Music Popularity Recommendation System

Milestone: Project Report

Group 12

Harshita Mishra

(857)-313-2520

mishra.har@northeastern.edu

Percentage of Effort Contributed by Student: 100%

Signature of Student: HARSHITA MISHRA

Submission Date: December 13th, 2023

Problem Setting:

Spotify stands out as a well-known online music streaming service that serves a diverse global audience, offering an extensive library of songs and podcasts. Users can enjoy their favorite music and podcasts while also having the option to provide feedback through ratings. Spotify utilizes these user ratings, among other metrics, to provide tailored recommendations, assisting users in exploring new music that suits their tastes. The goal of this project is to create a music popularity recommendation system by utilizing Spotify's user data and metrics, aiming to enhance the precision and effectiveness of music suggestions.

Problem Definition:

This project aims to conduct a comprehensive exploratory data analysis of the Spotify open dataset utilizing Python libraries. Our analysis will encompass a range of visualization techniques to unveil insights into the determinants of a song's popularity, including the correlations among different variables. We will implement supervised learning methods to forecast a song's popularity, assess the precision of our predictions, and identify potential errors.

The project seeks to address pertinent business inquiries, such as the factors influencing a song's popularity, the impact of variables like valence, and other elements contributing to a song's success. Furthermore, we will scrutinize the influence of an artist's genre on a song's popularity and its overall success. Our analysis aims to shed light on these crucial aspects and provide valuable recommendations for enhancing music recommendations and refining marketing strategies.

Data Sources:

Link of data sources as below:

- https://www.kaggle.com/vatsalmavani/spotify-dataset

Data Description:

Our dataset comprises three distinct files: data, genre_data, and year_data, all of which we will scrutinize to discern the key elements influencing a song's popularity. The primary dataset, data, is expansive, containing over 170,000 rows and 19 columns. Notable variables within this dataset include valence, acousticness, danceability, and duration. Additionally, it features crucial information like song ID, year, artist, energy, and liveness.

This extensive dataset will serve as the foundation for our in-depth analysis of the factors contributing to a song's popularity. Furthermore, we will draw insights from the genre_data and year_data datasets to understand the impact of an artist's genre and the song's release year on its overall popularity. Our analysis aims to provide a holistic comprehension of the diverse variables influencing a song's success, offering recommendations for enhancing both music suggestions and marketing strategies.

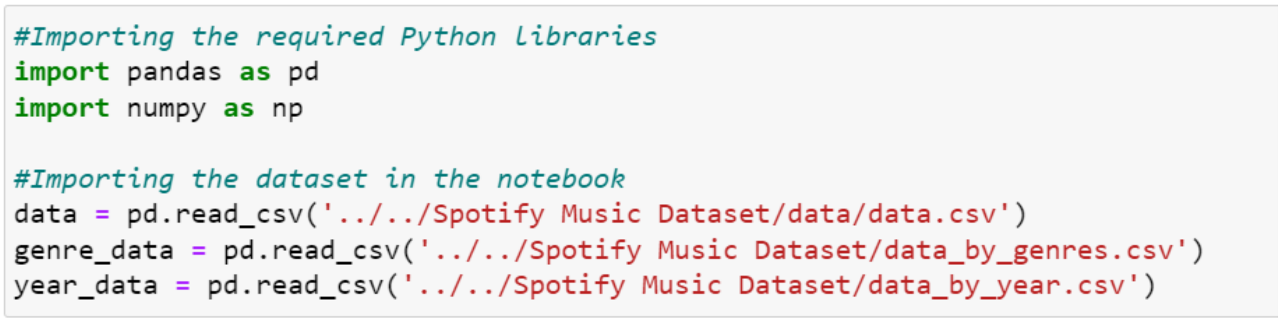
Data Exploration:

The process of collecting and processing data for this project involved gathering the necessary datasets from the Kaggle website, which serves as a vast repository of open-source data. We utilized the pandas library's read_csv() function to import the datasets into our analysis environment.

The project relies on three primary datasets, each containing critical information for our analysis. The first dataset, Data.csv, provides a comprehensive listing of Spotify songs, featuring essential variables such as song ID, valence, and liveness. The second dataset, genre_data.csv, contains information on each song's genre, along with all other song’s feature variables such as energy, speechiness, tempo and loudness.

Finally, the year_data.csv dataset offers insights into each song's release year, among song’s feature variables.

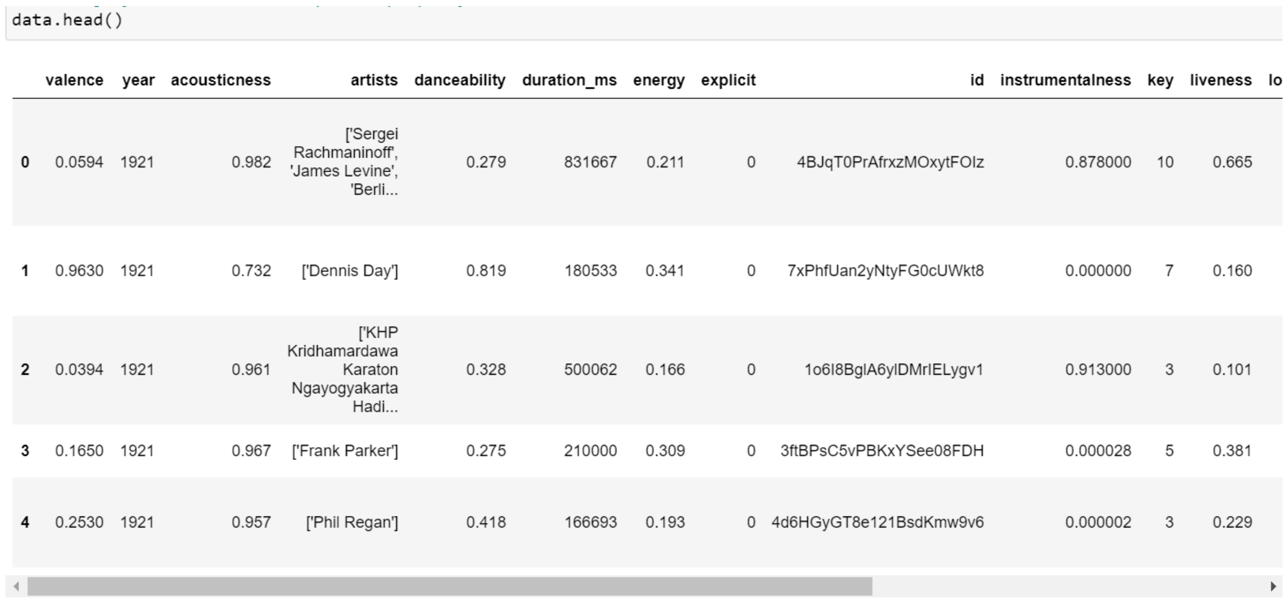
Code to import the all the datasets to Notebook:



```
#Importing the required Python libraries
import pandas as pd
import numpy as np

#Importing the dataset in the notebook
data = pd.read_csv('../../Spotify Music Dataset/data/data.csv')
genre_data = pd.read_csv('../../Spotify Music Dataset/data_by_genres.csv')
year_data = pd.read_csv('../../Spotify Music Dataset/data_by_year.csv')
```

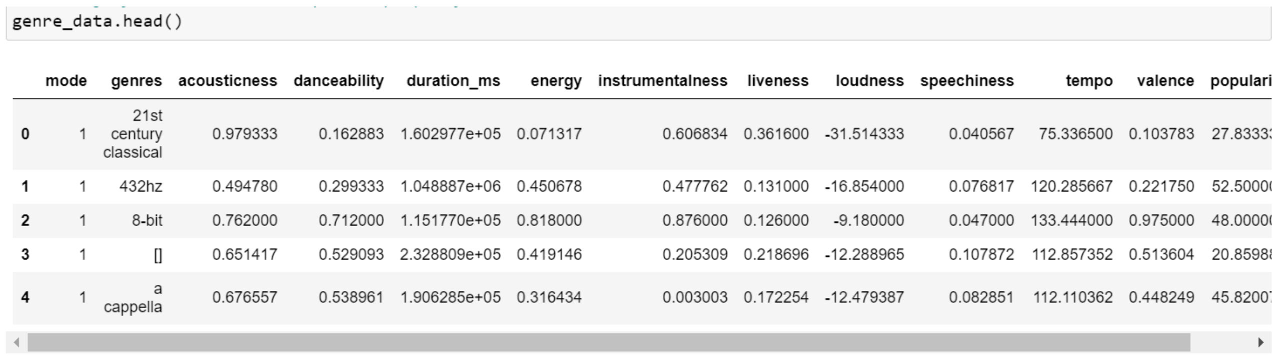
Music Features Data:



```
data.head()
```

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	lo
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665	
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160	
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101	
3	0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5	0.381	
4	0.2530	1921	0.957	['Phil Regan']	0.418	166693	0.193	0	4d6HGyGT8e121BsdKmw9v6	0.000002	3	0.229	

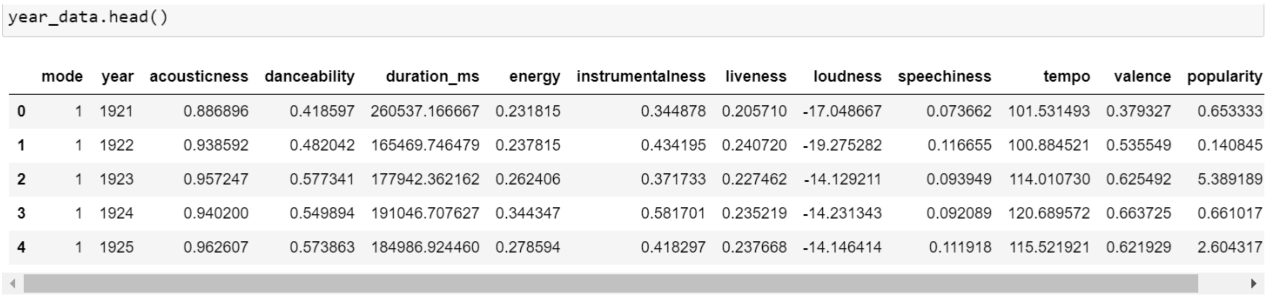
Genre Data:



```
genre_data.head()
```

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	populari
0	1	21st century classical	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514333	0.040567	75.336500	0.103783	27.8333
1	1	432hz	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854000	0.076817	120.285667	0.221750	52.5000
2	1	8-bit	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180000	0.047000	133.444000	0.975000	48.0000
3	1	[]	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288965	0.107872	112.857352	0.513604	20.8598
4	1	a cappella	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479387	0.082851	112.110362	0.448249	45.8200

Year Data:



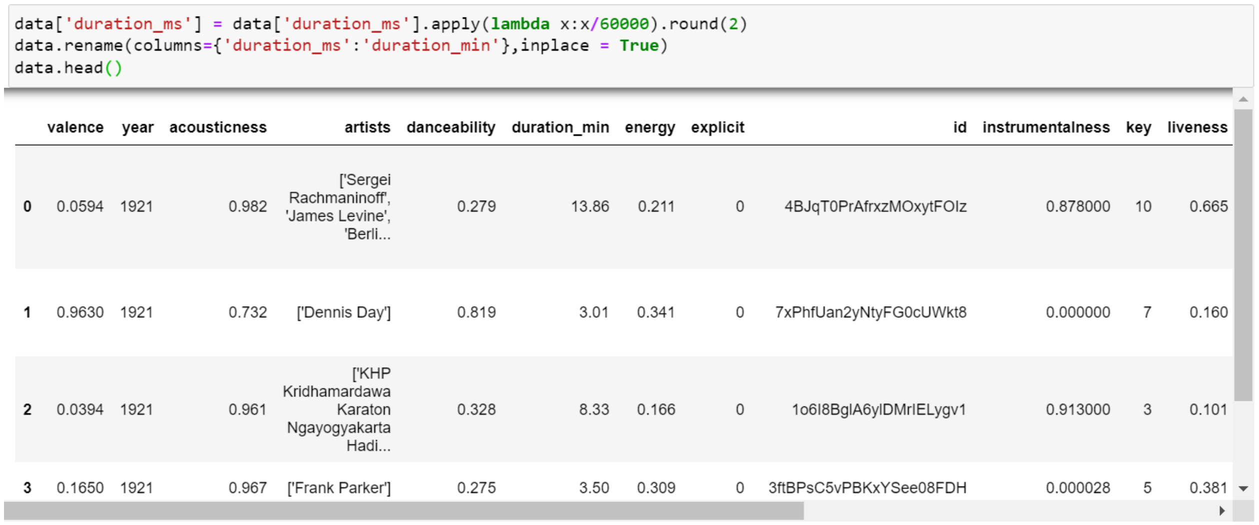
```
year_data.head()
```

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.344878	0.205710	-17.048667	0.073662	101.531493	0.379327	0.653333
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.434195	0.240720	-19.275282	0.116655	100.884521	0.535549	0.140845
2	1	1923	0.957247	0.577341	177942.362162	0.262406	0.371733	0.227462	-14.129211	0.093949	114.010730	0.625492	5.389189
3	1	1924	0.940200	0.549894	191046.707627	0.344347	0.581701	0.235219	-14.231343	0.092089	120.689572	0.663725	0.661017
4	1	1925	0.962607	0.573863	184986.924460	0.278594	0.418297	0.237668	-14.146414	0.111918	115.521921	0.621929	2.604317

Data Mining Tasks:

Data Transformation –

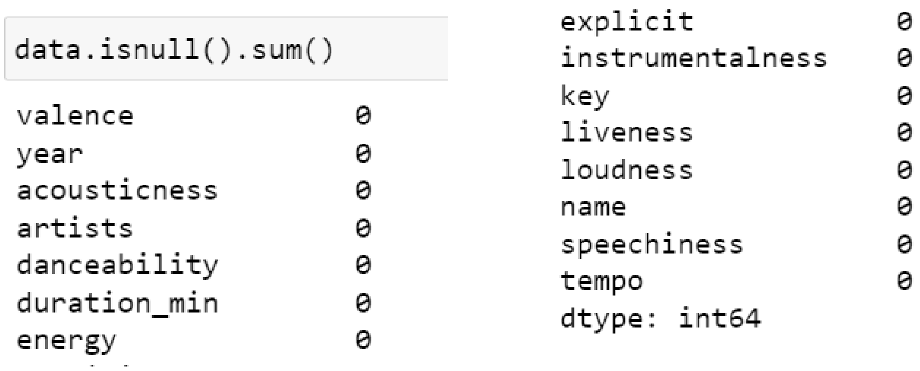
Changing the duration_ms column from ms min (i.e., duration_min):



```
data['duration_ms'] = data['duration_ms'].apply(lambda x:x/60000).round(2)
data.rename(columns={'duration_ms':'duration_min'},inplace = True)
data.head()
```

	valence	year	acousticness	artists	danceability	duration_min	energy	explicit	id	instrumentalness	key	liveness
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	13.86	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665
1	0.9630	1921	0.732	['Dennis Day']	0.819	3.01	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	8.33	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101
3	0.1650	1921	0.967	['Frank Parker']	0.275	3.50	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5	0.381

Checking for null values:



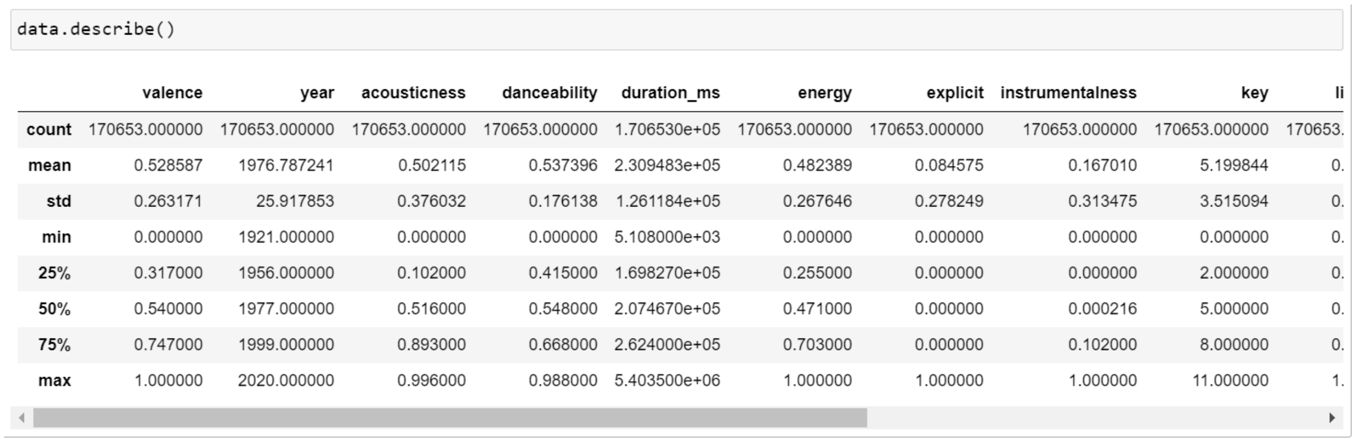
```
data.isnull().sum()
```

```
valence             0
year                0
acousticness        0
artists             0
danceability        0
duration_min        0
energy              0
```

```
explicit            0
instrumentalness    0
key                 0
liveness            0
loudness            0
name                0
speechiness         0
tempo               0
dtype: int64
```

Data Visualization:

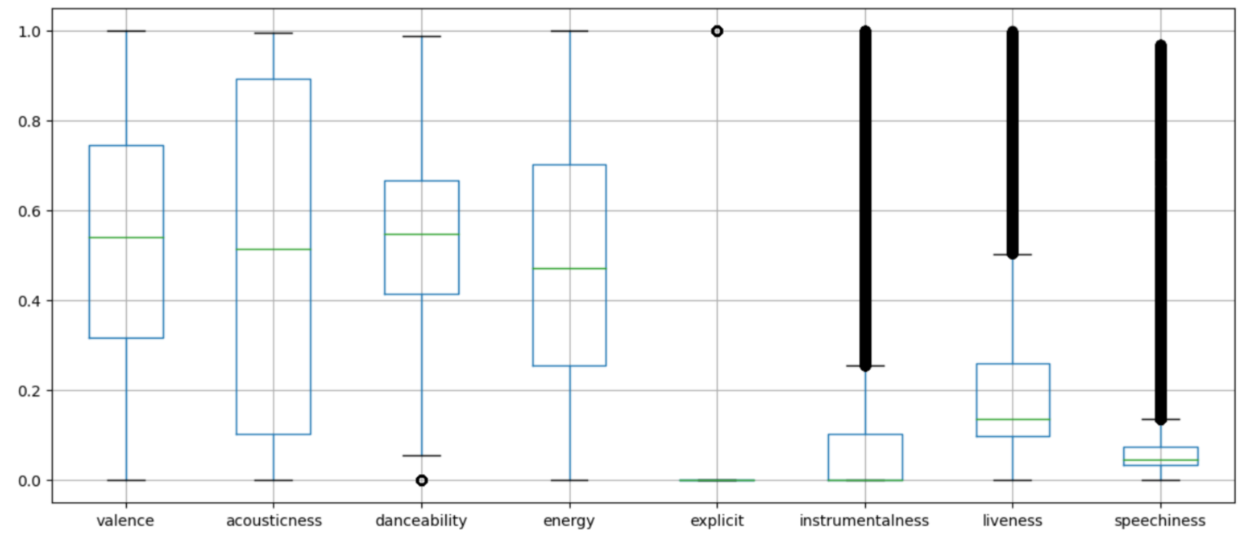
1. The below snippet of code shows the description and distribution of data overall.



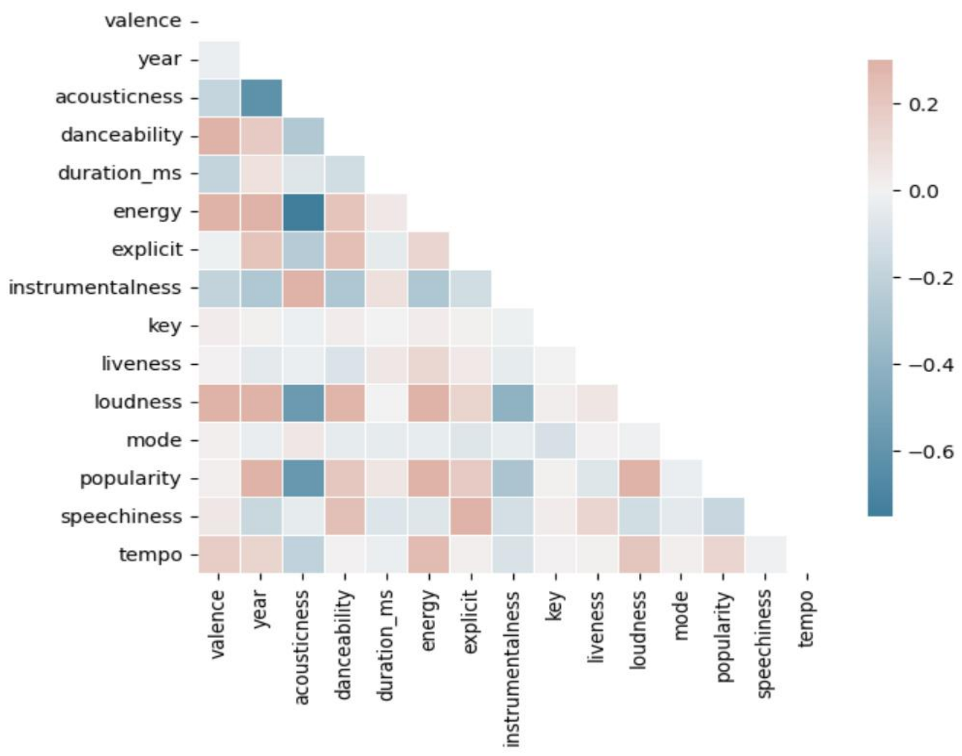
```
data.describe()
```

	valence	year	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	li
count	170653.000000	170653.000000	170653.000000	170653.000000	1.706530e+05	170653.000000	170653.000000	170653.000000	170653.000000	170653.
mean	0.528587	1976.787241	0.502115	0.537396	2.309483e+05	0.482389	0.084575	0.167010	5.199844	0.
std	0.263171	25.917853	0.376032	0.176138	1.261184e+05	0.267646	0.278249	0.313475	3.515094	0.
min	0.000000	1921.000000	0.000000	0.000000	5.108000e+03	0.000000	0.000000	0.000000	0.000000	0.
25%	0.317000	1956.000000	0.102000	0.415000	1.698270e+05	0.255000	0.000000	0.000000	2.000000	0.
50%	0.540000	1977.000000	0.516000	0.548000	2.074670e+05	0.471000	0.000000	0.000216	5.000000	0.
75%	0.747000	1999.000000	0.893000	0.668000	2.624000e+05	0.703000	0.000000	0.102000	8.000000	0.
max	1.000000	2020.000000	0.996000	0.988000	5.403500e+06	1.000000	1.000000	1.000000	11.000000	1.

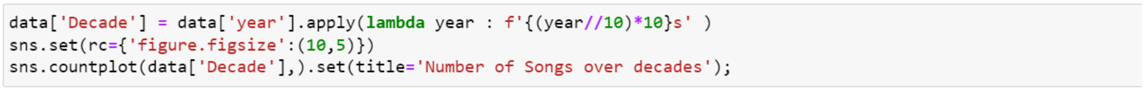
2. Box plot of various music features to see the outliers and range of values.



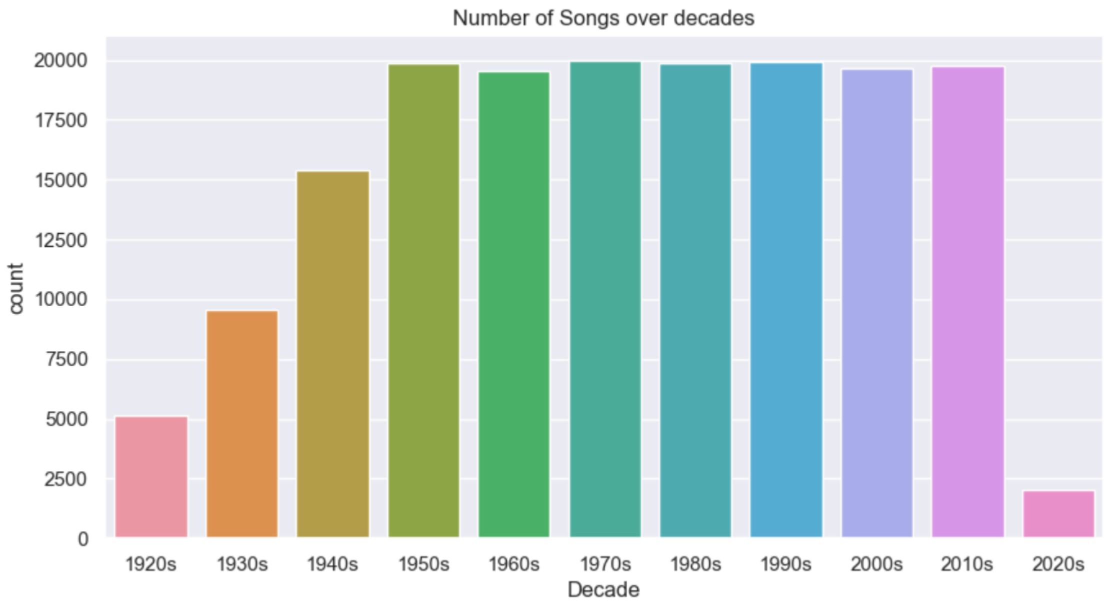
3. Correlation between the various music features of dataframe



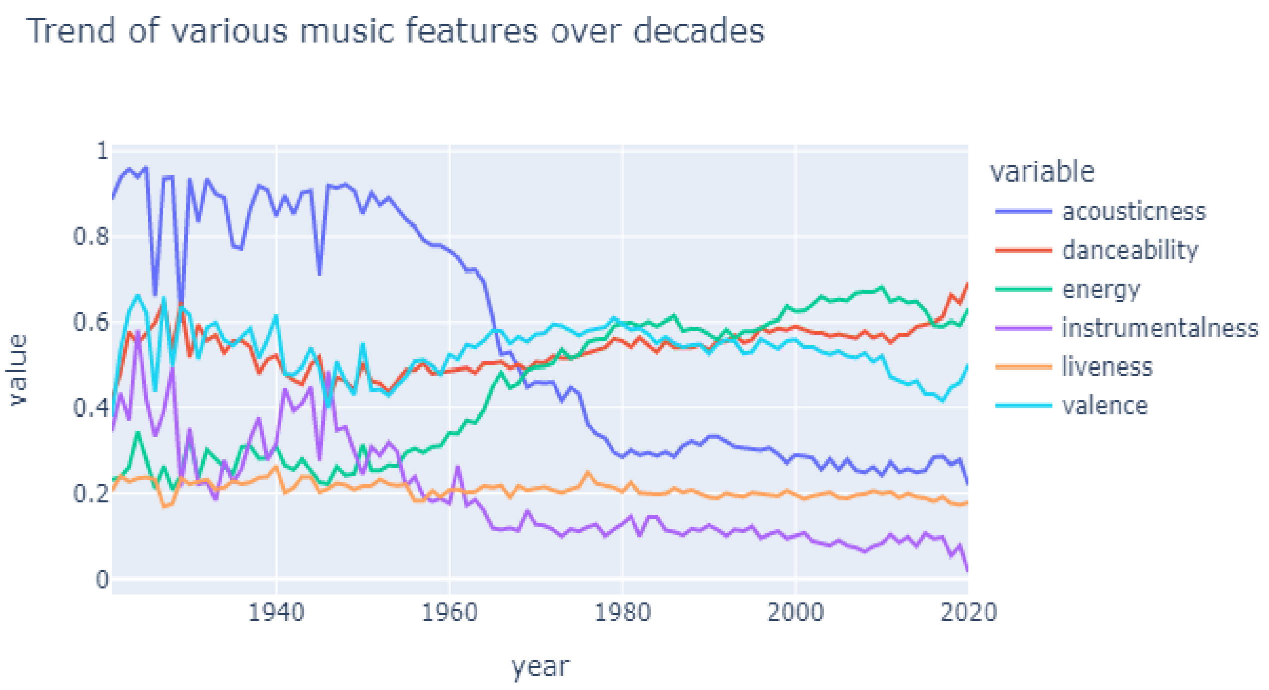
4. Number of songs released over the decades.



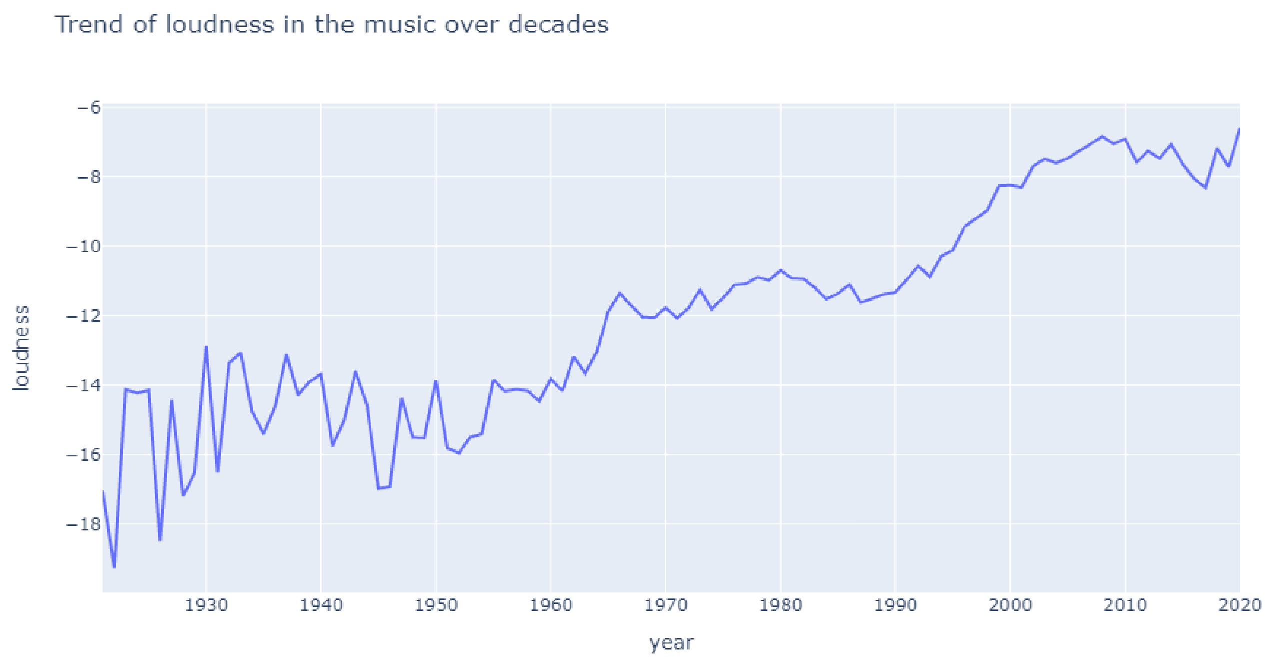
```
data['Decade'] = data['year'].apply(lambda year : f'{(year//10)*10}s' )
sns.set(rc={'figure.figsize':(10,5)})
sns.countplot(data['Decade'],).set(title='Number of Songs over decades');
```



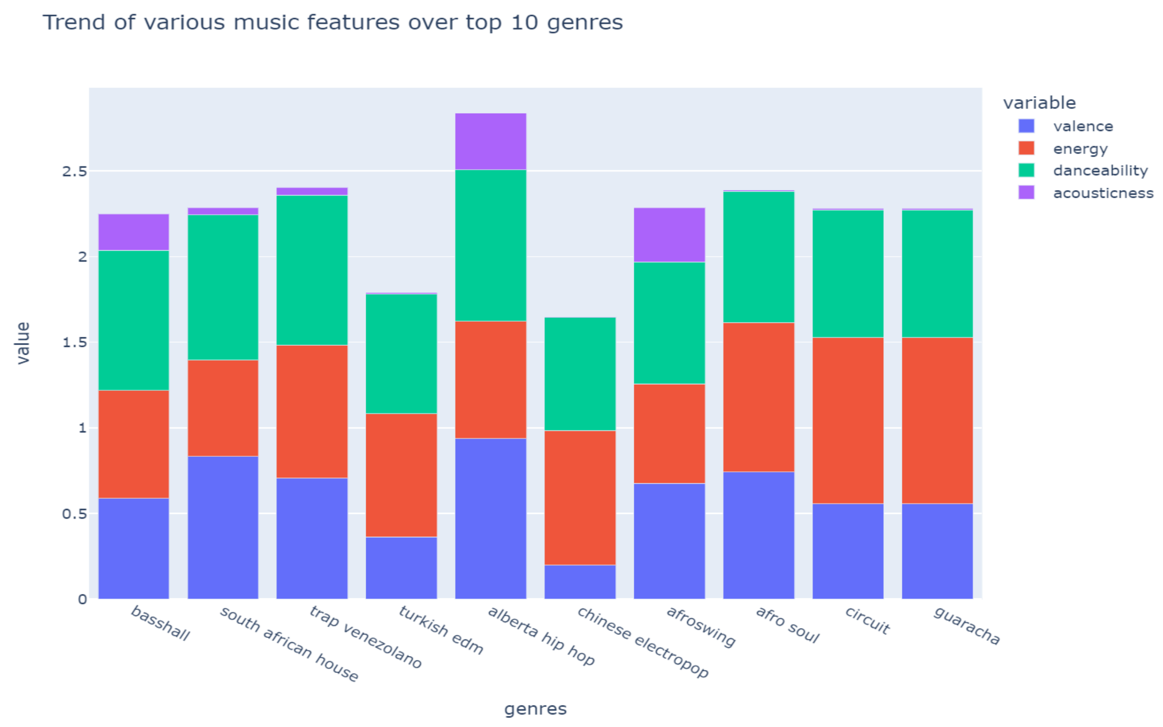
5. Trend of various music features over decades



6. Trend of loudness in the music over decades



7. Trend of various music features over top 10 genres



Conclusion:

a. Acousticness and Energy has a high correlation compared to other music features

b. The majority of the songs fall within the time range of 1950s to 2010s.

c. The energy in songs has risen over time, while the instrumentalness has declined.

d. The acousticness in songs has decreased over the years, particularly since the 1960s.

e. The trend of increasing loudness in songs is unmistakable and has reached its zenith in 2020.

f. Among the top 10 genres, energy and danceability are the most prominent characteristics.

Data Mining Models:

In constructing a collaborative filtering recommendation system for the product dataset, the initial step involves conducting sentiment analysis on the review dataset. The sentiment analysis process entails categorizing reviews into positive or negative sentiment classes (0 or 1) by considering factors like the review summary, voting on reviews, and the review text. This gives rise to a binary classification problem, prompting the exploration of various classification algorithms, including Logistic Regression, Naïve Bayes, Random Forest, XGBoost Classifier, and LSTM. The subsequent comparison of these models' performance will guide the selection of the most suitable one for the analysis.

To prepare the textual data for modeling, the review text will undergo tokenization and conversion into a vector format. This processed data will serve as input for the chosen classification model, enabling accurate sentiment analysis and, consequently, facilitating product recommendations based on customer feedback.

1. **Logistic Regression:**

Logistic regression is a commonly used binary classification algorithm that outputs conditional probability values. Although it provides predicted values like a regression model, it is an S-shaped and bounded function that gives probabilities of each observation falling into category '1'.

Benefits:

- Can be easily extended to multiple classes if needed
- Easier to implement and interpret and quick in classifying unknown records
- Interprets model coefficients as an indicator of feature importance

Limitations:

- Assumes a linear relationship between the dependent and independent variables
- Requires little or no multicollinearity between independent variables
- Can lead to overfitting the data if the number of observations is lesser than the number of features

2. **Naïve Bayes Classifier:**

Naïve Bayes classifier is a probabilistic ML model based on Bayes theorem used for classification tasks. Its assumptions are that the predictors are independent and equally important.

Benefits:

- Highly scalable as it scales linearly with the number of predictors.

- Simple and fast compared to exact bayes.
- Robust to noisy data.

Drawbacks:

- Requires a lot of records for reliable parameter estimates.
- Assumes independence of predictors which is not always true.
- Generates accurate propensity but not accurate probability.

3. **Random Forest:**

The Random Forest algorithm is an extension of the basic decision tree model. What sets it apart is its ensemble nature, as it combines the results of multiple decision trees, forming a 'Forest.' In contrast to a single decision tree, which considers all features when classifying an observation, each tree within the Random Forest model evaluates only a randomly chosen subset of the complete feature set.

This distinctive approach enhances the performance of the Random Forest model compared to a conventional decision tree. By randomly selecting feature subsets, individual trees in the forest can focus on isolating more crucial features, leading to an overall improvement in accuracy. This randomness and diversity among the trees contribute to the robustness and effectiveness of the Random Forest algorithm in handling various types of data and producing reliable predictions.

Benefits:

- It uses an ensemble learning technique and hence more stable and accurate than a simple decision tree.
- Can handle missing data automatically and is less impacted by noisy data.
- No need for feature scaling, such as standardization and normalization, as it can handle different feature scales.

Drawbacks:

- Time consuming to train as compared to decision trees.

- Building a Random Forest model can be computationally expensive as it involves creating a lot of decision trees.
- It can be complex with the increase in the number of trees and hyperparameters.

4. **XG Boost Classifier:**

Boosting is a machine learning technique that combines a set of weak learners to deliver improved prediction accuracy. XGBoost is a gradient boosting algorithm that has been optimized for speed and performance. This library is parallelized, which means it can run on a cluster of GPUs or a network of computers. It has internal parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, and much more.

Benefits:

- Requires less feature engineering and is less prone to overfitting.
- Works well with small, big, complicated data and data with subgroups
- Parallel processing through Scikit learn library's n-thread hyper-parameter.

Drawbacks:

- Sensitive to outliers and not highly scalable
- Doesn't work well with sparse, dispersed, and unstructured data.
- Possible overfitting if parameters are not tuned properly.

5. **Long Short-term Memory:**

LSTM is a type of artificial neural network architecture used in deep learning. It was developed as an improved version of RNN to address the issue of vanishing gradients. Unlike standard feedforward neural networks, LSTM includes feedback connections. Instead of retaining all data like a standard recurrent neural network, LSTM only stores short-term memory of the data.

Benefits and Drawbacks of LSTM:

Benefits:

- Models long-term sequence dependencies effectively.

- More robust to short memory problem compared to 'Vanilla' RNNs.

Drawbacks:

- Increases computing complexity due to more parameters to learn compared to RNN.
- Requires higher memory than 'Vanilla' RNNs due to the presence of multiple memory cells.

In the recommendation system, we will use KNN for item-based collaborative filtering and recommend the top 3 similar items.

6. **K-Nearest Neighbors:**

K-nearest-neighbor algorithm (k-NN) estimates the likelihood of a data point belonging to a group based on the group membership of its nearest neighbors. It is a lazy learner algorithm, which means it does not build a model until a query is performed.

Benefits:

- It is faster than other algorithms as it does not require a training period.
- New data can be easily added without impacting the accuracy of the algorithm.
- It is easy to implement with only two required parameters: the value of K and the distance function.

Drawbacks:

- The algorithm becomes slow and inefficient in large datasets.
- It does not work well with high-dimensional data.
- Feature scaling is required before applying the algorithm to a dataset to prevent generating wrong predictions.

7. **KNN Item based collaborative filtering:**

Collaborative filtering relies on the user's past preferences to recommend items and does not require extensive product features. Both users and items are represented by embedding or feature vectors, which cluster them together. This creates separate clusters for items and users, without the need for explicit feature engineering.

Benefits:

- Automatic learning of embeddings means domain knowledge is not required.
- The model can suggest new interests to users based on similar users' preferences.
- Only the feedback matrix is needed to train a matrix factorization model, and contextual features are not required.

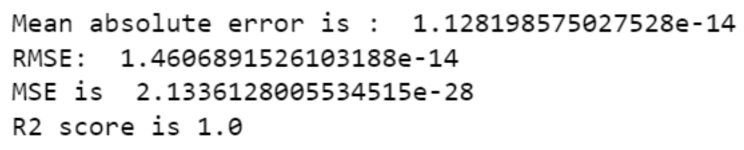
Drawbacks:

- The system can't create embeddings or make recommendations for items that were not seen during training, known as the cold-start problem.
- Including side features, such as country or age, can improve the model, but it may not be easy to incorporate them into the algorithm.

Performance Evaluation:

<u>Linear Regression Model –</u>

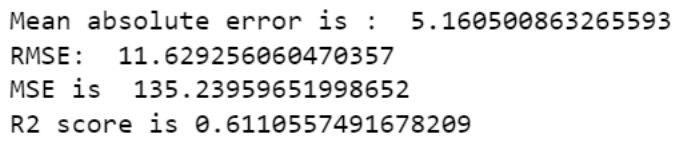
The linear regression graph fits the data well, with an R2 of 1. The root mean squared error (RMSE) and mean absolute error (MAE) are both low, but not zero. This indicates that the model is accurate, but there is still some error in the predictions.



```
Mean absolute error is :  1.128198575027528e-14
RMSE:  1.4606891526103188e-14
MSE is  2.1336128005534515e-28
R2 score is 1.0
```

<u>Polynomial Regression Model –</u>

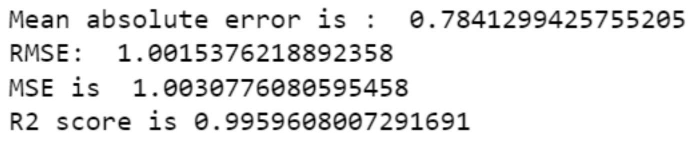
The polynomial regression model has higher RMSE and MAE values than the linear regression model. This is because the polynomial regression model is more complex and can fit the data more closely. However, it is important to note that the polynomial regression model may be more accurate on unseen data. This is because the polynomial regression model can fit the data more closely, which can help to reduce overfitting.



```
Mean absolute error is :  5.160500863265593
RMSE:  11.629256060470357
MSE is  135.23959651998652
R2 score is 0.6110557491678209
```

<u>Lasso Regression Model –</u>

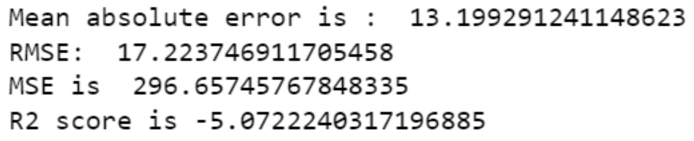
Error is close to 1, but not 0.



```
Mean absolute error is :  0.7841299425755205
RMSE:  1.0015376218892358
MSE is  1.0030776080595458
R2 score is 0.9959608007291691
```

<u>KNN Regression Model –</u>

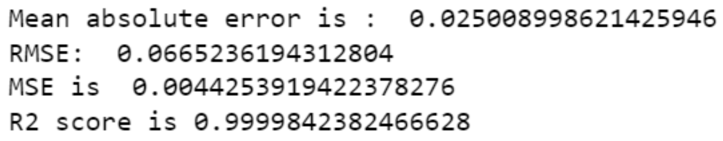
It gives really high values of error.



```
Mean absolute error is :  13.199291241148623
RMSE:  17.223746911705458
MSE is  296.65745767848335
R2 score is -5.0722240317196885
```

<u>Random Forest Regression Model –</u>

This is the last model, and it also has higher RMSE AND MAE values than Linear Regression.



```
Mean absolute error is :  0.025008998621425946
RMSE:  0.0665236194312804
MSE is  0.0044253919422378276
R2 score is 0.9999842382466628
```

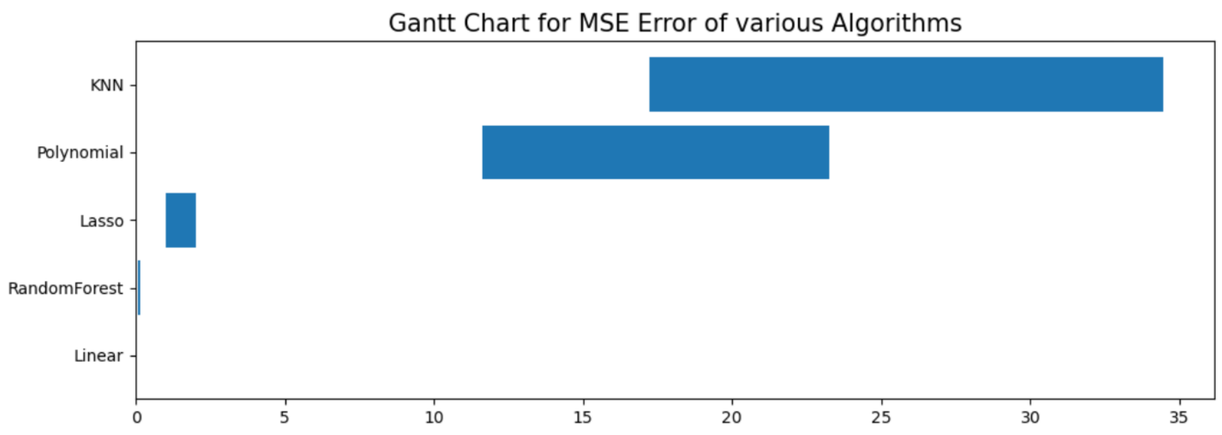
Feature Importance:

```
pd.DataFrame({'Column': X_train.columns, 'Feature Importance': model.feature_importances_})
```

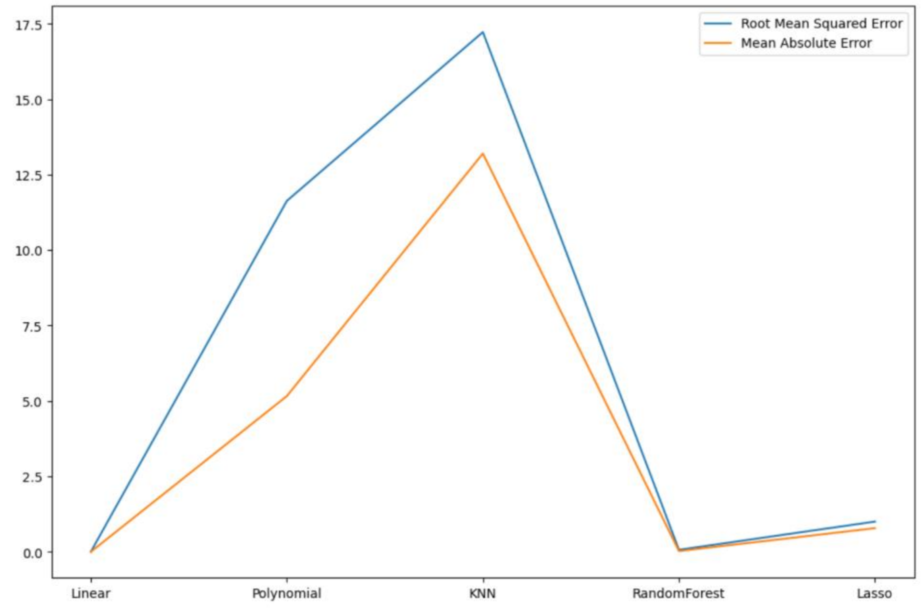
	Column	Feature Importance
0	mode	3.774427e-07
1	acousticness	4.884888e-06
2	danceability	3.347732e-06
3	duration_ms	3.157623e-06
4	energy	2.460758e-06
5	instrumentalness	3.508726e-06
6	liveness	2.477572e-06
7	loudness	2.982998e-06
8	speechiness	3.296016e-06
9	tempo	2.708022e-06
10	valence	2.443411e-06
11	popularity	9.999641e-01
12	key	3.204379e-06
13	cluster	1.022491e-06

Gantt Chart

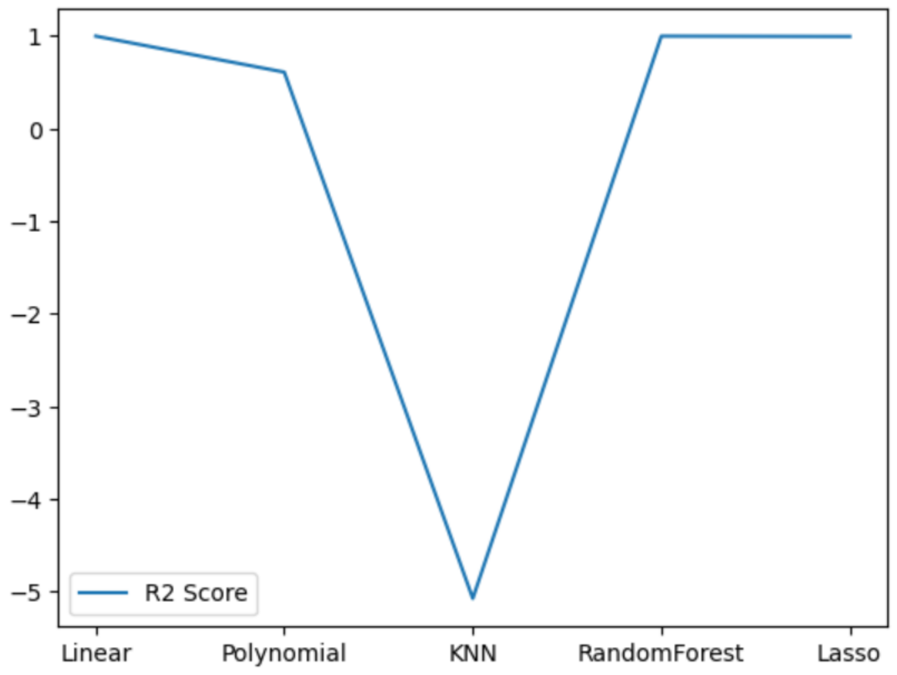
```
plt.figure(figsize=(12,4))
plt.barh(y=df_algo_error['Algorithm'], width=df_algo_error['Error Values'],left=df_algo_error['Error Values'])
plt.title('Gantt Chart for MSE Error of various Algorithms', fontsize=15)
plt.show()
```



Mean Aboslute Error and RMSE for various Regression Algorithms :



R2 Score for different Algorithms:



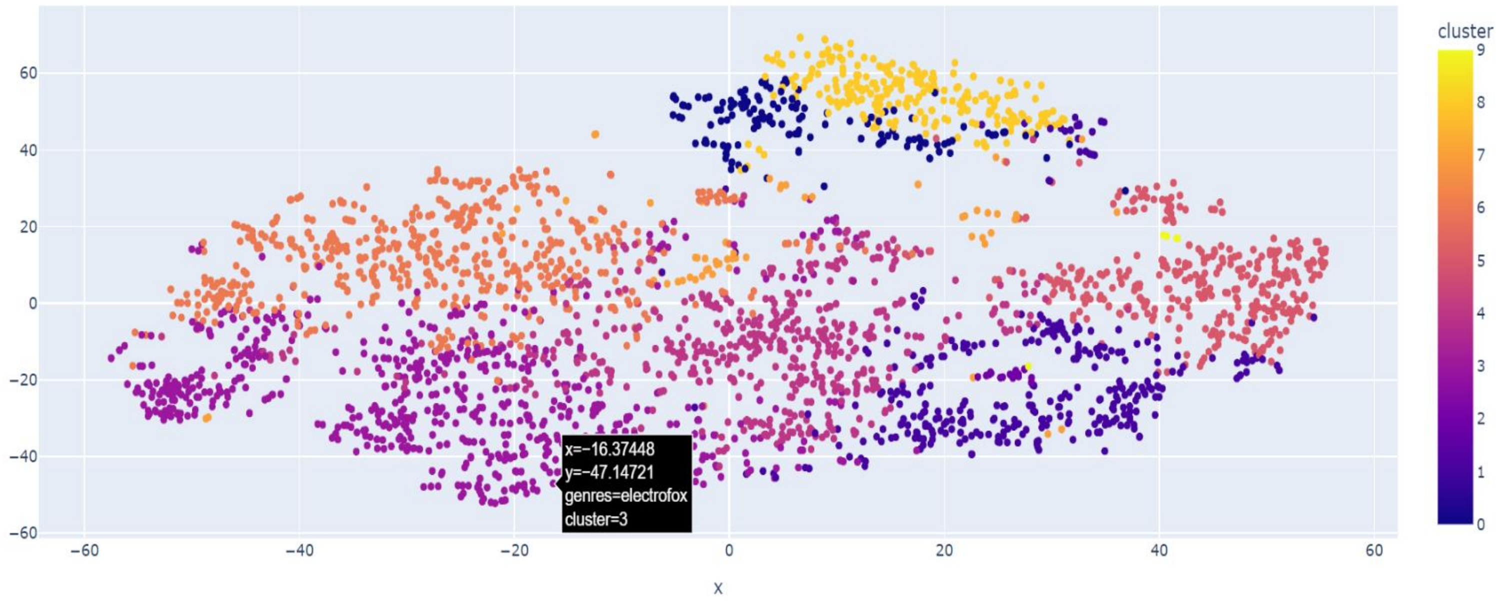
Other Data Exploration and Modeling:

To delve deeper into this dataset, we applied a clustering algorithm. KMeans was opted for clustering due to several reasons:

1. **Simplicity:** KMeans is straightforward and easy to comprehend, serving as an excellent starting point for clustering. Its implementation is uncomplicated, and the number of clusters can be easily specified.
2. **Scalability:** KMeans proves effective for large datasets with numerous features, demonstrating scalability. Its linear time complexity makes it particularly efficient for handling substantial datasets.
3. **Flexibility:** KMeans offers versatility by accommodating various distance metrics and cluster initialization methods, allowing for adaptability in the clustering process. It demonstrates capability in handling both numerical and categorical data with appropriate preprocessing.

When selecting a clustering algorithm, it is crucial to consider the data's characteristics and the specific application requirements. Nevertheless, K-Means stands out as a favorable choice due to its simplicity, scalability, flexibility, and established reliability. In this instance, the straightforward K-means clustering algorithm is employed to categorize genres in the dataset into ten clusters based on the numerical audio features of each genre. This exploration can subsequently inform future song recommendations for users.

This exploration can further be used to recommend songs to users in future.



Project Results:

The project's analysis focused on evaluating various regression models using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and R2 score. The linear regression model demonstrated a robust fit with an R2 score of 1, low MAE, and RMSE, indicating accuracy with minimal error. Conversely, the polynomial regression model, though more complex, exhibited higher RMSE and MAE values, suggesting potential overfitting while closely fitting the data. The Lasso regression model showed an error close to 1, and the KNN regression model resulted in significantly high error values. The Random Forest regression model, while the last in the analysis, displayed higher RMSE and MAE compared to Linear Regression. These outcomes provide a comprehensive understanding of model performance and trade-offs between complexity and accuracy.

In conclusion, we have evaluated the performance of all models in our dataset and determined that the linear regression model is the best fit for predicting song popularity with minimal errors.

Impact of the Project Outcomes:

The data mining effort, focused on metrics like MAE, RMSE, MSE, and R2 score, yields valuable insights into the predictive capabilities of various regression models. The project's outcomes emphasize the careful consideration required in balancing model complexity and accuracy. Linear regression offers a straightforward and accurate prediction, while polynomial regression, despite introducing complexity, might provide a better fit for unseen data. The findings also illuminate the limitations of models like KNN and Random Forest within this context. Overall, the project contributes to informed decision-making, guiding future regression model selections based on specific data characteristics and project objectives.