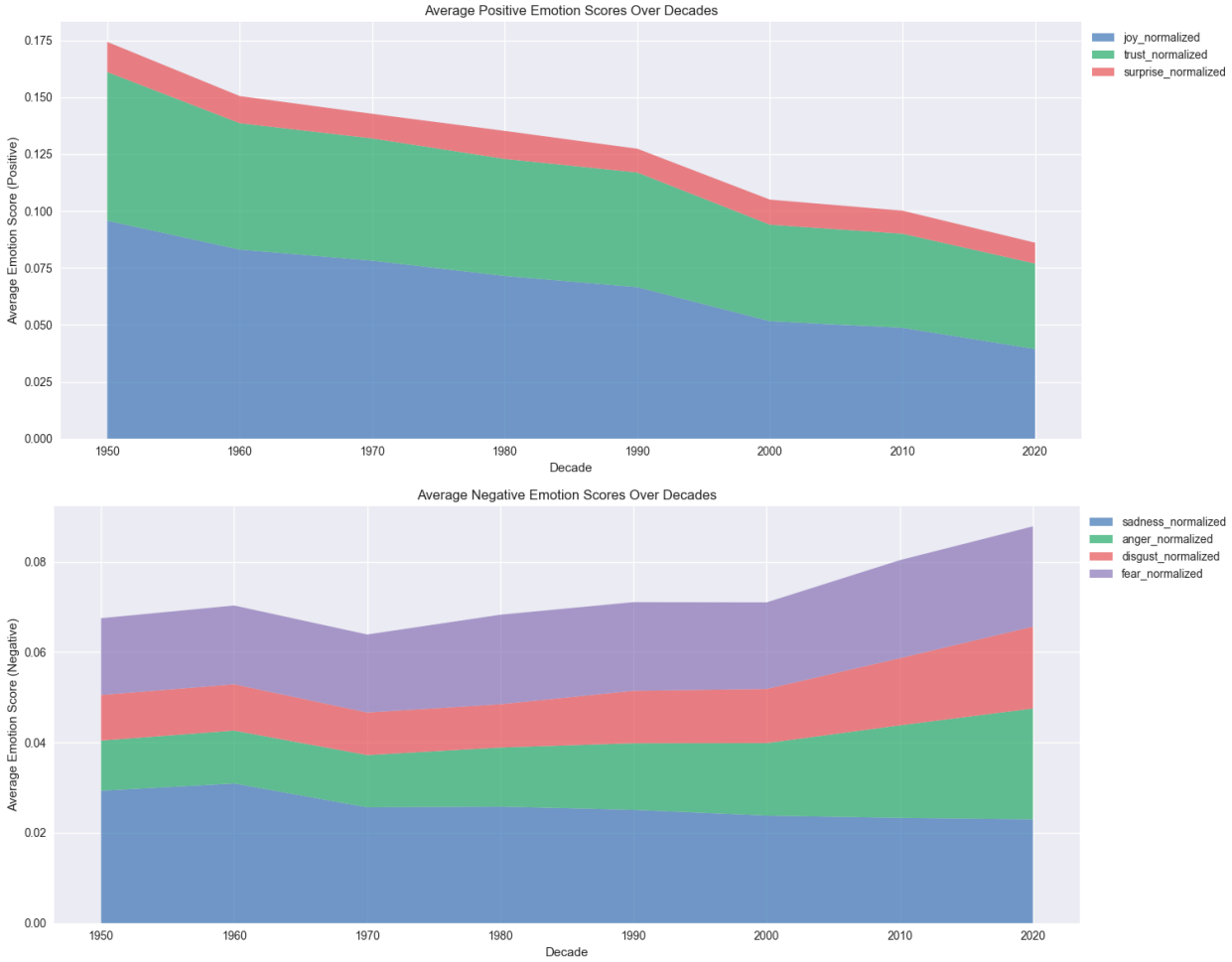
A histogram was plotted to show the distribution of songs per year. Emotion trends over decades were visualized using area charts.



As it is clearly highlighted by the feature the number of songs published in 1990-2000 was a significant drop.

Positive and Negative Emotions

Area charts displayed the average positive and negative emotion scores over decades, revealing interesting trends in lyrical content.



Very surprising is that all the positive emotions are trending downwards. That means the most popular songs have fewer and fewer words that convey these emotions as years have gone by.
It is evident that although they have been expressed in fewer quantities, the usage of negative emotion conveying words is increasing. Especially look at the emotion rivers of Fear, Anger, and Disgust. They start out so narrow and grow to be so large at the end.

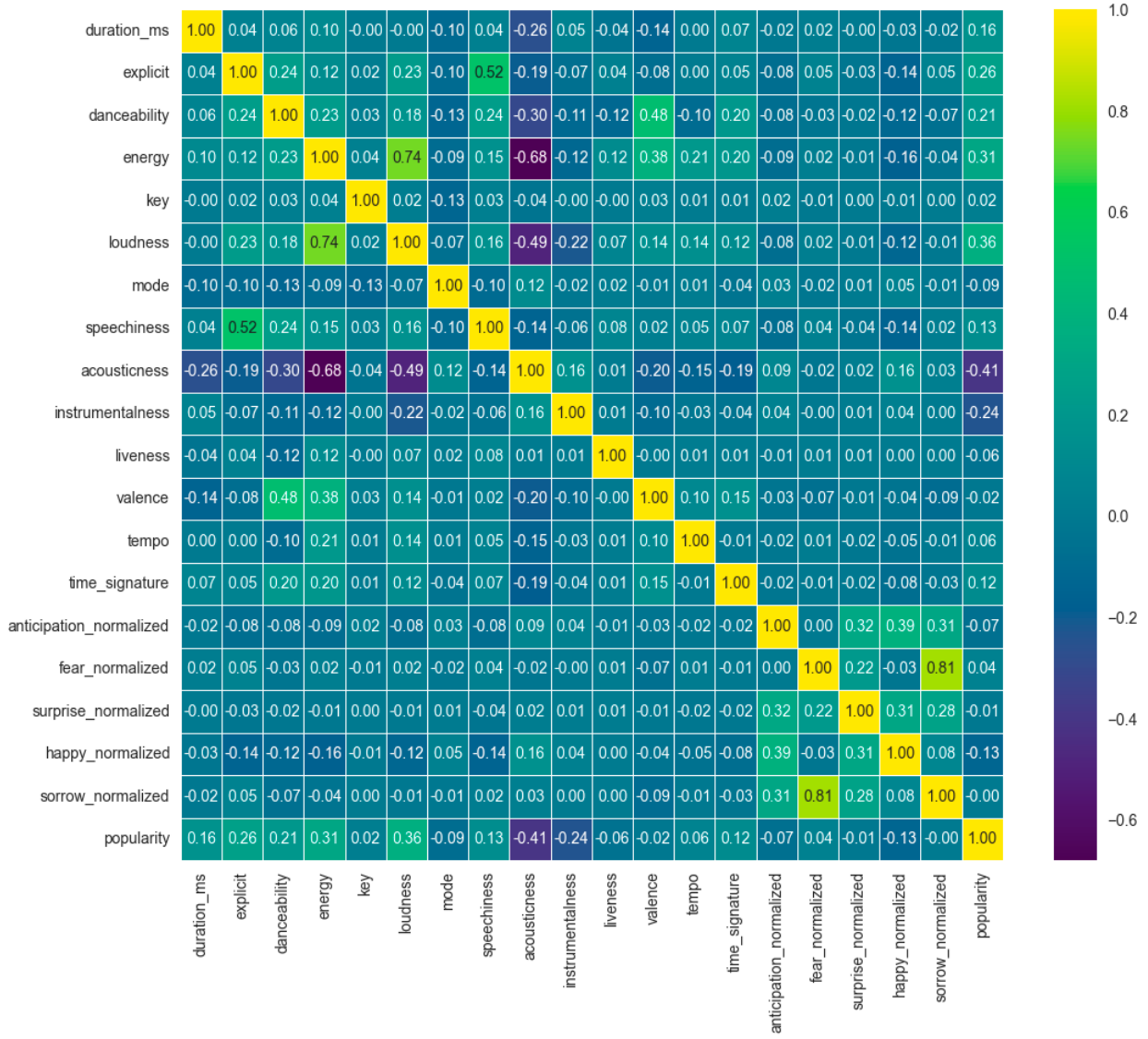
Word Clouds Comparison

Word clouds highlighted the shift in important words from the 1970s to the 2010s, indicating changes in lyrical themes.

The larger and brighter the word is, the more important it was in that decade. The importance of the word was calculated using tf-idf. As you can see, the most important words of the '70s like love, know, say which are clearly positive words get overshadowed in the 2010s.

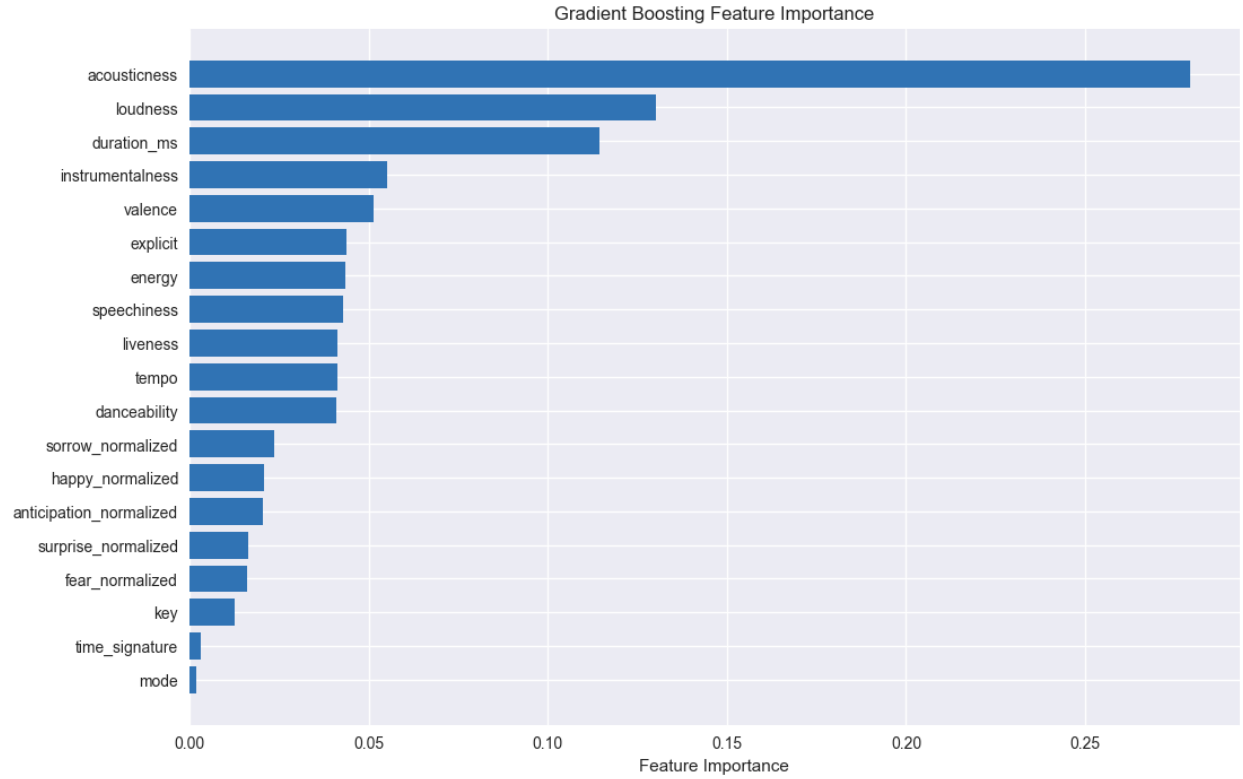
Correlation Analysis

Heatmaps illustrated the correlation between audio features and song popularity, providing insights into the impact of musical characteristics.



sorrow_normalized and fear_normalized have a correlation of 0.81, meaning that both of them affect each other highly. Analysisng this effectively can bring down our relevant features in data and help building a generalized model.

Feature Importance



Limitations:

Potential limitations in data quality, model assumptions, and generalizability were acknowledged, A key reason for high MSE for all models

Future Work:

Assuming feature importance its important to eliminate not relevant feature and try other dimensionality reduction to improve on model performace and achive a lower MSE score