Sentiment-Driven Spotify Music Recommendation: Leveraging Social Media Posts and User Playlists for Personalized Music Experiences

Md Badiuzzaman Pranto

January 9, 2024

Contents

1 Introduction

The primary objective of this project is to build a song recommendation system based on the sentiment extracted from users' social media posts. The initial phase involved training a Logistic Regression model using Twitter posts, followed by an evaluation of its performance in sentiment detection from social media content. The model demonstrated a commendable 76% test accuracy. Subsequently, leveraging the trained model, the sentiment of individual users is detected. The recommendation system then integrates both the user's sentiment and their existing music playlist to provide personalized music suggestions. This dual consideration aims to enhance the relevance and emotional resonance of the recommended songs, contributing to a more tailored and engaging music recommendation experience.

1.1 Project goals

In the ever-evolving landscape of personalized music experiences, the project seeks to redefine the art of music curation by integrating the dynamic realm of social media sentiment analysis. The primary objective is to offer users a tailored music recommendation system that not only aligns with their individual tastes but also resonates with the emotional context conveyed through their social media posts.

Key Components:

1. Sentiment Analysis Model: Train a logistic regression model to discern sentiment from social media posts. By understanding the emotional nuances expressed in user-generated content, I aim to capture the mood and preferences that influence music choices.

2. User Playlist Integration: Leverage the sentiment scores obtained from the logistic regression model alongside the user's existing playlist data. By incorporating individual playlist preferences, our system strives to provide a holistic understanding of a user's musical inclinations.

3. Recommendation Engine: Develop a sophisticated recommendation engine that synthesizes sentiment analysis results and playlist data. The engine will dynamically adapt to users' changing emotions and preferences, ensuring that music suggestions are not only personalized but also responsive to the evolving sentiments expressed in their social media interactions.

In essence, the project work envisions a music recommendation system that transcends traditional genre-based approaches. By harnessing the power of sentiment analysis and user playlists, I aspire to create a deeply personalized and emotionally intelligent music streaming experience, revolutionizing the way users discover and connect with their favorite tunes.

1.2 Why logistic regression?

Explainability in machine learning or deep learning models are very essential for building trust, ensuring ethical use, complying with regulations, improving model performance, and fostering collaboration between humans and AI systems. As machine learning applications continue to impact various aspects of society, the need for transparency and interpretability becomes increasingly critical. Over the last couple of years, the term *Explainable AI* has become one of the most concerning topic for researchers [2]. Many researches have been conducted to address why explaining the *black-box* models are important [2–4].

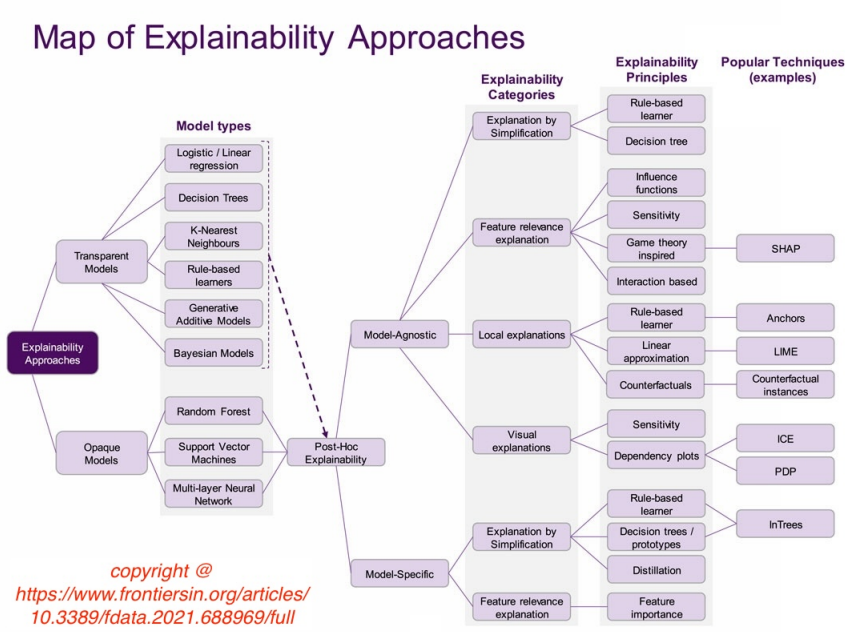

Figure 1: Map of explainability approaches.

The figure 1 above shows the explainablility approaches for different kind of models. Logistic Regression models are very simple, transparent and easy to explain. Though explaining the trained model does not fit in the scope of the project, but due to the simplicity and transparency of Logistic Regression, I started with such algorithm.

The subsequent sections of the report are structured as follows: Section 2 provides a detailed description of the datasets. Following that, Section 3 outlines the adopted methodology. The results of the project are presented in Section 4, while Section 5 encompasses the discussion. The section 6 explains limitations and future works. Section 6 delves into the challenges encountered during the project.

2 Data

In undertaking this project, I needed access to two distinct datasets. The first dataset was utilized to train a Logistic Regression model designed to discern sentiments from social media posts. The second dataset was focused on music information. Additionally, a dataset containing a user's playlist was necessary to gain insights into their musical preferences. For project evaluation, I employed a representative 10% sample (82 songs) selected randomly from the entire pool of songs to constitute the user's playlist. The section 2.1 explains the social media posts, 2.2 explains the music data, and 2.3 describes user's playlist data respectively.

2.1 Social media post (Sentiment140)

The Sentiment140 dataset is a very popular open source dataset that contains *1.6 millions* twitter posts by several users. The dataset was originally collected by Alec Go and colleagues [5]. I have collected the dataset from kaggle.

Column number	Column name	Description
0	target	Polarity of the tweet (0 = negative, 4 = positive).
1	id	The id of the tweet.
2	date	The date of the tweet.
3	flag	The query. If there is no query, then this value is NO_QUERY.
4	user	The user that tweeted.
5	text	The text of the tweet.

Table 1: summary of Sentiment140 dataset.

Column number	Column name	Description	Min	Max	Avg
0	track_name	Song name.	N/A	N/A	N/A
1	artist(s)_name	Artist(s) name.	N/A	N/A	N/A
2	artist_count	Number of artists.	1	8	1.568
3	released_year	Release year.	N/A	N/A	N/A
4	released_month	Release month.	N/A	N/A	N/A
5	released_day	Release day.	N/A	N/A	N/A
6	in_spotify_playlists	Song in number of Spotify playlist.	31	52898	4849.898
7	in_spotify_charts	Rank on Spotify charts.	0	147	11.722
8	streams	Number of streams on Spotify.	N/A	N/A	N/A
9	in_apple_playlists	Song in number of Apple playlist.	0	532	60.162
10	in_apple_charts	Song rank on Apple Music charts.	0	275	49.474
11	in_deezer_playlists	Song in number of Deezer playlist.	N/A	N/A	N/A
12	in_deezer_charts	Song rank on Deezer charts.	0	45	2.452
13	in_shazam_charts	Song rank on Shazam charts.	N/A	N/A	N/A
14	bpm	Beats per minute.	65	206	122.565
15	key	Key of the song.	N/A	N/A	N/A
16	mode	Mode of the song (major or minor).	N/A	N/A	N/A
17	danceability_%	How suitable the song is for dancing.	23	96	67.392
18	valence_%	Positivity of the song.	4	97	51.202
19	energy_%	Energy level of the song.	14	97	64.362
20	acousticness_%	Amount of acoustic sound.	0	97	26.310
21	instrumentalness_%	Amount of instrumental content.	0	91	1.677
22	liveness_%	Presence of live performance elements.	3	97	18.170
23	speechiness_%	Amount of spoken words in the song.	2	64	10.526

Table 2: summary of Spotify songs dataset.

The table 1 shows the summary of this dataset and a short description of data source is provided below:

- Metadata URL: https://www.kaggle.com/datasets/kazanova/sentiment140
- Data URL: https://www.kaggle.com/datasets/kazanova/sentiment140
- Data Type: ZIP

2.2 Spotify songs

This is an open source dataset that contains a comprehensive list of the most famous songs as listed on Spotify. The dataset offers a wealth of features beyond what is typically available in similar datasets. The dataset has total *943* rows. It provides insights into each song's attributes, popularity, and presence on various music platforms. *In this project, I will refer bpm, danceability_%, energy_%, acousticness_%, instrumentalness_%, liveness_%, speechiness_% as audio features.*

The table 2 shows the summary of this dataset and a short description of data source is provided below:

- Metadata URL: https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023
- Data URL: https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023
- Data Type: ZIP

2.3 User's playlist

A user's playlist is always a subset of all available songs. If all available songs are the dataset in section 2.2, then a user's playlist is a subset of this dataset. The simplest way to evaluate the project's performance was to use 10% *(82 songs)* of random data from section 2.2. as user's playlist in order to understand user's preference on music. The playlist is uploaded to /data/ directory of this repository as a csv file

3 Methodology

The methodology, as outlined in figure 2, has been followed while conducting this project work. The steps are as follows:

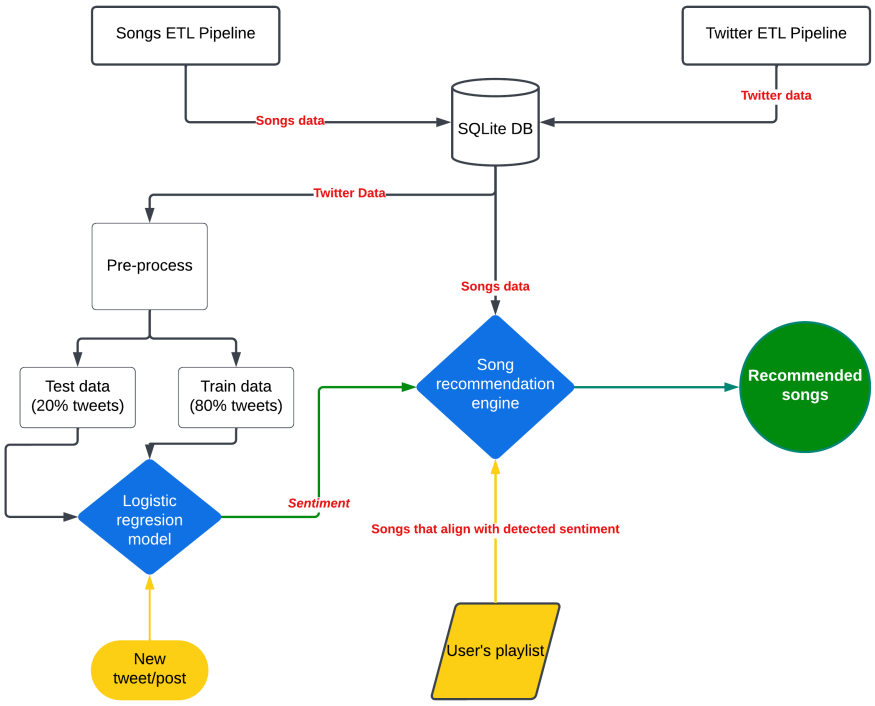

Figure 2: Methodology followed in this project.

Algorithm 1: Steps followed in this project's methodology.

1 Run the Extract Transform Load (ETL) pipelines to get required data.
2 Perform pre-processing on twitter data.
3 Split the twitter data into train and test set.
4 Train the Logistic regression model.
5 Evaluate and re-train the model until best parameters found.
6 Take new tweet/post as input.
7 Detect the sentiment of the user from the tweet.
8 Give the detected sentiment, user's playlist and all available songs to Song recommendation engine as input
9 Get the recommended songs as output.

Understanding user's preference and recommending songs takes place in the song recommendation engine, which makes up the user's songs preference algorithm. I have broken down this algorithm in detail in Section 3.4. This section explains the step-by-step processess I used to figure out and cater to each person's specific music preferences.

3.1 Extract Transform Load (ETL) Pipeline

Given the project's need for two distinct datasets, two ETL pipelines were meticulously developed for data collection. Each pipeline initiates with data extraction from its designated source, tailors transformations to meet individual

requirements (outlined in Section 3.1.1), and culminates by storing the processed data in a SQLite database for subsequent utilization.

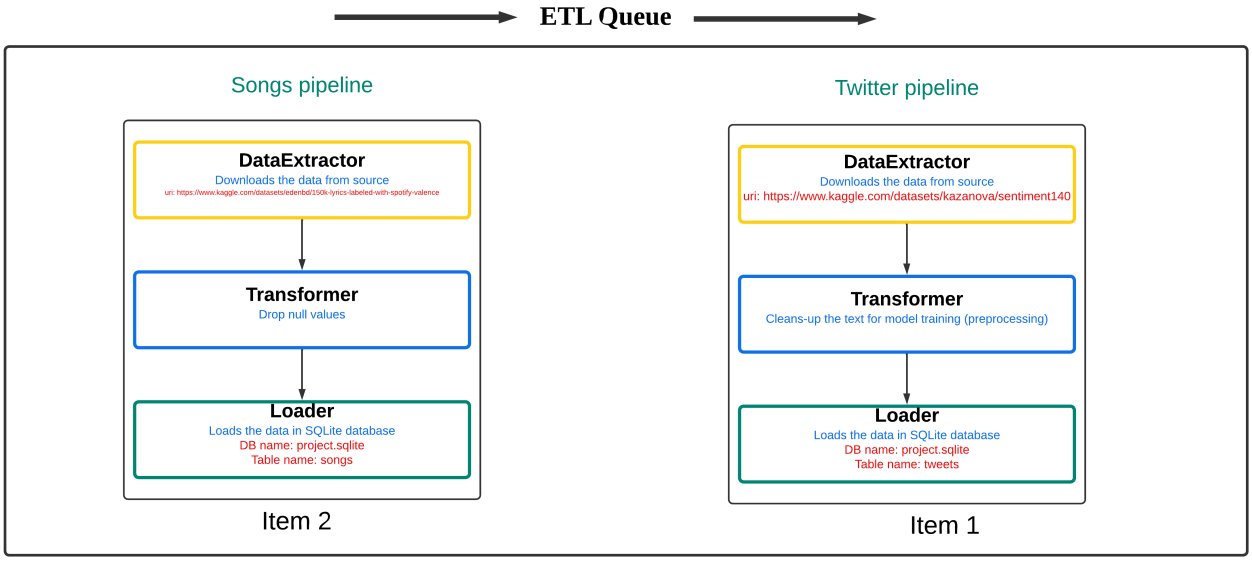

Figure 3: Extract Transform Load (ETL) Pipeline used in this project work.

The figure 3 visualizes the structure of ETL pipelines used in this project work. An ETL Queue has been used to run the pipelines sequentially. The item number in the figure shows the execution sequence of the pipelines in the queue.

After analysing the basic requirements of a data science/machine learning projects, I and my college Arni Islam [6] have developed a python package to Run ETL pipelines in python environment. The package is described in appendix A. I would like to request you to read and give us feedbacks accordingly.

3.1.1 Data transformation in the ETL pipeline

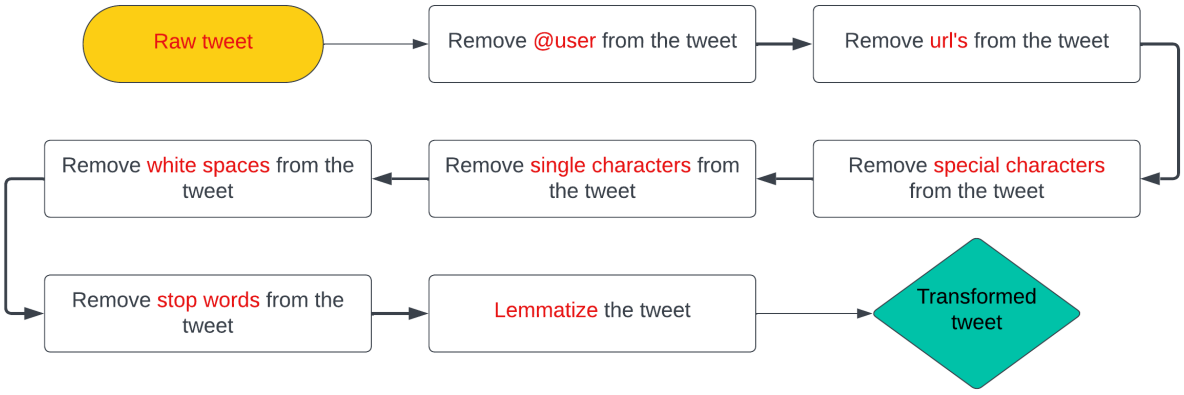

Figure 4: Transformation in ETL pipeline for Tweeter dataset.

The Twitter dataset served as the foundation for model training; however, it posed challenges due to the presence of special characters, @user tags, URLs, single characters, and extraneous white spaces within the tweets. To prepare the dataset for effective model training, a comprehensive cleaning process was initiated, involving the removal of these artifacts. Additionally, the dataset underwent a refinement process that included the removal of stop

words and lemmatization [7]. The cleaning process is visually represented in Figure 4, illustrating the step-by-step execution for each tweet. This cleaning procedure was integral to the transformation step of the Twitter ETL pipeline. The twitter pipeline took approximately 30 minutes to Extract, Transform and Load 1.6 million tweets the data. Conversely, for the other two pipelines, the transformation was limited to the removal of null values, and dropping unnecessary attributes with no further data alterations applied. There was no null values in twitter data.

3.2 Data pre-processing

Achieving success in sentiment analysis hinges significantly on adept text preprocessing, which involves converting raw textual data into a format compatible with machine learning algorithms. Much of the required preprocessing has already been completed in the transformation stage of the automated ETL pipeline. In the specific data preprocessing step:

1. The cleaned tweets underwent initial tokenization, breaking them down into individual words or phrases using the tokenization capabilities provided by nltk [8].
2. Subsequently, the tokenized data was transformed into a numerical format using TF-IDF algorithm of sklearn [9], making it suitable for input into machine learning models.

3.3 Logistic regression model

The logistic regression model transforms the linear combination of input features into a probability using the logistic function (sigmoid function). For a binary classification task, the logistic regression equation can be expressed as follows:

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_n X_n)}}$$

Here,

- $P(Y=1)$ is the probability of the event Y occurring (e.g., positive sentiment),
- β_0 is the intercept term,
- $\beta_1, \beta_2, ..., \beta_n$ are the coefficients corresponding to the input features $X_1, X_2, ..., X_n$,
- e is the base of the natural logarithm.

A GridSearchCV search was performed to find best hyper-parameter out of:
$penalty : [l1, l2, elasticnet]$,
$solver : [lbfgs, newton-cg, sag, saga]$,
$l1\_ratio : [1]$ and,
$max\_iter : [1000, 2000, 3000]$
The best parameter found was: $LogisticRegression(l1\_ratio = 1, max\_iter = 1000, solver =' saga')$

3.4 Song recommendation engine

To elevate the personalization of music recommendations, this project centers on conducting a thorough analysis of the user's musical preferences by examining key audio features (bpm, danceability, energy, acousticness, instrumentalness, liveness, and speechiness) present in their existing playlist. The algorithm 2 outlines the functionality of the song recommendation engine.

Algorithm 2: Algorithmic workflow of song recommendation engine.

1. Get user's sentiment from the trained model.
2. Get songs from the user's playlist that align with the predicted sentiment.
3. Calculate the average values of audio features based on the songs selected from the user's playlist.
4. Exclude user's playlist songs from all songs
5. Filter songs from step-4 whose valence aligns with the predicted sentiment and the average values of the audio features are in between $calculated\_values + / - 10$ in step 3.
6. Order the selected songs by a random audio feature.
7. Return a random song.

4 Results

In this section, I have presented the evaluation results for both the classifier and the recommendation engine. The labels 0 and 1 represent negative and positive sentiments, respectively.

The confusion matrices and ROC curve for the classifier are illustrated in Figure 5 and Figure 6, respectively. A confusion matrix provides valuable insights, allowing the calculation of accuracy, recall, precision, and F1 score. I computed these performance metrics using Equations 1, 2, 3, and 4, and the results are documented in Table 3.

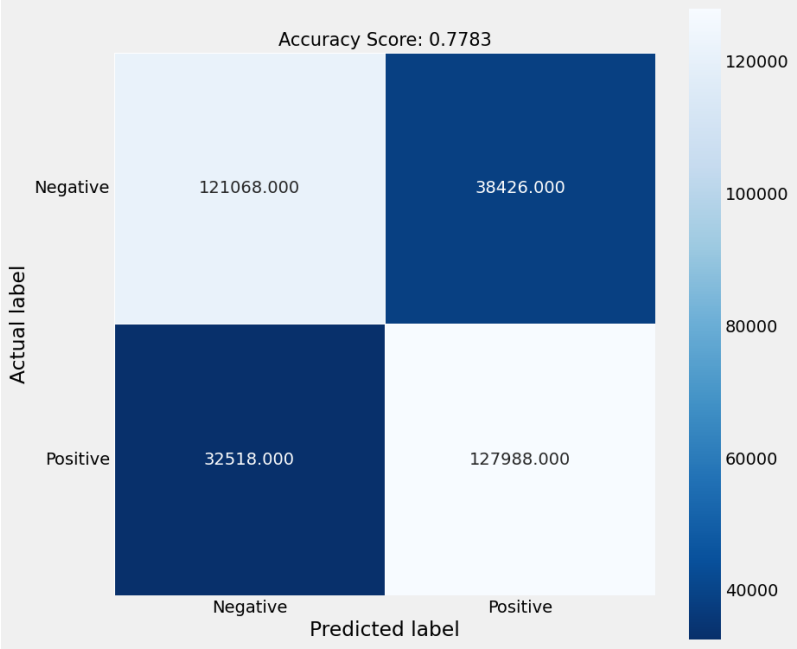

Figure 5: Confusion of the logistic regression classifier.

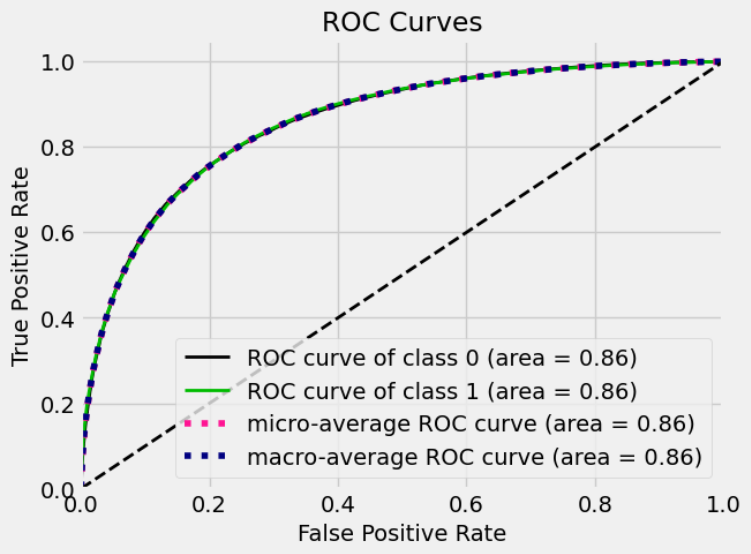

Figure 6: ROC curve of the classifier.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

Sentiment	Accuracy	Precision	Recall	F1-score
Positive	0.757	0.77	0.80	0.78
Negative	0.757	0.79	0.76	0.77

Table 3: Evaluation result of logistic regression classifier for sentiment detection.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{F1 score} = \frac{2 * recall * precision}{recall + precision} \tag{4}$$

Here,
$TP = TruePositive$,
$TN = TrueNegative$,
$FP = FalsePositive$,
$FN = FalseNegative$

In order to evaluate the recommendation engine, the user's playlist dataset in section 2.3 has been utilized. The figure 7 and 9 illustrates the audio feature summaries for negative and positive songs respectively from user's playlist.

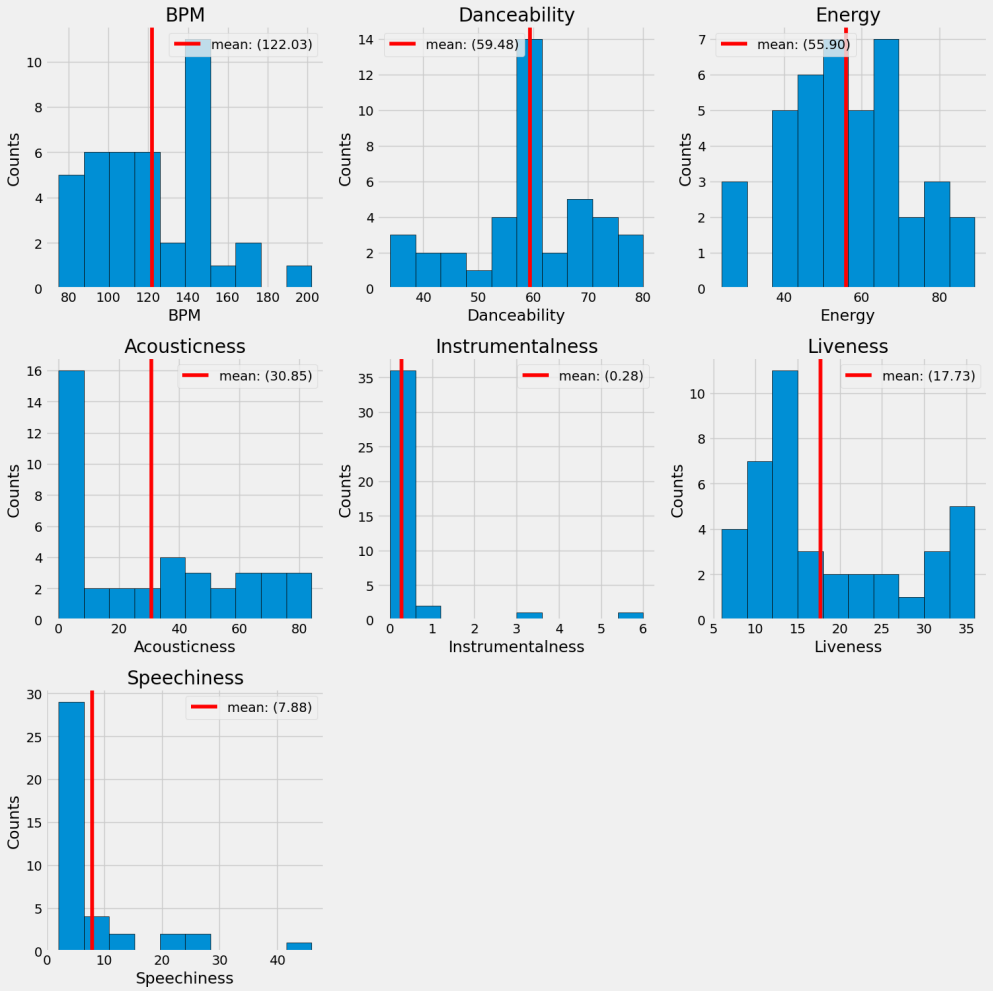

Figure 7: User playlist's audio feature summary for negative songs.

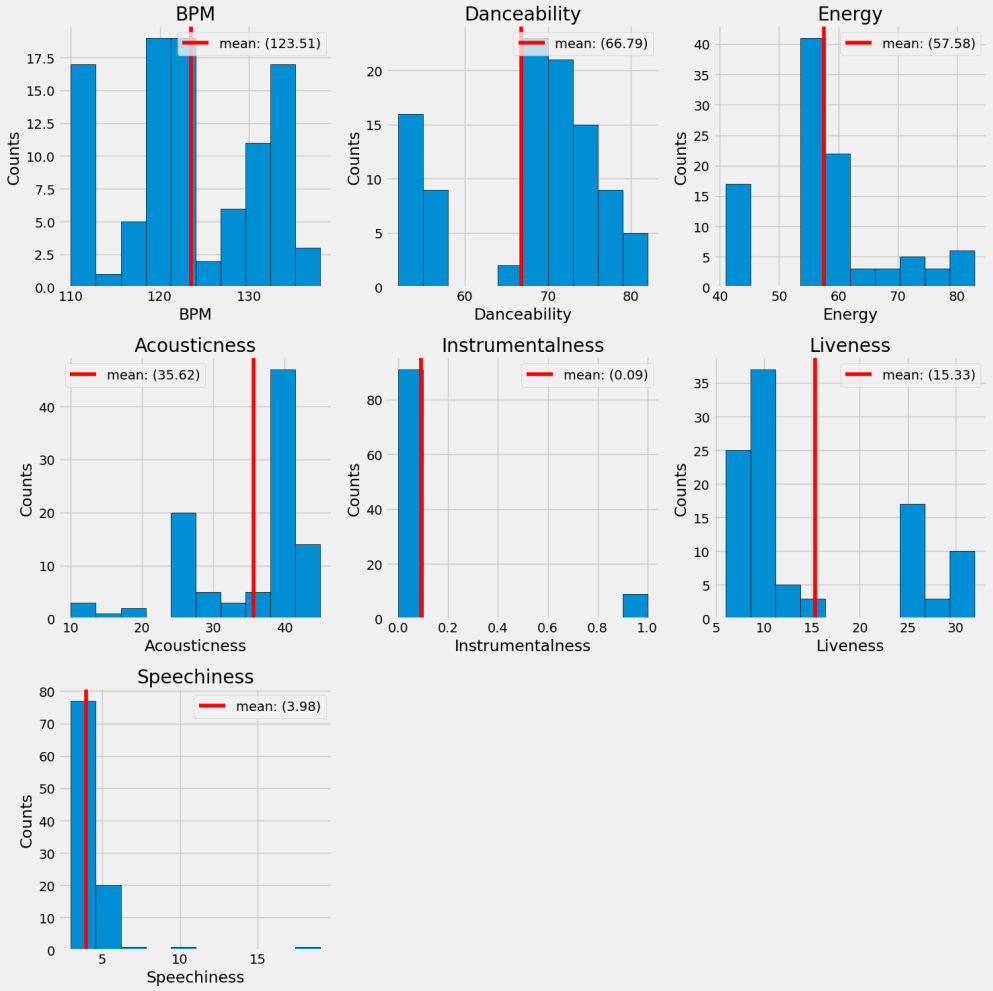

Figure 8: Audio feature summary for hundred recommended songs from negative tweets.

To get a performance measurement, hundred positive tweets and hundred negative tweets from test dataset have been used as inputs of the recommendation engine and the resulted recommended songs have been recorded. The figure 8 and 10 illustrates the audio feature summaries for negative and positive songs respectively recorded from the recommendation engine.

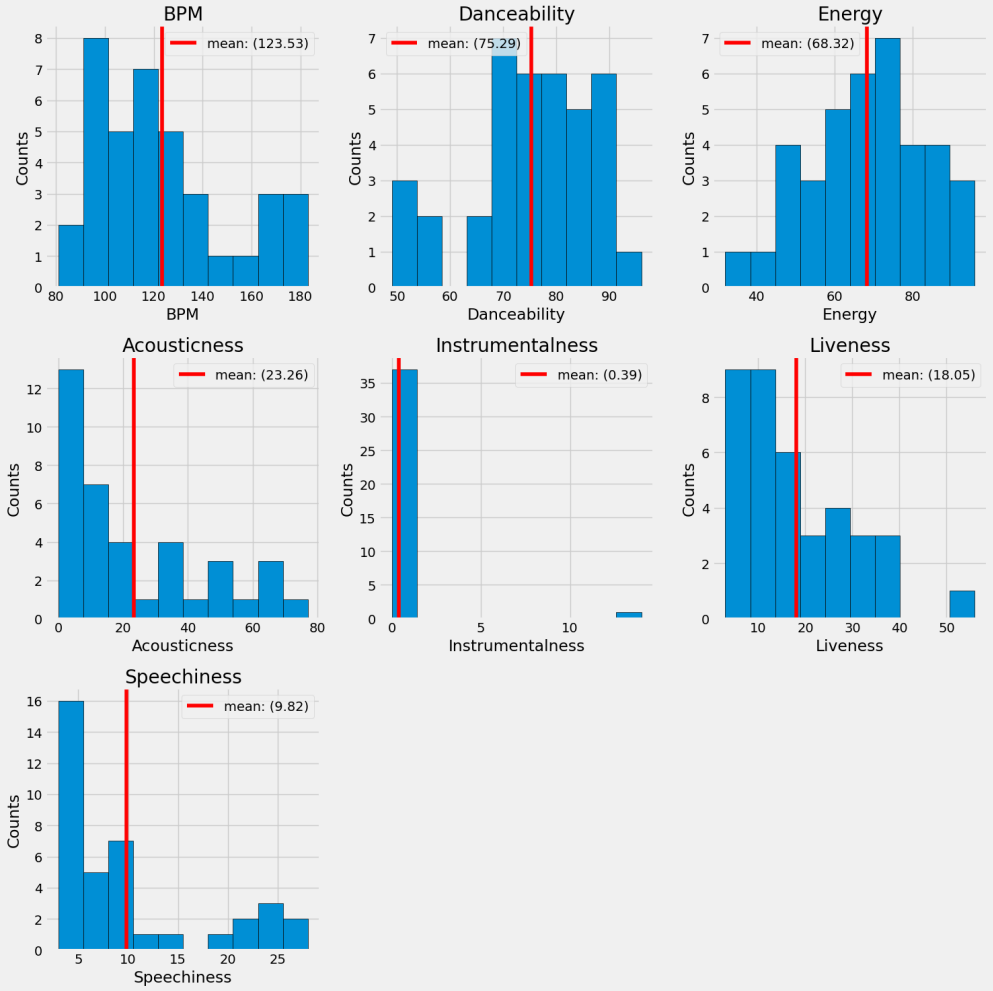

Figure 9: User playlist's audio feature summary for positive songs.

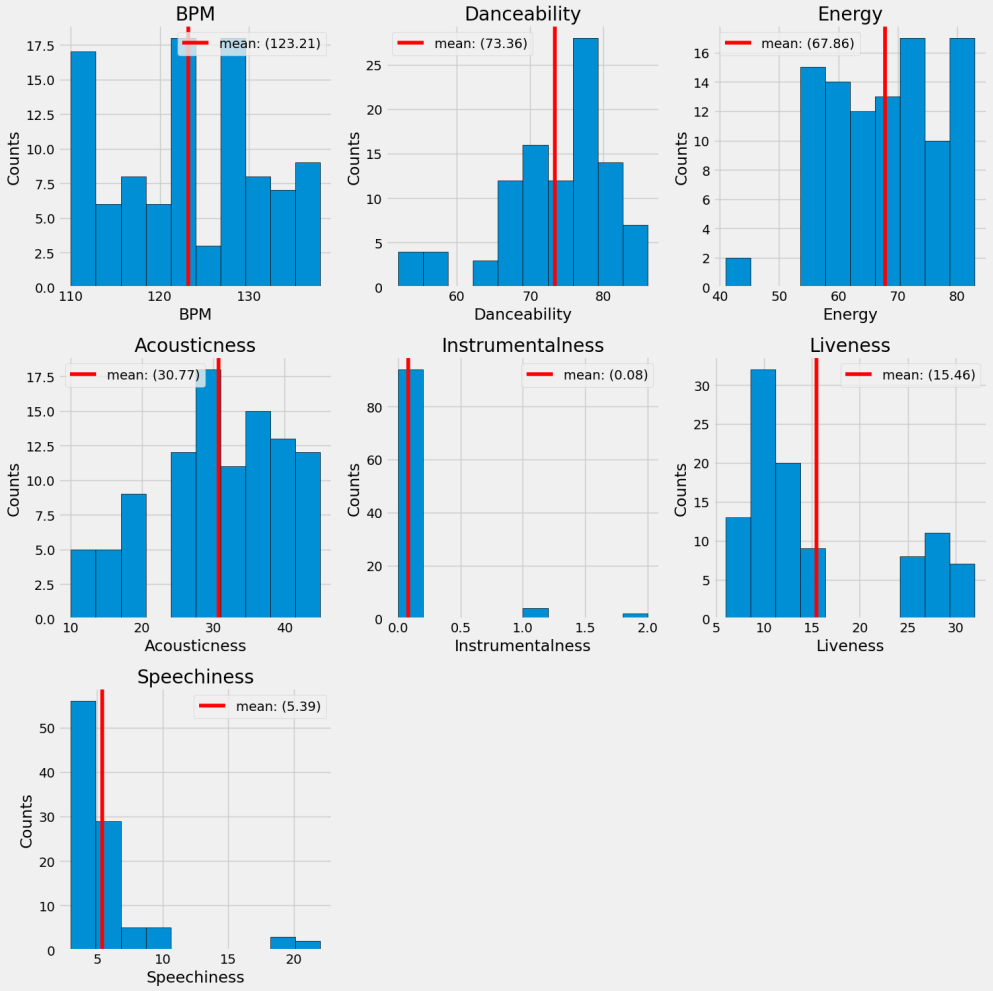

Figure 10: Audio feature summary for hundred recommended songs from positive tweets.

Source	Sentiment	BPM	Dancea.	Ener.	Acous.	Instru.	Live.	Speech.
Playlist	Negative	122.03	59.48	55.90	30.85	0.28	17.73	7.88
Recommendation Engine	Negative	123.51	66.79	57.58	35.62	0.09	15.33	3.98
Playlist	Positive	123.53	75.29	68.32	23.26	0.39	18.05	9.82
Recommendation Engine	Positive	123.21	73.36	67.86	30.77	0.08	15.46	5.39

Table 4: Recommendation engine performance report. All the audio feature values are average values.

The section 5 utilizes the results from this section and explain the output of the project.

5 Conclusion and Discussion

From the confusion matrix of figure 5, we can see that out of *159,494* negative samples, the model successfully detected *121,068* samples correctly. On the other hand, the model also detected *127,988* positive samples correctly out of *32,518* positive samples.

The table 3 shows us the accuracy, precision, recall and F1-score separately for positive and negative sentiment. With the simple logistic regression model, the scores looks good and acceptable for sentiment detection.

Another notable evaluation for the classification model is ROC curve, which is visualized in figure 6. The more area under the ROC curve, the better the model performs. For the trained logistic regression model, the area under the curve is 0.86, which is good performance.

The figure 7, 9, 8 and 10 were generated to evaluate the recommendation engine. The results are summeriezed in table 4. It is evident from the table and the figures that the recommendation engine performed well in recommending music based on the alogirithm we set.

Now let us try to find if we have meet the project's goal.

1. Sentiment Analysis Model: A logistic regression model has been trained and evaluated for sentiment analysis purposes.
2. User Playlist Integration: A user's playlist has been integrated to leverage the song recommendation.
3. Recommendation Engine: A recommendation engine has been developed that synthesizes sentiment analysis results and playlist data. The engine has dynamically adapted to users' emotions and preferences, ensuring that music suggestions are not only personalized but also responsive to the evolving sentiments expressed in their social media interactions.

Therefore, the project goal has been met.

6 Challenges

The pursuit of suitable datasets aligned with project requirements posed a significant challenge, necessitating several changes during the initial exploration. After multiple revisions, datasets that met the project's criteria were ultimately identified.

Further complications emerged during data extraction from Kaggle that requiring authentication. Overcoming these challenges was crucial for accessing the required data.

An additional noteworthy hurdle was encountered in the inconsistency of metadata. For instance, the Twitter dataset's metadata indicated three sentiment labels, yet only two were found in the actual data. This incongruity introduced complexities in accurately interpreting and utilizing the dataset.

Transformation time in ETL pipeline presented a substantial challenge, particularly with the Twitter dataset containing 1.6 million entries. The transformation takes approximately 30 minutes. To address this challenge, dummy tweets has been used in system test case instead of the real data.

7 Limitations and future works

The project has currently focused solely on assessing the logistic regression model for its suitability. There is potential for further exploration of simpler models in the context of sentiment analysis.

A noteworthy limitation is that the evaluation of the recommendation system relies solely on my implemented methods, without a comparative analysis against existing recommendation engine performances. Future work could involve conducting such comparisons to provide a more comprehensive understanding of the recommendation system's effectiveness.

References

[1] "etl-pipeline-runner." [Online]. Available: https://github.com/prantoamt/etl-pipeline-runner

[2] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/1/18

[3] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable ai: A brief survey on history, research areas, approaches and challenges," in *Natural Language Processing and Chinese Computing*, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds. Cham: Springer International Publishing, 2019, pp. 563–574.

[4] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, *Explainable AI Methods - A Brief Overview*. Cham: Springer International Publishing, 2022, pp. 13–38. [Online]. Available: https://doi.org/10.1007/978-3-031-04083-2_2

[5] B. R. Go A. and H. L., "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford*, p. 12, 2009.

[6] "Arni islam." [Online]. Available: https://github.com/islam15-8789

[7] C. U. Press, "Stemming and lemmatization." [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[8] N. team, "Nltk project." [Online]. Available: https://www.nltk.org/

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] P. of Open Source Software at the University of Erlangen, "The jvalue projec." [Online]. Available: https://jvalue.github.io/jayvee/

[11] F.-A. U. E.-N. Professorship for Open-Source Software, "Advanced data engineering." [Online]. Available: https://oss.cs.fau.de/teaching/specific/made/

A Appendix

A.1 The Python package for ETL pipeline [1]

Inspired by Jayvee [10], my colleague Arni Islam [6], and I collaborated on the development of an open-source Python package [1] throughout our coursework [11]. Our individual projects in this course had distinct objectives, resulting in the creation of ETL pipelines with varying structures. In my project, the challenge revolved around extracting data from Kaggle archives, applying transformations detailed in Figures 3 and 4, and subsequently loading it into the database. Conversely, Arni's project centered on extracting data from a source providing direct CSV files, accompanied by her unique set of transformations. Recognizing these are foundational requirements for any data science project, we deliberately structured our code in a generic manner, giving rise to the inception of this reusable Python package.

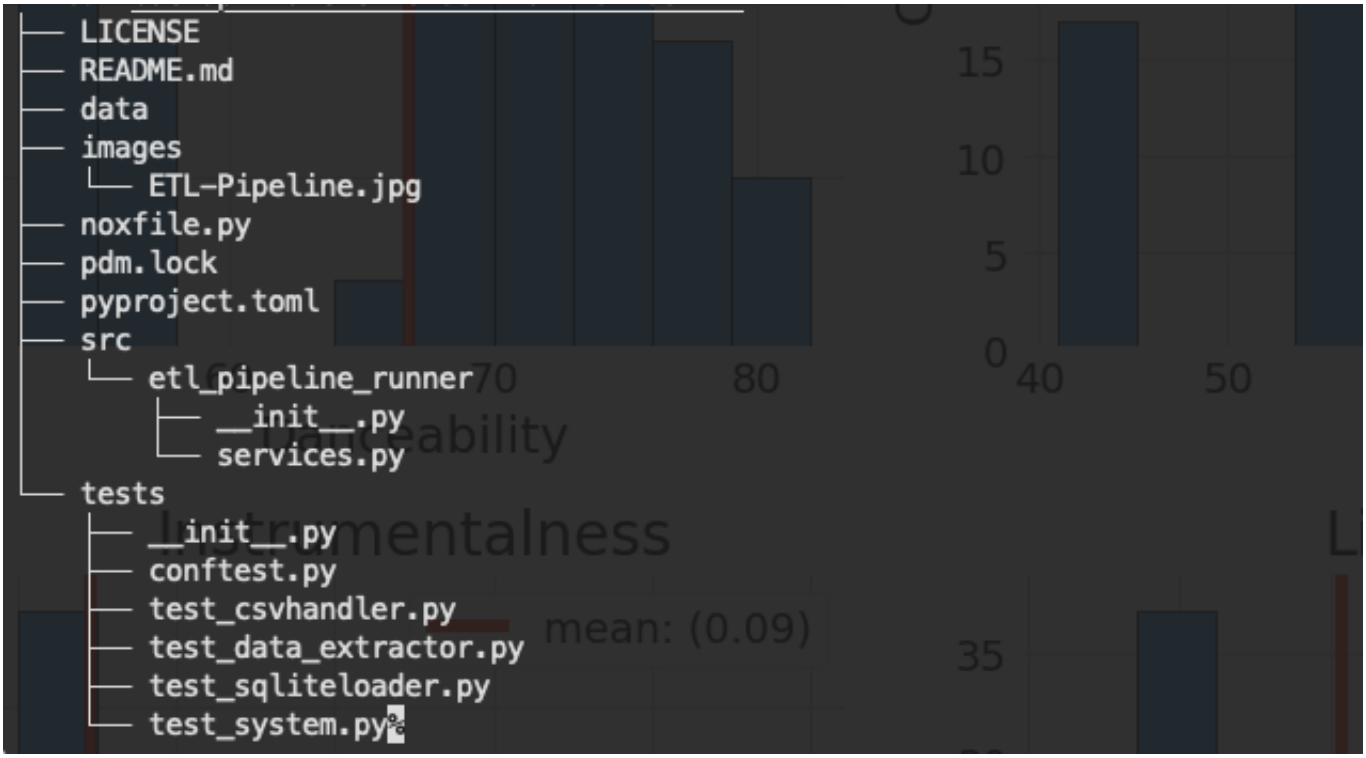

Figure 11: Directory tree of ETL Pipeline package.

Below is an example code to run an ETL Pipeline that extracts data from kaggle and stores it in a SQLite Database.

Data source: https://www.kaggle.com/datasets/edenbd/150k-lyrics-labeled-with-spotify-valence

Destination: Under song_lyrics table of project.sqlite Database. Suppose the database is located or will be created in /data directory.

```
## version 1.1.0

from etl_pipeline_runner.services import (
    ETLPipeline,
    DataExtractor,
    CSVHandler,
    SQLiteLoader,
    ETLQueue,
)
DATA_DIRECTORY = os.path.join(os.getcwd(), "data")


def transform_songs(data_frame: pd.DataFrame):
    data_frame = data_frame.drop(columns=data_frame.columns[0], axis=1)
    data_frame = data_frame.rename(columns={"seq": "lyrics"})
    return data_frame


```

```
songs_loader = SQLiteLoader(
    db_name="project.sqlite",
    table_name="song_lyrics",
    if_exists=SQLiteLoader.REPLACE,
    index=False,
    method=None,
    output_directory=DATA_DIRECTORY,
)

songs_dtype = {
    "#": "Int64",
    "artist": str,
    "seq": str,
    "song": str,
    "label": np.float64,
}

songs_csv_handler = CSVHandler(
    file_name="labeled_lyrics_cleaned.csv",
    sep=",",
    names=None,
    dtype=songs_dtype,
    transformer=transform_songs,
    loader=songs_loader,
)

songs_extractor = DataExtractor(
    data_name="Song lyrics",
    url="https://www.kaggle.com/datasets/edenbd/150k-lyrics-labeled-with-spotify-valence",
    type=DataExtractor.KAGGLE_ARCHIVE,
    file_handlers=(songs_csv_handler,),
)

songs_pipeline = ETLPipeline(
    extractor=songs_extractor,
)

if __name__ == "__main__":
    ETLQueue(etl_pipelines=(songs_pipeline,)).run()
```

A.1.1 Features as of version 1.1.0:

1. Data Extraction Capabilities:
   - Extract data from Kaggle (Contributed by me).
   - Extract data directly from sources providing CSV files (Contributed by Arni).
   - Handle CSV files seamlessly (Joint contribution).
2. Data Transformation and Loading:
   - Perform project-specific transformations with flexibility (Joint contribution).
   - Load data efficiently into SQLite databases (Joint contribution).
3. Pipeline Management:
   - Run multiple pipelines in a queue for streamlined execution (Joint contribution).

A.1.2 Possible features for future:

1. Expand Data Source Compatibility:
   - Enhance data extraction capabilities by allowing archives to be extracted from sources other than Kaggle.
2. Database Interaction:
   - Enable data extraction directly from databases, broadening the scope of data sources.
3. File Format Compatibility:
   - Handle XL/XLS files to accommodate a wider range of data formats.

4. Database Flexibility:
    - Extend data loading capabilities to support other types of databases.
5. Parallel execution:
    - Execute multiple pipelines concurrently.

You are warmly invited to submit feature requests, actively contribute to the package's development, and collectively address the evolving needs of the data science community.