MUSIC RECOMMENDATION SYSTEM

ANANYA RAO

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In today's digital music world, we have access to an overwhelming amount of songs. While this offers a lot of choices, it can be tricky to find music that truly suits our tastes. Traditional methods of finding new songs might not always capture the diversity of what's out there.

This challenge can be tackeld using the power of machine learning. Imagine a system that learns what kind of music you like based on what you listen to and then suggests songs you're likely to love. The goal is to make discovering new music a personalized and enjoyable experience, going beyond generic recommendations and helping users connect with the music that speaks to them.

Recommendation systems, also known as recommender systems, are a type of software application or algorithm that provides personalized suggestions or recommendations to users. These systems are widely used in various online platforms to enhance user experience and engagement.

There are several types of recommendation systems, and they can be categorized into three main approaches: collaborative filtering, content-based filtering, and hybrid methods.

COLLABORATIBE FILTERING:

The collaborative filtering method is based on gathering and analyzing data on user's behavior. This includes the user's online activities and predicting what they will like based on the similarity with other users.

Two kinds of collaborative filtering techniques used are:

- User-User collaborative filtering
- Item-Item collaborative filtering

One of the main advantages of this recommendation system is that it can recommend complex items precisely without understanding the object itself. There is no reliance on machine analyzable content. It calculates item similarity based on user interactions and recommends items that are similar to those the user has liked or engaged with in the past.Common similarity metrics include cosine similarity and Pearson correlation.

CONTENT-BASED FILTERING:

Content-based filtering methods are based on the description of a product and a profile of the user's preferred choices. In this recommendation system, products are described using keywords, and a user profile is built to express the kind of item this user likes.

Effective content-based filtering requires a well-defined representation of item features. This representation can be a set of keywords, tags, or numerical values associated with each item. Content-based filtering tends to produce diverse recommendations because it focuses on item features rather than user-user similarities. Similarity between items or between items and the user profile is calculated using various similarity metrics, such as cosine similarity or Euclidean distance.

HYBRID METHODS:

Hybrid methods in recommendation systems combine multiple recommendation techniques, such as collaborative filtering, content-based filtering, and sometimes other approaches, to overcome the limitations of individual methods and provide more accurate and diverse recommendations. Hybrid methods aim to leverage the strengths of different recommendation techniques and enhance overall system performance.

Feature combination hybrid methods merge the features extracted from different recommendation techniques into a single model. For example, content-based and collaborative features may be combined to create a unified user-item interaction model.

1.2 PROBLEM

The music industry has witnessed an explosion in the volume of available music, making it challenging for users to discover new songs or artists that align with their preferences. Traditional methods of music consumption often rely on manual selection, leading to limited exposure to diverse music genres. This problem highlights the need for an intelligent music recommendation system that can enhance user experience by offering personalized and relevant music suggestions. This problem highlights the need for an intelligent music recommendation system that can enhance user experience by offering personalized and relevant music suggestions.

1.3 RESEARCH QUESTION

How can machine learning techniques be leveraged to build an effective music recommendation system that provides users with personalized song suggestions based on their preferences and historical interactions?

1.4 PURPOSE

The purpose of this project is to revolutionize the landscape of music recommendation systems through the application of cutting-edge machine learning techniques. Our primary objective is to create a system that not only understands individual user preferences but also adapts in real-time to provide highly personalized and diverse music recommendations. By delving into advanced algorithms, including collaborative filtering and content-based recommendation models, we seek to optimize the user experience and transform music exploration into a dynamic and engaging process. Through the exploration of diverse genres and continuous learning from user interactions, our project aspires to redefine how users discover and connect with music, offering a transformative platform that aligns with their evolving tastes and preferences. This project serves as a pivotal step towards enhancing the user-centric nature of music recommendation systems and contributing to the broader field of intelligent digital content discovery.

1.5 GOALS

The goal of the music recommendation system is to enhance user satisfaction and engagement by:

- Offering personalized song recommendations.
- Increasing the diversity of music exploration.
- Improving the overall user experience in discovering new and relevant music.

By achieving these goals, the recommendation system intends to create a more enjoyable and efficient music discovery process for users. This improves the user interface by making it more predictive and enjoyable.

CHAPTER 2

LITERATURE SURVEY

Recommendation systems, including collaborative filtering, content-based filtering, and hybrid methods, play a crucial role in addressing information overload by providing personalized suggestions. Collaborative filtering, particularly, suggests items based on the preferences of users with similar tastes, relying on user-user similarities.

We were awestruck with Spotify's recommendation engine. We always wondered how Spotify manages to recommend that perfect song, playlist, or even that 'daily mix'. We now have more technology than ever before to ensure that if you're the smallest, strangest musician in the world, doing something that only 20 people in the world will dig, we can now find those 20 people and connect the dots between the artist and listeners. This has been the motivation for this project to use various machine learning techniques and to develop a music recommendation engine similar to that of Spotify, which takes music listening experience to another level. Music Recommendation Systems.

Various methods for developing recommendation systems have recently been created, including collaborative filtering, content-based filtering, and hybrid filtering. The collaborative filtering approach is the most developed and widely used. Collaborative filtering suggests things by locating other users who have similar tastes to the current user and using their recommendations. Collaborative recommender systems have been used in a variety of settings. problems – new items and new users.

The common characteristics in these systems are constant when using users' preferences compared with users' context (location, mood, weather, etc.). For instance, in the library when people are sitting there maybe they need quiet and melodious music to listen according to the environment where they are in. Last.fm, All music, Spotify, Pandora and Shazam are commercial music recommendation systems which are considered to be excellent systems by focusing on the music already played in order to help the users to find more music. Users are able to connect to a web-based music streaming service to access the recommendations. All the tracks that are played on this stream are recommended. It is Based on songs or artists which users either upload from your iTunes playlists or add as favourites on the site where users start managing their library of music with tags and keep tracking of the music the friends who listening to and getting multiple recommendations per song played.

Meanwhile, many researchers have used social media (Twitter & Facebook) to identify user's mood (tension, depression, anger, vigor, fatigue, confusion) and also identify user's personality (openness, conscientiousness, extraversion, 21 agreeableness, neuroticism) where these are very important factors which influence on user's music taste (Wang et al., 2014; Roberts et al., 2012; Pandarachalil et al., 2015; Ross et al., 2009; Bachrach et al., 2012; Back et al., 2010) and also contextual features (location & event) can lead to different emotional effects due to objective features of the situation or subjective perceptions of the listeners (Scherer et al., 2001).

The people may want to listen to music which has the same mood of them when they are in specific mood and in contrast the people want to listen to different kind of music which encourage them to enhance their mood and this thing depend on the psychological studies and therefore, the author produced a contextual mood-based music recommender system which is able to regulate the driver's mood and also try to put the driver in a positive mood when driving because listening to the music while driving has always been one of the most favourite activities carried out by people. Finally, similarly, active learning approaches suffer from various limitations.

CHAPTER 3

TECHNOLOGIES USED

3.1 MACHINE LEARNING ALGORITHMS

Machine learning algorithms are computational models that enable machines, particularly computers, to learn from data and make predictions or decisions without being explicitly programmed. These algorithms utilize statistical techniques to recognize patterns, learn from experience, and improve their performance over time as they are exposed to more data.

Machine learning algorithms find applications across various domains, including image and speech recognition, natural language processing, recommendation systems, and autonomous vehicles. The selection of the appropriate algorithm depends on the specific task, the nature of the data, and the desired outcome. As machine learning continues to advance, researchers and practitioners explore new algorithms and techniques to tackle complex problems and improve predictive accuracy.

3.1.1 MACHINE LEARNING LIBRARIES

NumPy (Numerical Python):

NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

Multi-dimensional arrays: NumPy provides an array object that represents arrays of numerical data. These arrays can be one-dimensional, two-dimensional, or even multi-dimensional.

Mathematical functions: NumPy includes a wide range of mathematical functions for performing operations on arrays, such as element-wise operations, linear algebra, random number generation, and more.

Broadcasting: NumPy allows operations between arrays of different shapes and sizes through a mechanism called broadcasting.

Integration with other libraries: Many other libraries in the scientific Python ecosystem, such as SciPy, scikit-learn, and Matplotlib, build on NumPy arrays.

Pandas:

Pandas is a data manipulation and analysis library. It provides data structures for efficiently storing and manipulating large datasets, along with tools for data cleaning, exploration, and analysis.

DataFrame: The core data structure in Pandas is the DataFrame, a two-dimensional table with labeled axes (rows and columns). It is similar to a spreadsheet or SQL table.

Series: Pandas also provides a one-dimensional labeled array called a Series, which can be thought of as a single column of a DataFrame.

Data alignment and missing data handling: Pandas automatically aligns data based on label and efficiently handles missing or NaN (Not a Number) values.

GroupBy: Allows splitting data into groups based on some criteria and then applying a function to each group independently.

Time series functionality: Pandas has powerful tools for working with time-series data.

3.2 t-SNE (t-distributed stochastic neighbor embedding)

t-SNE is a technique used to visualize high-dimensional data in a lower-dimensional space (typically 2D or 3D) while preserving the pairwise similarities between data points as much as possible. It is particularly useful for visualizing clusters and patterns in complex datasets.The algorithm works by modeling the similarity between pairs of data points in the high-dimensional space and the low-dimensional space. It focuses on preserving the local relationships and clusters of points. t-SNE is effective at revealing the structure of the data but is computationally intensive.

The algorithm starts by computing pairwise similarities between data points in the original space using a Gaussian distribution. It then seeks to find a mapping to a lower-dimensional space where these similarities are preserved as much as possible. Through iterations, t-SNE adjusts the positions of data points in the lower-dimensional space, minimizing the mismatch between pairwise similarities in the high and low-dimensional representations. The final embeddings reveal clusters and local structures, making t-SNE a valuable tool for data exploration and visualization. However, users should be mindful of its sensitivity to parameters and interpret visualizations cautiously. . The result is a scatter plot where each point represents a data instance, and the proximity of points in the plot reflects their similarity in the high-dimensional space. Clusters of similar points will be grouped together.

3.3 LINEAR REGRESSION

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features. When the number of the independent feature, is 1 then it is known as Univariate Linear regression, and in the case of more than one feature, it is known as multivariate linear regression. The relationship is assumed to be linear, meaning that changes in the independent variables are associated with a constant change in the dependent variable. In a simple linear regression (one independent variable), the graph is a scatter plot with the dependent variable on the y-axis and the independent variable on the x-axis. The regression line represents the best-fit line through the data points.

In multiple linear regression (more than one independent variable), it becomes challenging to visualize in a two-dimensional graph. However, you can create partial regression plots or use 3D plots to visualize the relationship between the dependent variable and each independent variable separately.

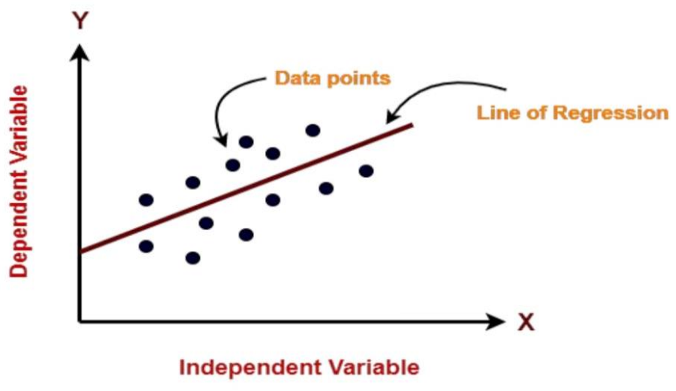


Fig 3.3.1 Linear Regression

3.4 CONTENT BASED FILTERING

Content-Based Filtering is a recommendation technique used in recommender systems to suggest items to users based on the characteristics of the items and the preferences expressed by the user. This method relies on analyzing the content of the items and building a profile for each user to recommend items that match their preferences.

Working of content-based filtering:

Item Representation:

Each item in the system is described by a set of features or attributes. For example, in the context of movies, features might include genre, director, actors, and keywords.

User Profile Creation:

A user profile is created based on the features of items that the user has interacted with or liked in the past. The profile reflects the user's preferences for specific features.

Similarity Calculation:

Similarity between items or between items and the user profile is calculated using various similarity metrics. Common metrics include cosine similarity, Euclidean distance, or Pearson correlation.

Recommendation Generation:

Recommendations are generated by selecting items that are most similar to the user's profile. Items with features closely aligned with the user's preferences are recommended.

3.5 MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. It provides a variety of functions to visualize data and create static, animated, and interactive plots. Matplotlib is widely used for tasks ranging from simple line plots to complex visualizations. Basic plots of Matplotlib are Line plot, Bar plot, Scatter Plot, Histogram, Box plot.

3.5.1 Bar Graph

A bar graph is a graphical representation of data in which rectangular bars of varying lengths are used to represent different categories or groups. The length of each bar corresponds to the quantity or value it represents. Bar graphs are effective for displaying and comparing categorical data, making them a popular choice for visualizing information.

Bar graph is generated using plt.bar() in matplotlib

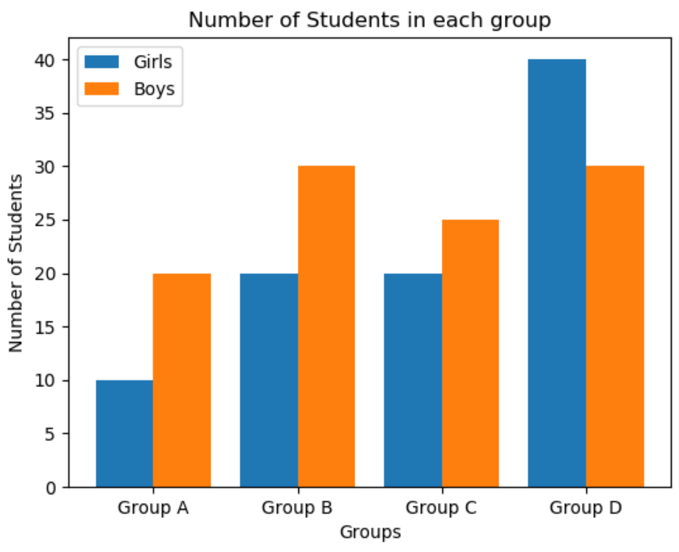


Fig 3.5.1.1 Bar Graph

3.5.2 HISTOGRAM USING MATPLOTLIB

A histogram is a graphical representation of the distribution of a dataset. It is a way to visualize the underlying frequency distribution of a set of continuous or discrete data. In a histogram, data is divided into intervals called "bins." The bins are represented as bars, where the height of each bar corresponds to the frequency or count of data points falling into that bin. Histograms are commonly used for visualizing the distribution of continuous data (e.g., heights, weights) and discrete data (e.g., counts of occurrences). The height of each bar can represent either the absolute frequency (number of data points) in that bin or the relative frequency (proportion or percentage). In the latter case, the histogram represents a probability density.

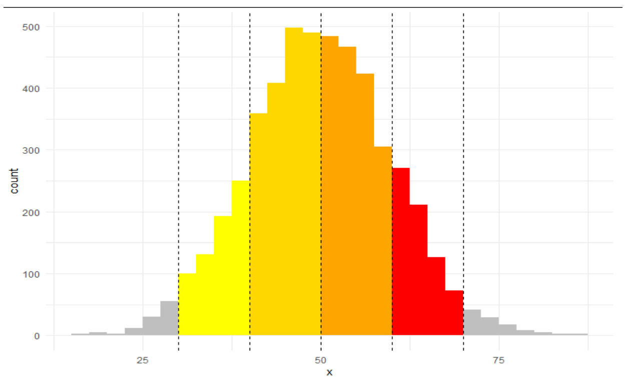


Fig 3.5.2.1 Histogram

3.5.3 SCATTER PLOT USING MATPLOTLIB

A scatter plot is a graphical representation of individual data points in a two-dimensional space. Each point on the plot corresponds to the values of two variables, making it useful for visualizing the relationship between them. In a scatter plot, each data point is represented by a dot or marker on the graph. The position of the dot is determined by the values of two variables—one along the x-axis (horizontal) and the other along the y-axis (vertical). Clusters of points on a scatter plot may suggest the presence of subgroups or patterns within the data. Analyzing these clusters can provide insights into the underlying structure of the dataset.

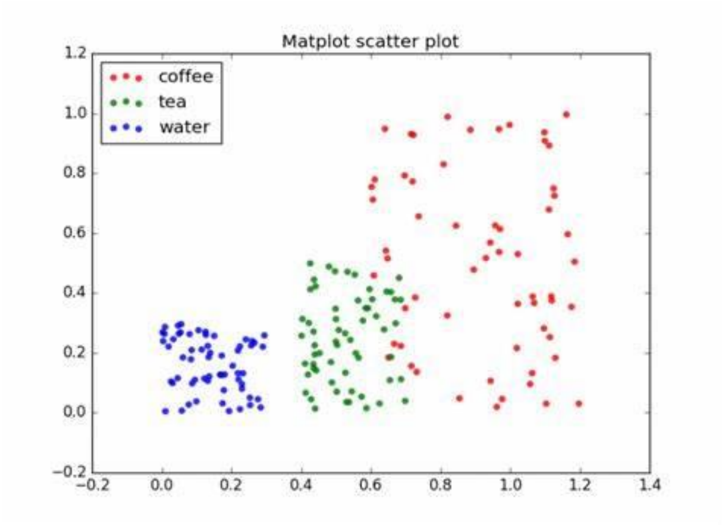


Fig 3.5.3.1 Scatter Plot

3.6 SEABORN

Seaborn is a statistical data visualization library in Python that is built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations and enhances the aestheSeaborn provides a range of statistical plots, including scatter plots, line plots, bar plots, histograms, box plots, violin plots, pair plots, and more.

It has built-in themes and color palettes that improve the visual appeal of plots compared to the default Matplotlib styles. Seaborn works well with Pandas DataFrames, making it convenient for visualizing data stored in tabular formats of plots.

Seaborn provides a variety of common plots for data visualization, some of them are:

Scatter Plot :

Displays individual data points in a two-dimensional space, useful for visualizing relationships between two continuous variables.

Line Plot :

Connects data points with lines, often used to show trends or patterns in data over a continuous variable (e.g., time).

Box Plot:

Illustrates the distribution of a continuous variable, providing information about the median, quartiles, and potential outliers.

Violin Plot :

Combines aspects of box plots and kernel density plots, providing insights into the distribution and density of a continuous variable.

Histogram :

Displays the distribution of a single variable, showing the frequency of values within specified bins.

Heatmap:

Visualizes the correlation matrix of variables, with color intensity representing the strength and direction of correlations.

Count Plot:

Description: Shows the count of observations in each category of a categorical variable, creating a bar plot without the need for aggregation.

3.6.1 COUNT PLOT

A count plot in Seaborn is a categorical plot that shows the count of observations in each category of a categorical variable. It is essentially a bar plot where the height of each bar represents the number of occurrences of a particular category. Count plots are useful for visualizing the distribution of categorical data and understanding the frequency of different categories within a dataset.

The primary function for creating a count plot in Seaborn is sns.countplot()

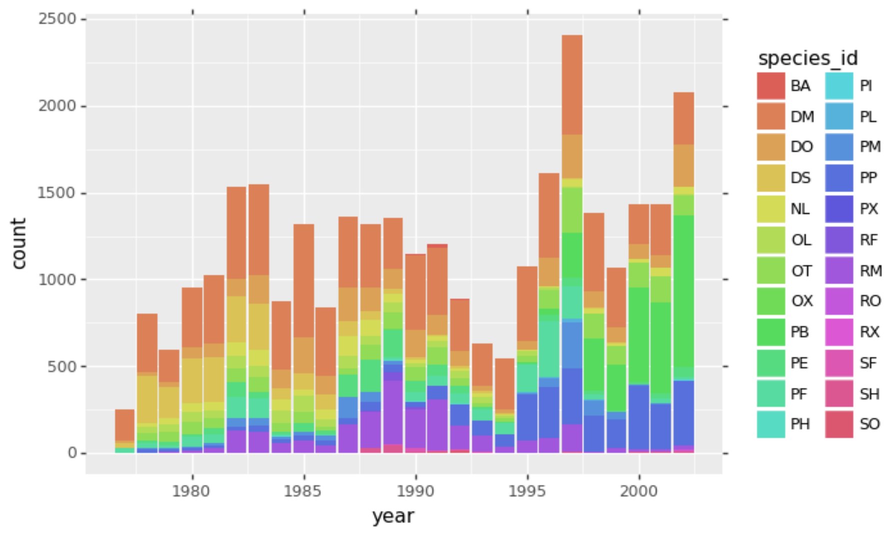


Fig 3.6.1.1 Count Plot

CHAPTER 4

CODING

4.1 IMPORT NECESSARY LIBRARIES

The import statement in a programming language, such as Python, is used to bring external libraries or modules into your code. It allows you to access and use the functions and features provided by those libraries.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.metrics.pairwise import cosine_similarity

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.manifold import TSNE

import warnings

warnings.filterwarnings('ignore')
```

4.2 LOAD THE DATASET

Loading a dataset is a fundamental step in data analysis and machine learning. It involves bringing external data into your programming environment, making it accessible for further exploration, analysis, and model training. Loading a dataset refers to the process of reading and importing the dataset from an external source into your programming environment. This source could be a local file on your computer, a URL, a database, or any other storage location.

```
tracks = pd.read_csv('Spotify-2000.csv')

tracks.head()
```

4.2.1 DISPLAY BASIC INFORMATION ABOUT THE DATASET

```
tracks.info()

tracks.shape
```

4.2.2 DISPLAY SUMMARY STATISTICS

```
print(tracks.describe())
```

4.3 DATA PREPROCESSING

4.3.1 CHECK FOR MISSING VALUES

Checking for missing values in a dataset is an essential step in data preprocessing. Missing values can affect the accuracy of your analysis or machine learning models, and dealing with them appropriately is crucial**.**

```
tracks.isnull().sum()
```

4.3.2 REMOVE MISSING VALUES (IF ANY)

```
tracks_cleaned=tracks.dropna()
```

4.3.3 REMOVE COLUMN’S THAT WON’T BE USED

```
tracks = tracks.drop(['Index'], axis = 1)
```

4.3.4 CHECKING FOR DUPLICATE VALUES

Checking for duplicate values typically refers to examining a dataset to identify and handle instances where the same data points or examples appear more than once. Duplicate values can introduce bias, affect model performance, and lead to incorrect evaluations.

```
duplicates = tracks[tracks.duplicated(subset='Title', keep=False)]
print(duplicates[['Title']])
```

4.3.5 REMOVING DUPLICATE VALUES

```
tracks = tracks.sort_values(by=['Popularity'], ascending=False)
tracks.drop_duplicates(subset=['Title'], keep='first', inplace=True)
```

4.3.6 FEATURE ENGINEERING

Feature engineering is a crucial aspect of the machine learning pipeline that involves creating new features from existing data or transforming existing features to improve the performance of a machine learning model. Effective feature engineering can significantly impact the model's ability to learn patterns and make accurate predictions.

```
def generate_ratings(Popularity):
return np.round((Popularity / max(tracks['Popularity'])) * 5, 2)
tracks['rating'] = tracks['Popularity'].apply(generate_ratings)
print(tracks[['Title', 'Artist', 'Popularity', 'rating']])
```

4.3.7 SPLITTING DATA INTO TRAINING AND TESTING SETS

Splitting a dataset into training and testing sets is a fundamental step in machine learning to evaluate the performance of a model on unseen data. The purpose is to train the model on one subset of the data and then assess its performance on another, independent subset. This process helps ensure that the model generalizes well to new, unseen examples.

```
from sklearn.model_selection import train_test_split
tracks.dropna(inplace=True)
tracks.drop_duplicates(inplace=True)
train_data, test_data = train_test_split(tracks, test_size=0.2, random_state=42)
```

4.3.8 TEXT VECTORIZATION

Text vectorization is the process of converting text data into a numerical format that can be used by machine learning algorithms. In natural language processing (NLP) and text-based machine learning tasks, it's essential to represent text in a way that algorithms can understand. There are several techniques for text vectorization, each with its strengths and use cases.

```
%%capture
song_vectorizer = CountVectorizer()
song_vectorizer.fit(tracks['Top Genre'])
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(train_data['Top Genre'])
X_test = vectorizer.transform(test_data['Top Genre'])
tracks = tracks.sort_values(by=['Popularity'], ascending=False)
```

4.4 PLOTTING

4.4.1 PLOTTING t-SNE VISUALIZATION

t-SNE (t-Distributed Stochastic Neighbor Embedding) visualization is a technique used in machine learning and data analysis to reduce the dimensionality of high-dimensional data while preserving its local structure. The method is particularly useful for visualizing complex datasets in two or three dimensions, making it easier to interpret patterns and relationships.

```
numerical_data = tracks.select_dtypes(include=['number'])
model = TSNE(n_components = 2, random_state = 0)
tsne_data = model.fit_transform(numerical_data.head(500))
plt.figure(figsize = (7, 7))
plt.scatter(tsne_data[:,0], tsne_data[:,1])
plt.show()
```

4.4.2 VISUALIZING NUMBER OF SONGS RELEASED EACH YEAR

A count plot to visualize the distribution of song releases over different years. Each bar represents a year, and the height of the bar corresponds to the number of songs released in that year. This type of visualization provides a quick overview of trends and patterns in the data related to song releases.

```
songs_per_year = tracks['Year'].value_counts().sort_index()
plt.figure(figsize=(12, 6))
sb.countplot(x='Year', data=tracks, palette='viridis')
plt.title('Count of Songs Released Each Year')
plt.xlabel('Year')
plt.ylabel('Number of Songs')
plt.xticks(rotation=45)
plt.show()
```

4.4.3 PLOTTING NUMERICAL COLUMN DISTRIBUTIONS

A histogram to visualize the distribution of values in a numerical column. The x-axis represents the numerical values, and the y-axis represents the frequency or count of observations in each bin. Histograms are useful for understanding the central tendency, spread, and shape of a numerical distribution. Adjusting parameters like bin size and adding customization allows for a more tailored and informative visualization**.**

```
numerical_columns = tracks.select_dtypes(include=['number']).columns
for column in numerical_columns:
   plt.figure(figsize=(8, 5))
   sb.histplot(data=tracks, x=column, bins=20, kde=True)
   plt.title(f'Histogram of {column}')
   plt.xlabel(column)
   plt.ylabel('Frequency')
   plt.show()
```

4.5 BUILDING A RECOMMENDATION SYSTEM

4.5.1 SONG SIMILARITY CALCULATION

Song similarity calculation involves quantifying the degree of similarity or closeness between two or more songs. The goal is to determine how alike songs are in terms of musical content, lyrics, or other relevant features.

```
def get_similarities(song_name, data):
  text_array1 =
song_vectorizer.transform(data[data['Title']==song_name]['Top
Genre']).toarray()
  num_array1 =
data[data['Title']==song_name].select_dtypes(include=np.number).to_nump
y()
  sim = []
  for idx, row in data.iterrows():
    name = row['Title']
    text_array2 = song_vectorizer.transform(data[data['Title']==name]['Top
Genre']).toarray()
    num_array2 =
data[data['Title']==name].select_dtypes(include=np.number).to_numpy()
    text_sim = cosine_similarity(text_array1, text_array2)[0][0]
    num_sim = cosine_similarity(num_array1, num_array2)[0][0]
    sim.append(text_sim + num_sim)
  return sim
```

4.5.2 SONG RECOMMENDATION FUNCTION

A song similarity function is a mathematical or computational method that quantifies the similarity between two or more songs. This function takes input, typically in the form of features or representations of songs, and produces a numerical measure indicating how similar or dissimilar the songs are. The goal is to capture certain aspects of the songs, such as musical content, lyrics, or other relevant features, and provide a basis for comparison.

```
def recommend_songs(song_name, data=tracks):
    if tracks[tracks['Title'] == song_name].shape[0] == 0:
      print('This song is either not so popular or you\
      have entered invalid_name.\n Some songs you may like:\n')
      for song in data.sample(n=5)['Title'].values:
          print(song)
      return
    data['similarity_factor'] = get_similarities(song_name, data)
    data.sort_values(by=['similarity_factor', 'Popularity'],
          ascending = [False, False],
          inplace=True)
```

4.6 WORKING OF RECOMMENDATION SYSTEM

4.6.1 RECOMMENDATIONS FOR A GIVEN INPUT

Recommendations for a given input-output scenario involve suggesting items or actions based on a model's understanding of user preferences, historical behavior, or other relevant information.

```
recommend_songs('Music')
```

```
recommend_songs('The Pretender')
```

```
recommend_songs('rose')
```

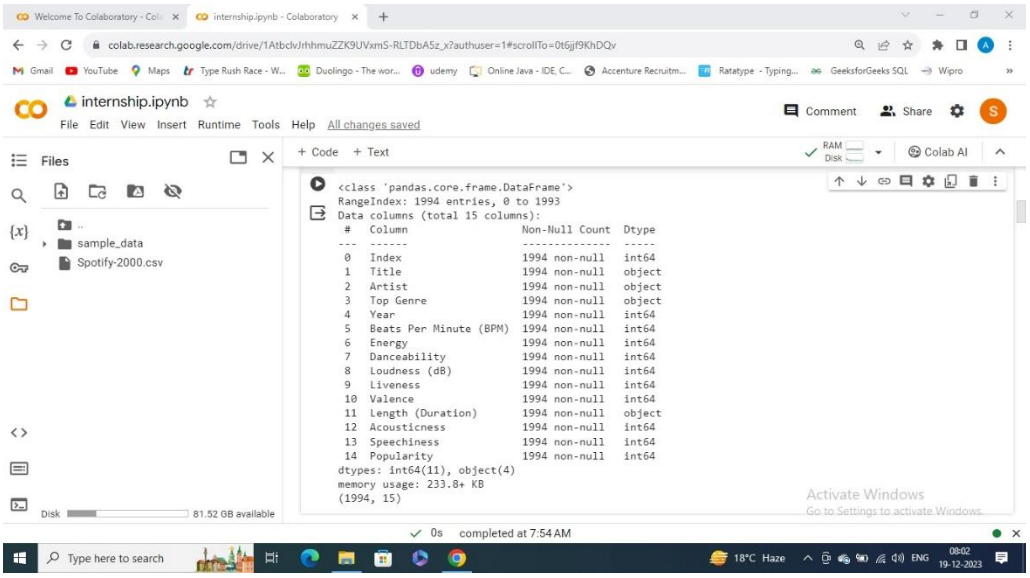
4.7 EVALUATE THE MODEL

Evaluating a machine learning model is a critical step to assess its performance and generalization ability. The evaluation process involves comparing the predictions made by the model against the actual outcomes, using various metrics and techniques.

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn.metrics import precision_score

model = LinearRegression()

model.fit(X_train, train_data['rating'])

predicted_ratings = model.predict(X_test)

true_ratings = test_data['rating']

mae = mean_absolute_error(true_ratings, predicted_ratings)

mse = mean_squared_error(true_ratings, predicted_ratings)

rmse = np.sqrt(mse)

rating_threshold =1.37

ground_truth_labels = (true_ratings >= rating_threshold).astype(int)

binary_predictions = (predicted_ratings >= rating_threshold).astype(int)

precision = precision_score(ground_truth_labels, binary_predictions)


print(f'Mean Absolute Error (MAE): {mae}')

print(f'Mean Squared Error (MSE): {mse}')

print(f'Root Mean Squared Error (RMSE): {rmse}')

print(f'Precision: {precision}')
```

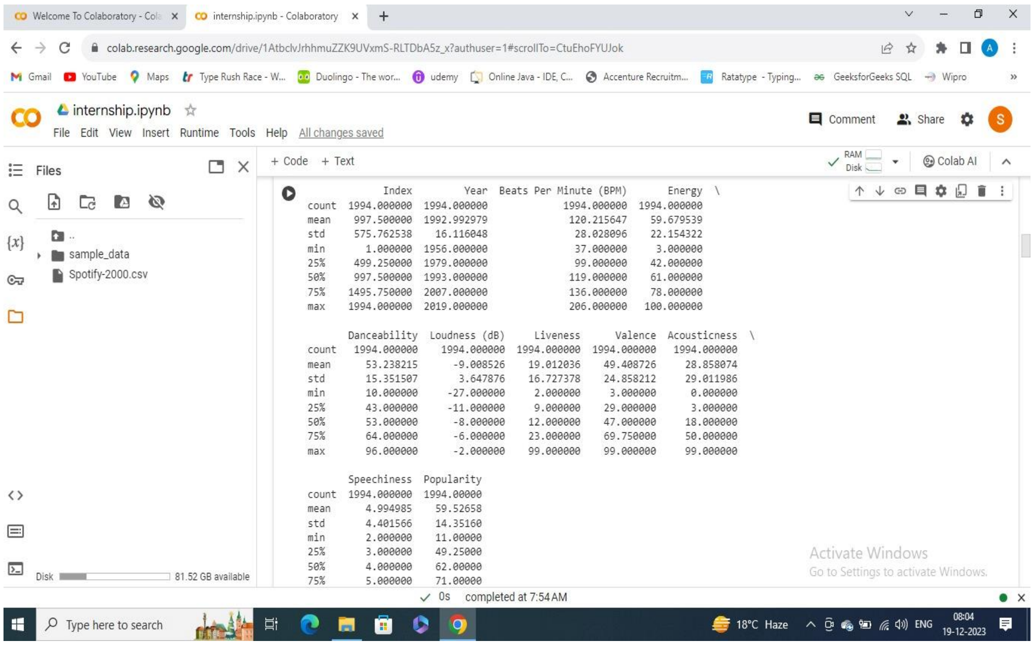
CHAPTER 5

SCREENSHOTS AND OUTPUT



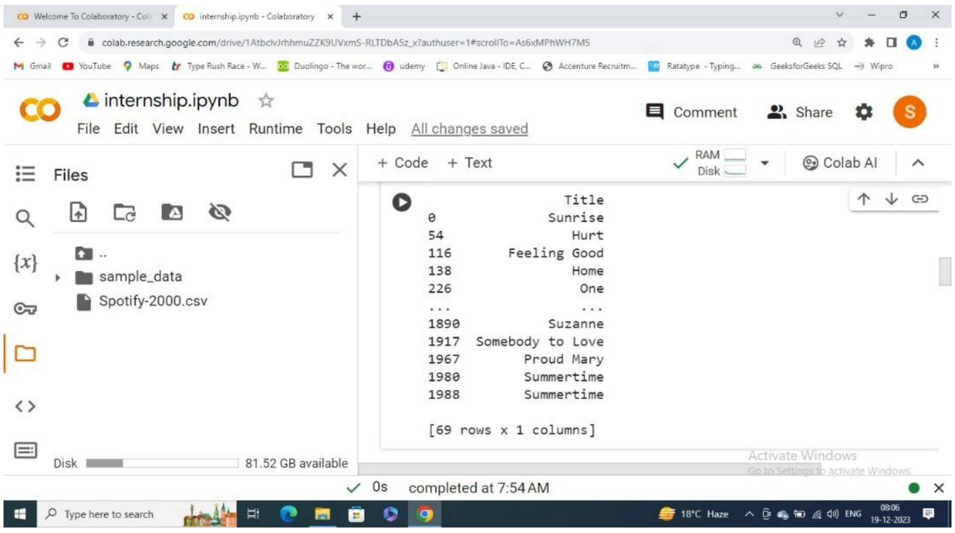
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1994 entries, 0 to 1993
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Index                   1994 non-null   int64 
 1   Title                   1994 non-null   object
 2   Artist                  1994 non-null   object
 3   Top Genre               1994 non-null   object
 4   Year                    1994 non-null   int64 
 5   Beats Per Minute (BPM)  1994 non-null   int64 
 6   Energy                  1994 non-null   int64 
 7   Danceability            1994 non-null   int64 
 8   Loudness (dB)           1994 non-null   int64 
 9   Liveness                1994 non-null   int64 
 10  Valence                 1994 non-null   int64 
 11  Length (Duration)       1994 non-null   object
 12  Acousticness            1994 non-null   int64 
 13  Speechiness             1994 non-null   int64 
 14  Popularity              1994 non-null   int64 
dtypes: int64(11), object(4)
memory usage: 233.8+ KB
(1994, 15)
```

Fig 5.1 Basic Information of Dataset



```
             Index         Year  Beats Per Minute (BPM)       Energy  \
count  1994.000000  1994.000000             1994.000000  1994.000000   
mean    997.500000  1992.992979              120.215647    59.679539   
std     575.762538    16.116048               28.028096    22.154322   
min       1.000000  1956.000000               37.000000     3.000000   
25%     499.250000  1979.000000               99.000000    42.000000   
50%     997.500000  1993.000000              119.000000    61.000000   
75%    1495.750000  2007.000000              136.000000    78.000000   
max    1994.000000  2019.000000              206.000000   100.000000   

       Danceability  Loudness (dB)     Liveness      Valence  Acousticness  \
count   1994.000000    1994.000000  1994.000000  1994.000000   1994.000000   
mean      53.238215      -9.008526    19.012036    49.408726     28.858074   
std       15.351507       3.647876    16.727378    24.858212     29.011986   
min       10.000000     -27.000000     2.000000     3.000000      0.000000   
25%       43.000000     -11.000000     9.000000    29.000000      3.000000   
50%       53.000000      -8.000000    12.000000    47.000000     18.000000   
75%       64.000000      -6.000000    23.000000    69.750000     50.000000   
max       96.000000      -2.000000    99.000000    99.000000     99.000000   

       Speechiness  Popularity  
count  1994.000000  1994.00000  
mean      4.994985    59.52658  
std       4.401566    14.35160  
min       2.000000    11.00000  
25%       3.000000    49.25000  
50%       4.000000    62.00000  
75%       5.000000    71.00000  
```

Fig 5.2 Statistical Summary



```
                 Title
0              Sunrise
54                Hurt
116       Feeling Good
138               Home
226                One
...                ...
1890           Suzanne
1917  Somebody to Love
1967        Proud Mary
1980        Summertime
1988        Summertime

[69 rows x 1 columns]
```

Fig 5.3 Duplicate Data

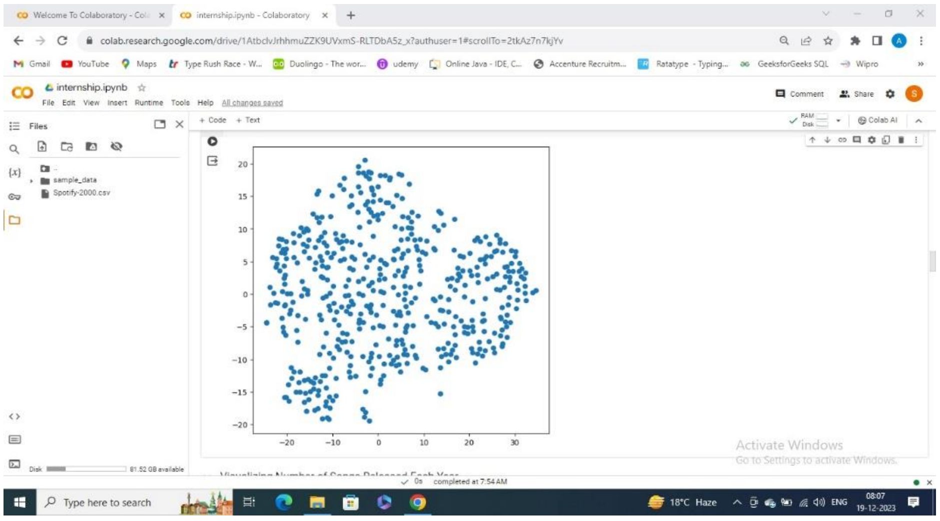


Fig 5.4 t-SNE Visualization

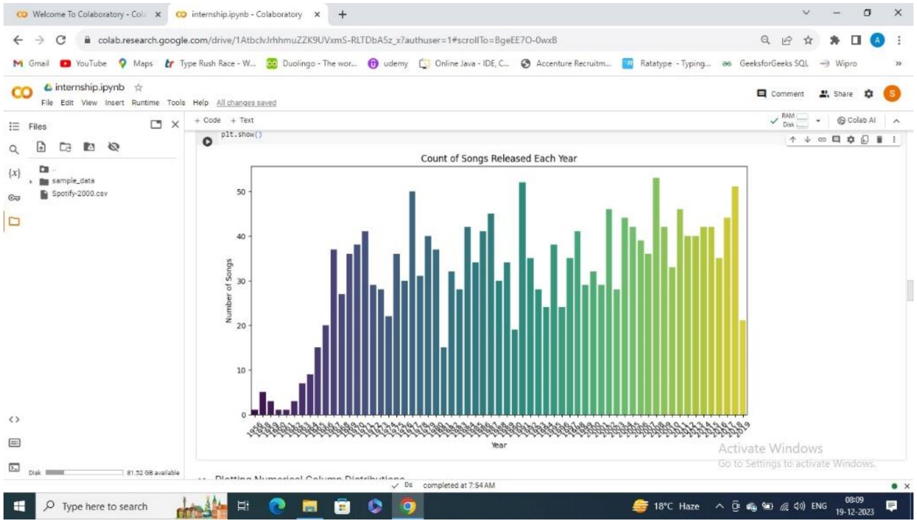


Fig 5.5 Count Plot for Songs

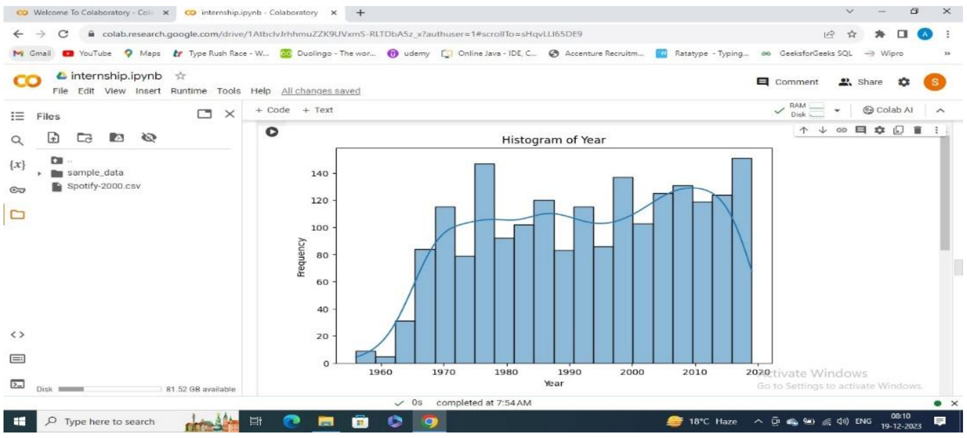


Fig 5.6 Histogram Of Year

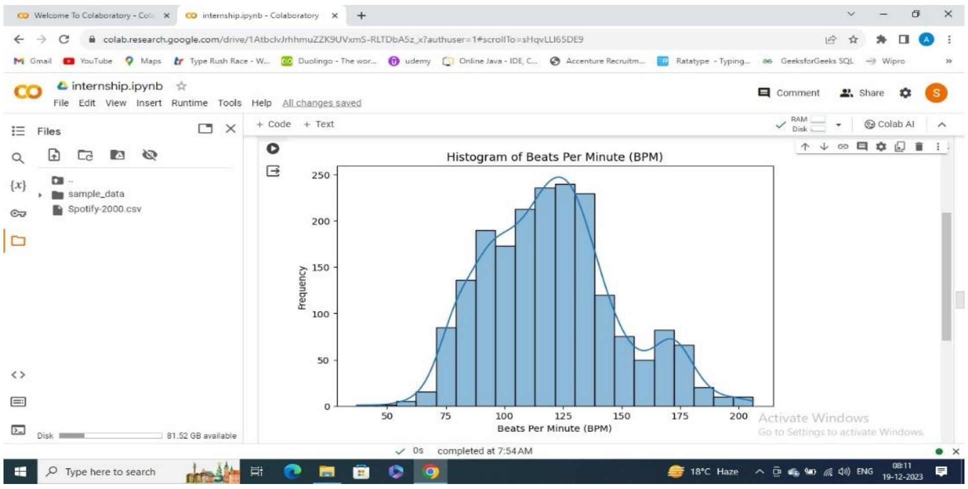


Fig 5.7 Histogram Of Beats per Minute

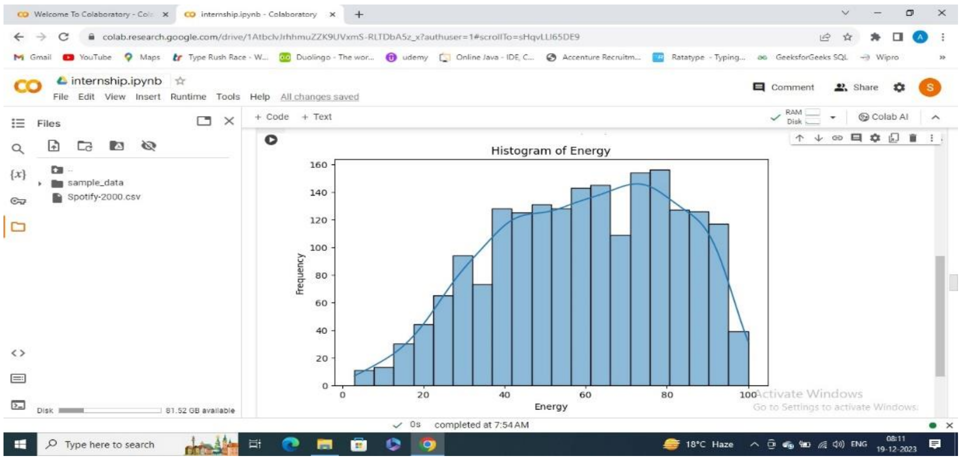


Fig 5.8 Histogram of Energy

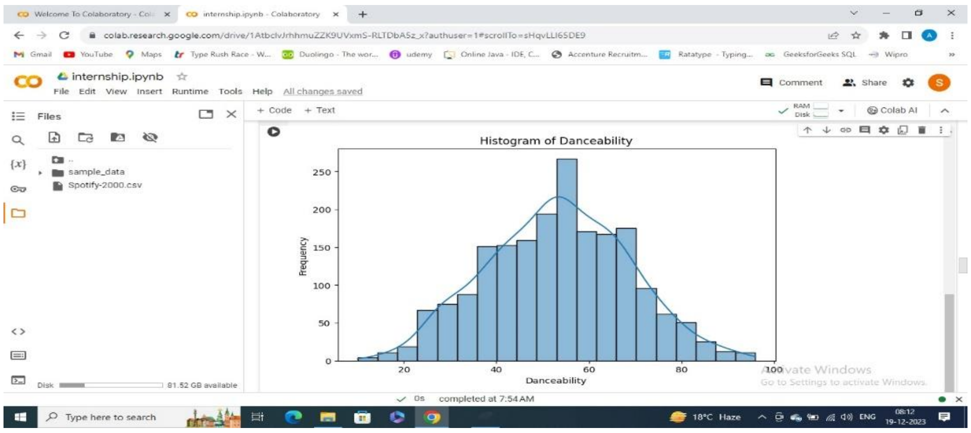


Fig 5.9 Histogram of Danceability

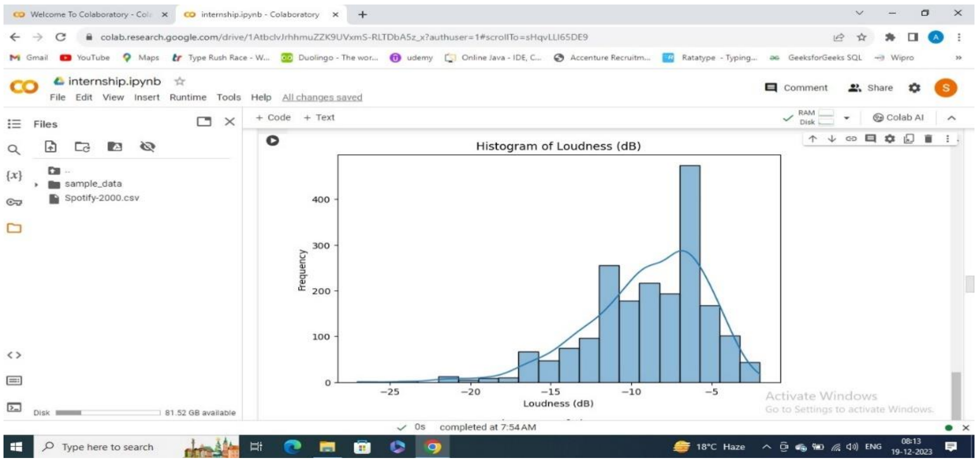


Fig 5.10 Histogram of Loudness

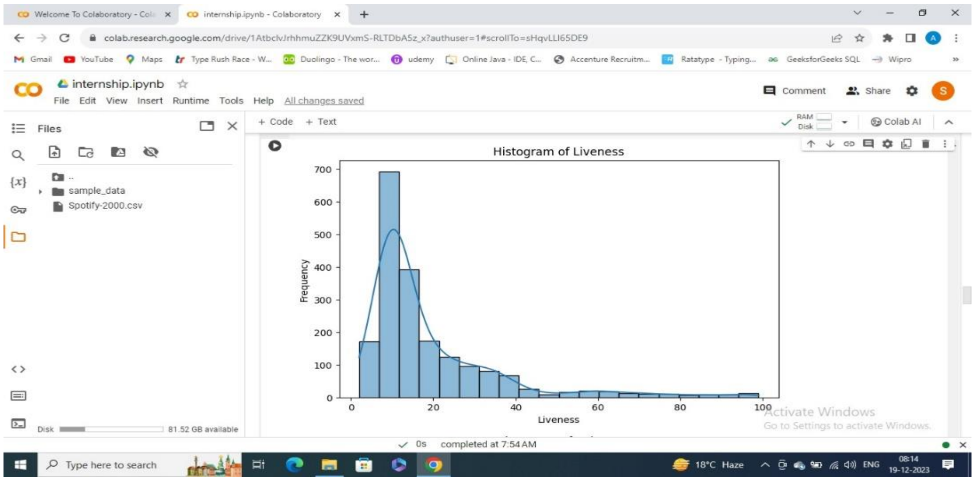


Fig 5.11 Histogram of Liveness

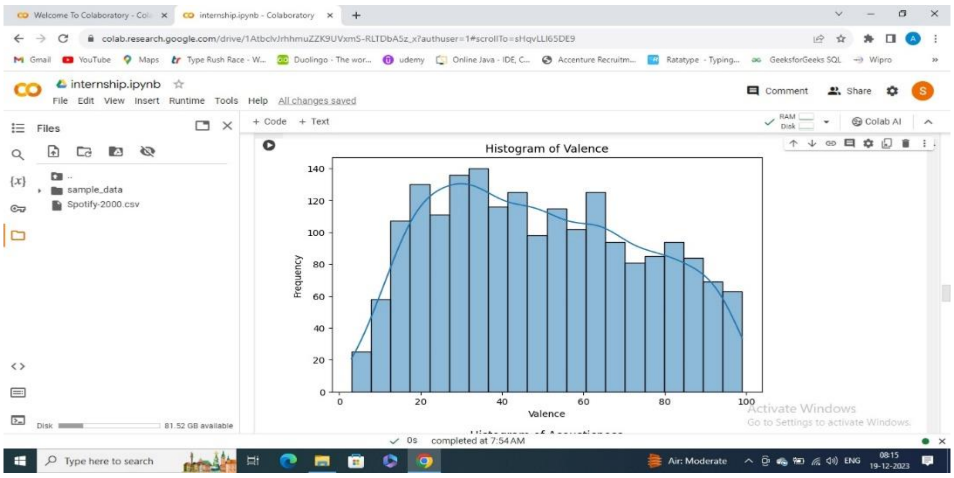


Fig 5.12 Histogram of Valence

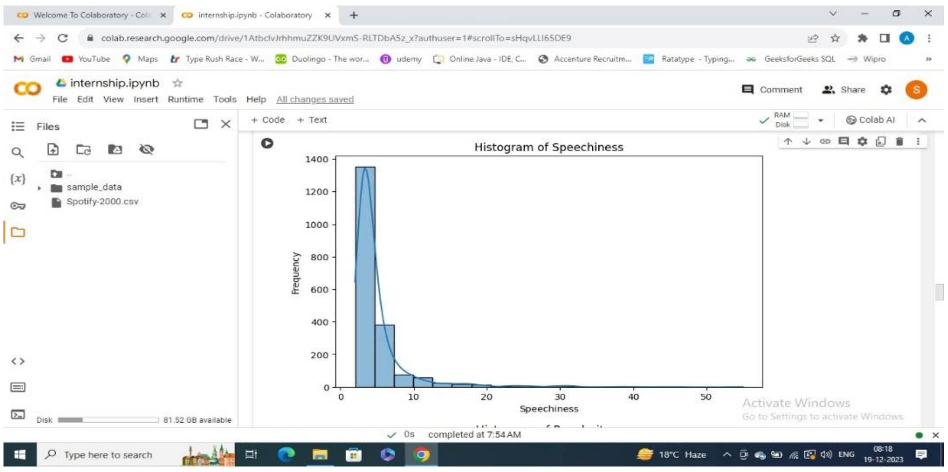


Fig 5.13 Histogram of Speechiness

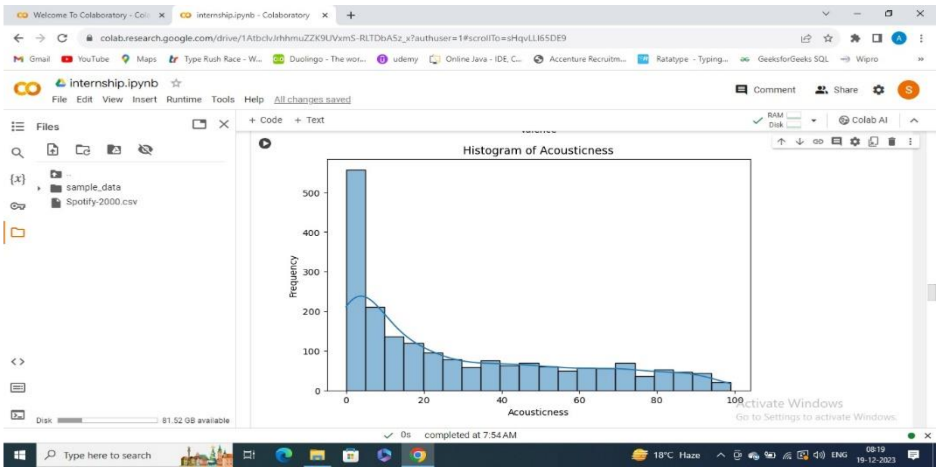


Fig 5.14 Histogram of Acousticness

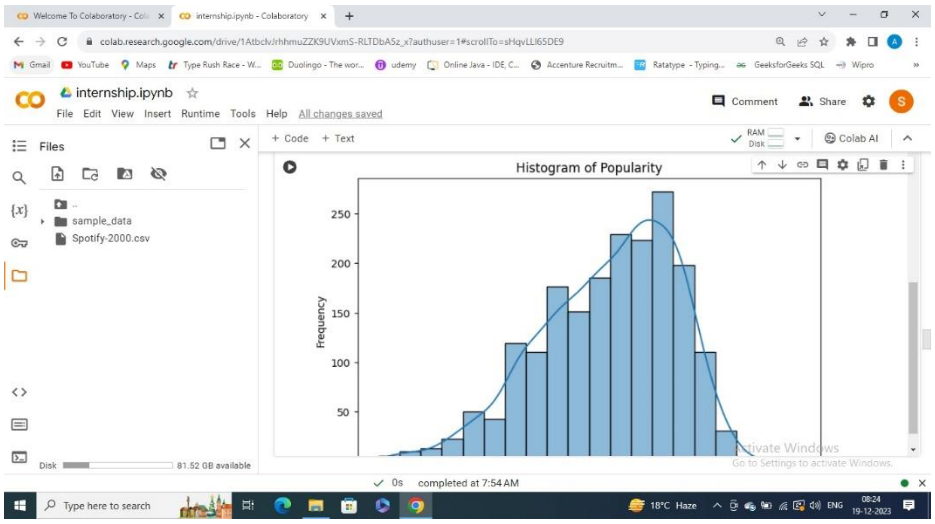


Fig 5.15 Histogram of Popularity

CHAPTER 6
CONCLUSION

In conclusion, the implementation of a music recommendation system using machine learning has shown promising results and signifies a significant step towards achieving personalized and accurate music recommendations. The evaluation metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Precision, provide valuable insights into the system's performance.

THE MEAN ABSOLUTE ERROR(MAE), MEAN SQUARED ERROR(MSE), ROOT MEAN SQUARED ERROR(RMSE) VALUES SHOW THAT:

Mean Absolute Error (MAE): 0.4795737750447214

Mean Squared Error (MSE): 0.3618399169673623

Root Mean Squared Error (RMSE): 0.6015313100474174

The low values of MAE, MSE, and RMSE indicate the effectiveness of the recommendation system in accurately predicting user ratings.

THE PRECISION SCORE SHOW THAT:

Precision: 0.9770408163265306

The precision score of 0.977 reflects a high level of accuracy in recommending songs that align with user preferences.

CHAPTER 7
BIBLIOGRAPHY

[1] Tewari, A.S. Kumar, and Barman, A.G, “Book recommendation system based on combining features of content-based filtering, collaborative filtering and association rule mining,” International Advance Computing Conference (IACC), IEEE, pp 500 – 503, April 2014.

[2] Fang, J., Grunberg, D., Luit, S., & Wang, Y. (2017, December). Development of a music recommendation system for motivating exercise. In Orange Technologies (ICOT), 2017 International Conference on (pp. 83-86). IEEE.

[3] Luo Zhenghua, “Realization of Individualized Recommendation System on Books Sale,” IEEE 2012 International Conference on Management of e-Commerce and e-Government.

[4] C. L. Liu and Y. C. Chen, “Background music recommendation based on latent factors and moods,” Knowledge-Based Systems, vol. 159, no. NOV.1, pp. 158–170, 2018.

[5] H. Pan and Z. Zhang, “Research on context-awareness mobile tourism e-commerce personalized recommendation model,” Journal of Signal Processing Systems, vol. 93, no. 2-3, pp. 147–154, 2021.

[6] Millecamp, Martijn, et al. ”Controlling Spotify recommendations: effects of personal characteristics on music recommender user Interfaces.” Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization. ACM, 2018.

[7] Schedl, Markus, Arthur Flexer, and Julin Urbano. ”The neglected user in music information retrieval research.” Journal of Intelligent Information Systems 41.3 (2013): 523-539.

[8] Knees, Peter, and Markus, S., ”A survey of music similarity and recommendation from music context data.” ACM Transactions on Multimedia Computing, Communications, and Applications 10,2.

[9] Ramasuri, A., Ajay, K., Bhoomireddy, P., Athukumsetti, J., Achanta, S., Mudunuru, V., (2021, July), Music Recommendation System With Advanced Classification, IJERT Volume 10, Issue 7.

[10]]O Bryant, Jacob. ”A survey of music recommendation and possible improvements.” (2017).

[11] Pengfei Sun (2022). Music Individualization Recommendation System Based on Big Data Analysis, Comput Intell Neurosci, Hindawi Limited.

[12] A. Schindler and A. Rauber, “Harnessing music-related visual stereotypes for music information retrieval,” ACM Transactions on Intelligent Systems& Technology, vol. 8, no. 2, pp. 1–21, 2016.

[13] R. Wang, X. Ma, C. Jiang, Y. Ye, and Y. Zhang, “Heterogeneous information network-based music recommendation system in mobile networks,” Computer Communications, vol. 150, pp. 429–437, 2020.