MIT | Academy of Engineering (An Autonomous Institute Affiliated to Savitribai Phule Pune University)	**PROJECT REPORT**	
	ACADEMIC YEAR	2022-2023
Alandi (D), Pune – 412105	**SEMESTER**	VI
SCHOOL OF ELECTRICAL ENGG.	**CLASS & DIVISION**	TYBTECH

COURSE CODE	ET363
COURSE	Soft Computing

Name: Shruti Somankar

Roll No.: 158

Seat no.: T227056

PRN: 0120200288

Name: Aniket Kamble

Roll No.: 137

Seat No.: T227012

PRN: 0120200044

ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the kind of help and guidance received from several people in the preparation of this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation, and guidance. First and foremost, I wish to record my sincere gratitude to the management of this college and to our dean for her constant support and encouragement in providing all the necessary components, for the availability of library and laboratory facilities needed to prepare the entire project and this report.

I express my sincere gratitude to Guide, Mrs. Smita Kulkarni, for guiding us in investigations of the project and in carrying out experimental work and for encouragement and inspiration throughout the project work. Without her valuable guidance, this work would never have been successful. Our numerous discussions were extremely helpful. We respect her esteem for the guidance, encouragement, and inspiration received from her.

Introduction

Nowadays, a recommender system can be found on almost every information-intensive website. For example, a list of likely preferred products is recommended to customers when browsing the target product on Amazon.

Moreover, when watching a video clip on YouTube, a recommender system employed in the system suggests some relevant videos to users by learning the users' behaviors that were generated previously. So to speak, recommender systems have deeply changed how we obtain information. Recommender systems make it easier and more convenient for people to receive information and provide great potential for economic growth. As more and more people realize the importance and power of recommender systems, the exploration of designing high-quality recommender systems has remained an active topic in the community over the past decade. Due to the continuous efforts in the field, many recommender systems have been developed and used in various domains. Based on this, a key question is how to know the performance of recommender systems so that the most suitable ones can be found to apply in certain contexts or domains.

The answer goes to evaluating recommender systems by conducting rigorous and scientific evaluation experiments Evaluating recommender systems has become increasingly important with the growing popularity of recommender systems in some applications. It is often the case that an application designer needs to choose between a set of candidate recommendation algorithms. This goal can be achieved by comparing the performance of these algorithms in evaluation experiments. Besides, evaluating recommender systems can help researchers select, tune and design recommender systems. This is because when designing a recommender system, some key factors influencing the system's quality often appear in the evaluation process. When considering evaluation metrics, evaluators should not only take into account the accuracy metrics, but also some extra quality metrics, or to say beyond accuracy metrics, which attach importance to the fact that users are often not interested in the items that they already know and surely like but sometimes in discovering new items and exploring diverse items.

Dataset Description

The Spotify dataset is a collection of information about songs, albums, and artists available on the Spotify music streaming platform. It contains data on millions of tracks and thousands of artists and can be used for a wide range of analysis and research purposes.

The dataset contains both structured and unstructured data, including:

- Metadata: This includes the artist's name, album name, release date, genre, popularity score, and track length.
- Audio features: These are features that describe the characteristics of the audio signal, such as loudness, tempo, key, and mode.
- User interactions: This includes data on how users interact with the platform, such as the number of times a song has been played, skipped, or added to a playlist.
- Textual data includes user-generated content such as song titles, artist names, and user comments.

The dataset is available in several formats, including CSV, JSON, and SQL. It can be accessed through the Spotify Web API (used in our program to access the platform's data)

The dataset is accessed by performing the following steps:

1. Register for a Spotify Developer account at https://developer.spotify.com/. Once registered, we can access the Spotify Web API and obtain an API key.
2. Use the API key to make requests to the Spotify API. Several libraries are available in various programming languages to interact with the API, including the Spotipy library for Python, the Spotify-web-api-js library for JavaScript, and the Spotify-web-api-php library for PHP.
3. Once we have authenticated our API key, we can start making requests to the Spotify API. For example, we can use the API to search for tracks, albums, or artists, retrieve information about individual tracks or albums, and retrieve audio features for individual tracks.
4. To request the Spotify API, we must specify the endpoint we want to access and any parameters or filters we want to apply. For example, if we want to retrieve information about a specific track, we can use the /tracks/{id} endpoint and specify the ID of the track we want to retrieve.
5. Once we have requested the Spotify API, we will receive a response in JSON format containing the requested data. We can then parse this data and use it for analysis or research purposes.

The dataset is frequently used for music recommendation, genre classification, and artist similarity analysis. It is also commonly used in machine learning research, particularly in natural language processing and audio analysis.

The Spotify dataset contains a wide range of features and columns. Here is an overview of some of the most important ones:

- id: A unique identifier for each track, album, or artist in the dataset.
- name: The name of the track, album, or artist.
- artist: The name of the artist associated with the track or album.
- album: The name of the album associated with the track.
- release_date: The date on which the track or album was released.
- popularity: A score between 0 and 100 that indicates the popularity of a track or artist on Spotify.
- duration_ms: The length of the track in milliseconds.
- explicit: A boolean value that indicates whether the track contains explicit lyrics.
- danceability: A score between 0 and 1 indicates a track's suitability for dancing.
- energy: A score between 0 and 1 indicates a track's energy level.
- key: The musical key in which the track is written.
- mode: A binary value that indicates whether the track is written in a major or minor key.
- speechiness: A score between 0 and 1 indicates the presence of spoken words in a track.
- acousticness: A score between 0 and 1 indicates the level of acoustic instrumentation in a track.
- instrumentalness: A score between 0 and 1 indicates a track's level of instrumental content.
- liveness: A score between 0 and 1 indicates the likelihood that a track was recorded live.
- valence: A score between 0 and 1 indicates a track's positivity or happiness.
- tempo: The tempo of the track in beats per minute.
- time_signature: The time signature of the track.

In addition to these features, many other columns and features provide additional information about the tracks, albums, and artists in the dataset. For example, some columns provide information about each track or artist's genre, label, and country of origin.

Problem Statement

“Build a Recommendation System using Spotify API to recommend a user a song based on the song the user likes.”

Tools & Libraries

Spotipy Library: A Python library provides a convenient interface for accessing the Spotify Web API. It simplifies interacting with the API and extracting data from the Spotify dataset using Python. It offers a range of methods for searching and retrieving data, handling authentication and authorization, and managing access credentials. Spotipy is widely used in data science and machine learning projects involving music data.

Sklearn: Sklearn for building KNN or Random Forest classifiers, we can import the relevant classes from the sklearn.neighbors and sklearn.ensemble modules, respectively. We can then train the classifier on the training data using the fit method and make predictions on new data using the prediction method. To evaluate the model's performance, we can use a range of metrics provided by sklearn.metrics module, including accuracy, precision, recall, F1 score, and more. These metrics can be calculated using relevant functions from the module.

Pandas and Numpy: To perform numeric calculations on the data frame(dataset) and visualize the dataset

Data Description

- id: A unique identifier for each track, album, or artist in the dataset.
- name: The name of the track, album, or artist.
- artist: The artist's name associated with the track or album.
- album: The name of the album associated with the track.
- release_date: The date on which the track or album was released.
- popularity: A score between 0 and 100 that indicates the popularity of a track or artist on Spotify.
- duration_ms: The length of the track in milliseconds.
- explicit: A boolean value that indicates whether the track contains explicit lyrics.
- danceability: A score between 0 and 1 indicates a track's suitability for dancing.
- energy: A score between 0 and 1 indicates a track's energy level.
- key: The musical key in which the track is written.
- mode: A binary value that indicates whether the track is written in a major or minor key.
- speechiness: A score between 0 and 1 indicates the presence of spoken words in a track.
- acousticness: A score between 0 and 1 indicates the level of acoustic instrumentation in a track.
- instrumentalness: A score between 0 and 1 indicates a track's level of instrumental content.
- liveness: A score between 0 and 1 indicates a track's likelihood of being recorded live.
- valence: A score between 0 and 1 indicates a track's positivity or happiness.
- tempo: The tempo of the track in beats per minute.
- time_signature: The time signature of the track.

index	Apop	Popularity	danceability	energy
count	5564.0	5564.0	5564.0	5564.0
mean	73.62005751258087	72.41570812365205	0.6507018332135155	0.6349772106398275
std	11.516495149982232	17.557882841759504	0.14335414034004632	0.16917214850985818
min	19.0	0.0	0.245	0.0191
25%	66.0	65.0	0.561	0.524
50%	74.0	77.0	0.662	0.651
75%	83.0	83.0	0.753	0.766
max	100.0	100.0	0.978	0.974

Data Cleaning

As the dataset was directly extracted from Spotify using Spotify Web API there were no null values to be seen apart from the genre column which contained 6% NULL data which were dropped.

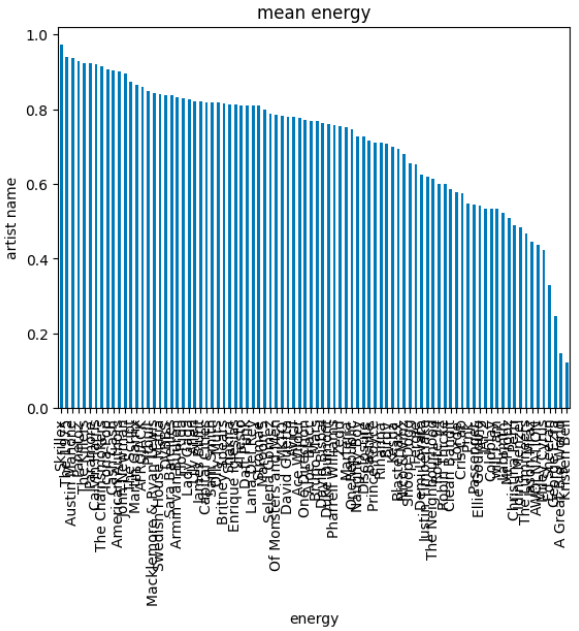
index	Total	Percent
Genres	389	6.99137311286844
Tname	0	0.0
instrumentalness	0	0.0
duration_ms	0	0.0
analysis_url	0	0.0
track_href	0	0.0
uri	0	0.0
id	0	0.0
type	0	0.0
tempo	0	0.0
valence	0	0.0
liveness	0	0.0
acousticness	0	0.0
TAlbum	0	0.0
speechiness	0	0.0
mode	0	0.0
loudness	0	0.0
key	0	0.0
energy	0	0.0
danceability	0	0.0
Popularity	0	0.0
Apop	0	0.0
Aname	0	0.0
time_signature	0	0.0

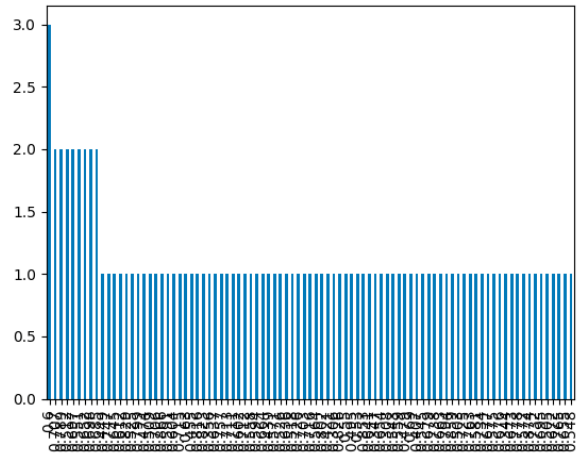
EDA

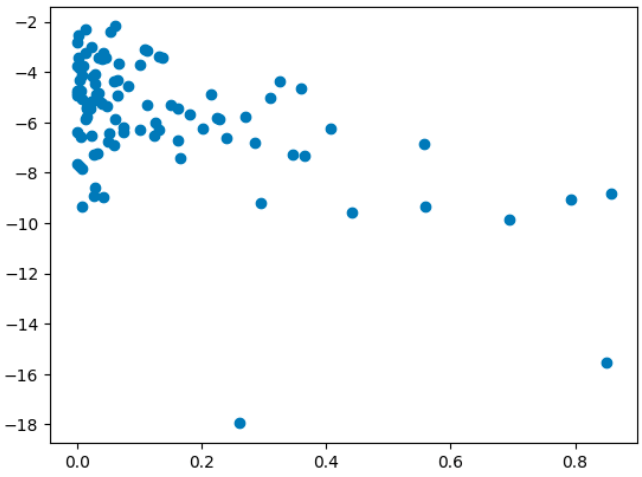
By the graph, there seems to be a positive correlation between danceability and energy, which implies that more energetic songs tend to be more danceable. Similarly, there seems to be a negative correlation between loudness and acousticness, indicating that louder songs tend to have lower acousticness values. These relationships are not surprising since danceability and energy are related to a song's tempo, rhythm, and overall feel. In contrast, loudness and acousticness are related to the production and instrumentation of a song. However, it is important to note that correlation does not necessarily imply causation, and other factors could

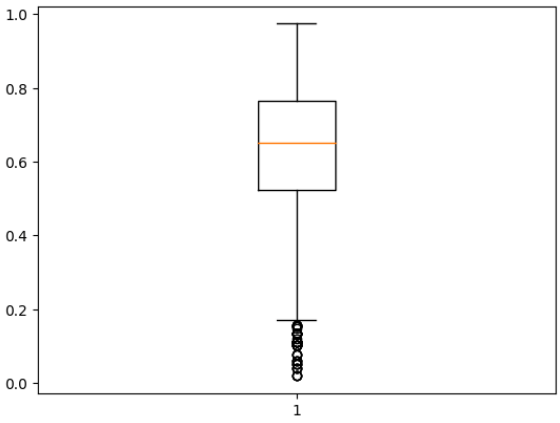
also contribute to these relationships. Therefore, it is important to consider these relationships within the context of the broader musical and cultural landscape.

There seems to be a positive correlation between loudness and energy. This implies that louder songs tend to be more energetic, which makes sense since loudness is a measure of the overall volume of a song. Energy is related to the intensity and excitement of the music. However, it is important to note that correlation does not necessarily imply causation and other factors could contribute to this relationship. Additionally, personal preferences and cultural trends can also play a role in determining the popularity and perception of loud and energetic music. Therefore, it is important to consider this relationship within the broader musical and cultural context.

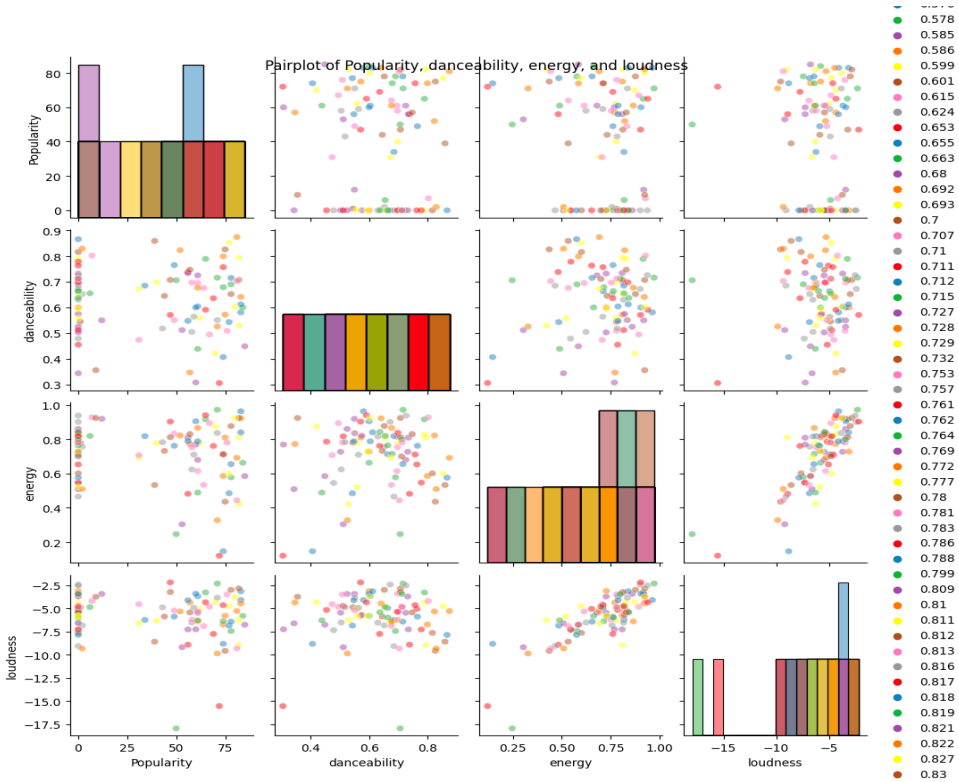


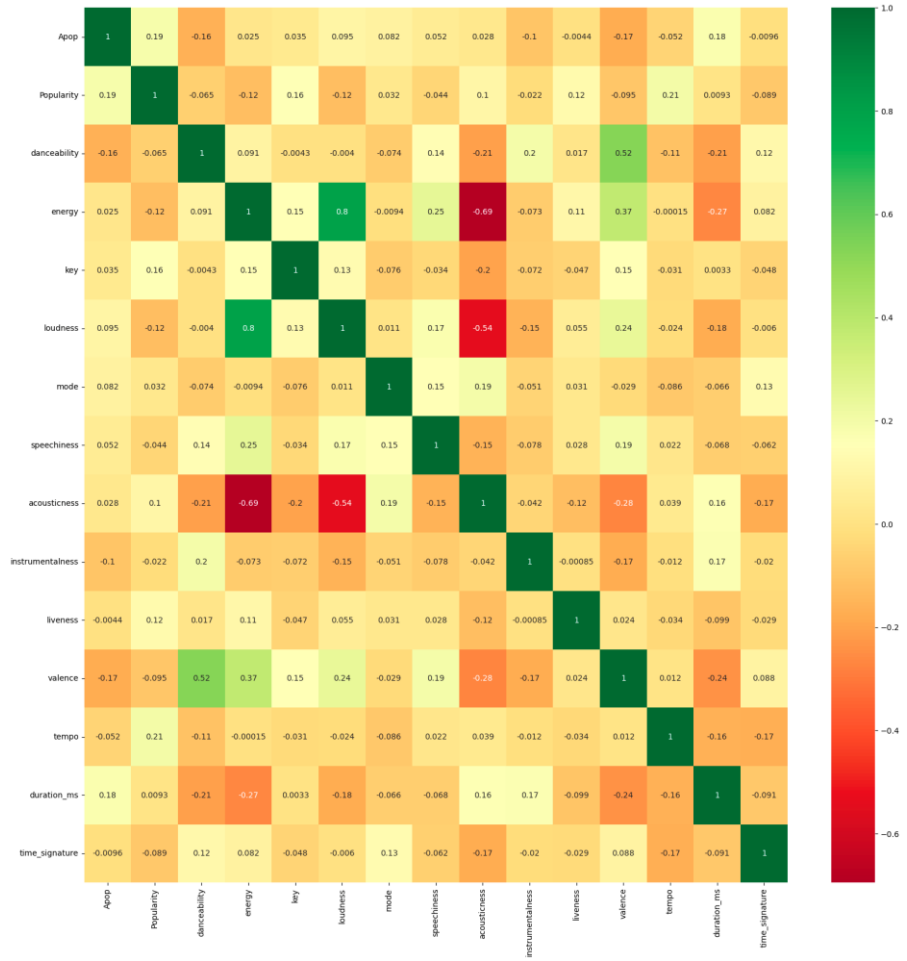






There seems to be a negative correlation between loudness and acousticness. This implies that songs that are louder tend to have lower acousticness values, meaning that they have less emphasis on natural sounds and more emphasis on electronic or amplified sounds. This relationship makes sense since loudness and acousticness are both related to the production and instrumentation of a song. Higher loudness levels usually require more amplification and electronic processing, which can reduce the natural acoustic quality of a recording. In contrast, lower loudness levels may allow for more natural and acoustic elements to be heard in the mix.


Pairplot of Popularity, danceability, energy, and loudness
Popularity
danceability
energy
loudness
0.578
0.585
0.586
0.599
0.601
0.615
0.624
0.653
0.655
0.663
0.68
0.692
0.693
0.7
0.707
0.71
0.711
0.712
0.715
0.727
0.728
0.729
0.732
0.753
0.757
0.761
0.762
0.764
0.769
0.772
0.777
0.78
0.781
0.783
0.786
0.788
0.799
0.809
0.81
0.811
0.812
0.813
0.816
0.817
0.818
0.819
0.821
0.822
0.827
0.83
0.833


Apop
Popularity
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
duration_ms
time_signature

The Spotify dataset contains several variables that can measure different aspects of a song's musical features, such as danceability, energy, loudness, instrumentalness, and others. Analyzing the correlation matrix of all these variables, we can observe several interesting patterns:

- There is a strong positive correlation between loudness and energy, which indicates that louder songs tend to have higher energy levels.
- There is also a strong negative correlation between acousticness and instrumentalness, which indicates that songs with more acoustic instruments tend to have lower levels of electronic or synthesized instrumentation.
- Valence, which measures the degree of positivity conveyed by a musical piece, has a moderate positive correlation with danceability, indicating that more danceable songs tend to have a more positive mood.
- On the other hand, there is a weak negative correlation between valence and energy, indicating that songs with a more positive mood tend to have lower energy levels.
- There is also a weak positive correlation between danceability and energy, indicating that more danceable songs tend to have higher energy levels.

These correlations suggest that different musical features are related in complex and multifaceted ways.

ML Model Implementation

KNN Model

KNN is a non-parametric algorithm for classification and regression problems. It works by finding the K number of closest data points (i.e., songs) to a given song based on their similarity in terms of features like danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. The algorithm calculates the distance between the features of the query song and those of the training songs and selects the K nearest training points as the neighbors.

Once the closest K neighbors have been identified, the algorithm can make recommendations based on the most common labels (i.e., song names) among the neighbors. In other words, the algorithm determines which song names appear most frequently among the K's nearest neighbors, and recommends those songs to the user. The idea behind this approach is that the same listeners will likely enjoy songs with similar features.

To implement KNN in Python, we can use the KNeighborsClassifier class from the sci-kit-learn library. We can first split the dataset into training and testing sets using the train_test_split function. We can then use the training set to train the KNN model by fitting the KNeighborsClassifier object to the features of the training data. Once the model is trained, we can use the prediction method to predict new data points (i.e., songs) and recommend similar songs to the user.

KNN is a simple yet effective algorithm that can recommend songs to users based on their listening history and preferences.

Random Forest

Random forests are an ensemble learning method for classification, regression, and other tasks. The basic idea behind the random forest algorithm is to build many decision trees using bootstrap aggregating (or "bagging") and random feature subsets, and then use the collective predictions of these trees to make a final prediction. To use random forests for song prediction in Spotify, we first split the dataset into training and testing sets using train_test_split from sci-kit-learn. Then we create a RandomForestClassifier object and fit it to the training data. We can specify the number of trees in the forest using the n_estimators parameter, and other hyperparameters, such as the maximum depth of each tree and the minimum number of samples required to split a node.

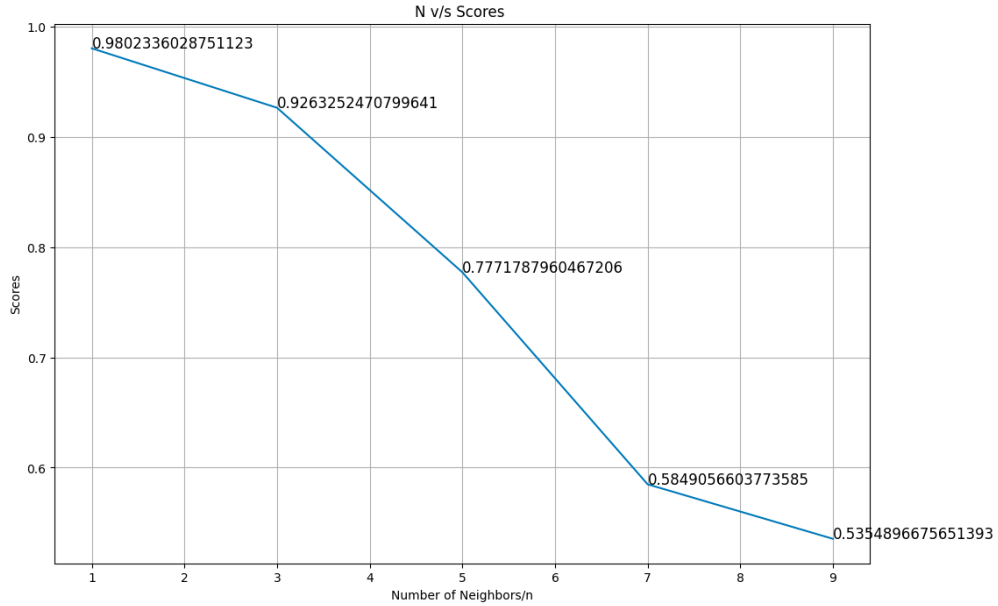
Once the model is trained, we can predict the test data using the prediction method. In addition to accuracy, other metrics such as precision, recall, and F1-score can be used to evaluate the model's performance. Precision measures the proportion of true positives among all predicted positives, while recall measures the proportion of true positives among all actual positives. The F1-score is the harmonic mean of precision and recall and provides a balanced measure of both.

Overall, random forests are a powerful algorithm that can be used for song prediction in Spotify. They are robust to noise and can handle high-dimensional datasets with many features, making them a popular choice for machine learning tasks.

Comparative Analysis of ML Models

Both KNN and Random Forest models perform very well on the Spotify dataset. The Random Forest model has a slightly higher accuracy of 0.989 compared to KNN's accuracy of 0.981. The precision of both models is also very high, with Random Forest having a slightly higher precision of 0.983 compared to KNN's precision of 0.974. The recall score, which measures the proportion of correct positives identified, is also high for both models, with Random Forest having a slightly higher score of 0.989 compared to KNN's score of 0.981. Finally, the F1-score, a harmonic mean of precision and recall, is also very high for both models, with Random Forest having a slightly higher score of 0.986 compared to KNN's score of 0.977. Overall, both models perform very well on the Spotify dataset, but Random Forest has a slight edge in terms of accuracy and precision.

Effect of different K values on the KNN model in terms of accuracy of the model.



KNN:

- Has an accuracy of 0.9811, which indicates that it correctly predicts the genre of songs in the dataset with a high degree of accuracy.
- Has a precision of 0.974, which means that out of all the songs the model predicted to be in a certain genre, 97.4% were actually in that genre.
- Has a recall of 0.9811, meaning that out of all the songs in a certain genre, 98.1% of them were correctly identified by the model.
- Has an F1-score of 0.9767, a weighted average of precision and recall, and indicates a good balance between the two metrics.

Random Forest:

- The accuracy of 0.9892 is slightly higher than the KNN model, indicating that it may be slightly better at predicting the genre of songs in the dataset.
- Has a precision of 0.984, which is slightly higher than the KNN model, meaning that out of all the songs that the model predicted to be in a certain genre, 98.4% of them were actually in that genre.
- Has a recall of 0.9892, which is slightly higher than the KNN model, meaning that out of all the songs in a certain genre, 98.9% of them were correctly identified by the model.
- Has an F1-score of 0.9858, which is slightly higher than the KNN model and indicates a good balance between precision and recall.

The Random Forest model has a slightly better performance than the KNN model. However, the difference in performance between the two models is relatively small, so the choice of which model to use may depend on other factors, such as ease of implementation or computational efficiency.