

Music Genre Classification Using Machine Learning

Through Spotify Data



Team Members:

P. Tarun Reddy (20481A5443)

Y. Sreshta (20481A5464)

Mohammad Saajid (20481A5433)

Ch. Siva Naga Prakash (20481A5409)

INDEX

Pg. No:

1. INTRODUCTION 1
   1. Overview 1
   2. Purpose. 1
2. LITERATURE SURVEY 2
   1. Existing problem 2
   2. Proposed solution 3
3. THEORITICAL ANALYSIS 4
   1. Block diagram 4
   2. Hardware / Software designing 5
4. EXPERIMENTAL INVESTIGATIONS 6
5. FLOWCHART 7
6. RESULT 7
7. ADVANTAGES & DISADVANTAGES 8
8. APPLICATIONS 10
9. CONCLUSION 11
10. FUTURE SCOPE 12
11. BIBILOGRAPHY 13

11.1APPENDIX 13

1. INTRODUCTION:
   1. Overview:

The project aims to develop a machine learning model for classifying song genres based on Spotify data. Spotify is a popular music streaming platform that provides a vast amount of information about songs, including audio features like tempo, energy, danceability, instrumentalness, and more. Leveraging this data, the goal is to build a robust and accurate classification system capable of predicting the genre of a given song.

* 1. Purpose:

Use cases for the effort to categorize music genres using machine learning and Spotify data include the following:

* + - * Music recommendations: Systems for recommending music can include the created model for categorizing music by genre. To improve the user's music discovery experience, the system may examine a song's audio attributes and suggest further songs from the same genre.
      * Playlist generation: The system can automatically construct and curate playlists based on user preferences by precisely predicting the genre of a song. Users may get playlists that are customized for their favorite musical genres, activities, or emotions.
      * Personalized User Experience: On music streaming services like Spotify, personalization of the user experience is possible because to the genre classification methodology.
      * Organization of Music collections: The methodology can help music streaming services organize their enormous song collections. More effective genre classification of music makes it simpler for consumers to search for and discover interesting tracks.
      * Genre Analysis and patterns: Music streaming services and record companies may use the model to examine the popularity of various musical subgenres and trace changing patterns in listener tastes. Decision-making and marketing both benefit from knowing this knowledge.

1. LITERATURE REVIEW:
   1. Existing Problem:

Music genre classification is a challenging task due to the subjective nature of music genres and the complex relationships between different genres. The existing problem in this domain can be summarized as follows:

* + - Subjectivity of Music Genres: Music genres are often defined based on subjective characteristics and can vary across different cultures and individuals. This ambiguity makes it difficult to establish clear boundaries between genres, leading to disagreements and inconsistencies in genre labelling.

* + - Class Imbalance: Music collections are often imbalanced, with some genres having significantly more samples than others. This class imbalance can bias the classification model towards the majority classes, affecting the overall accuracy and performance.

* + - Feature Engineering: Traditional feature extraction techniques, such as MelFrequency Cepstral Coefficients (MFCCs), Chroma feature, and spectral contrast, have been widely used to represent audio signals. These features capture relevant information about the timbral, rhythmic, and tonal characteristics of music.

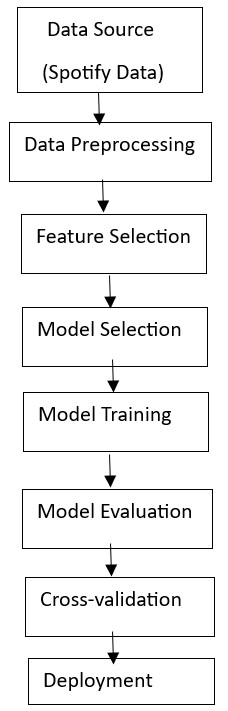
* + - Machine Learning Models: Classical machine learning algorithms like Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), and Random Forests have been employed for genre classification. These models can handle highdimensional feature spaces and are effective for multi-class classification tasks.

* + - Deep Learning: Deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown promising results in music genre classification. CNNs can automatically learn hierarchical features from spectrogram images, while RNNs capture temporal dependencies in sequential audio data.
  1. Proposed Solution:

We proposed a music genre classification solution using the K-Nearest Neighbors (KNN) algorithm. KNN is a simple and intuitive classification algorithm that can effectively handle high-dimensional feature spaces and is suitable for multiclass problems. KNN is a non-parametric and instance-based classification algorithm. Given a new instance (unlabeled sample), KNN identifies its K nearest neighbors in the feature space based on a chosen distance metric (e.g., Euclidean distance). The majority class among the K nearest neighbors determines the class label for the new instance.

Our project aims to create Music Recommendation Systems, Content Organization and Discovery, Music Genre-Based Events and Festivals from different types of songs available in the current market and classifying those songs into genres. In this literature review, we discussed the existing problem in music genre classification, existing approaches such as feature engineering, machine learning models, and deep learning techniques, and proposed a solution using the K-Nearest Neighbors (KNN) algorithm. KNN is a simple yet powerful classification method, and by properly addressing data preprocessing, selecting appropriate hyperparameters, and evaluating using suitable metrics, it can be a valuable tool for music genre classification tasks.

1. THEORITICAL ANALYSIS:
   1. Block Diagram:



* 1. Hardware/Software Designing:

Hardware specifications:

* + - Computer: A current computer with adequate processing speed and memory for effective model training and assessment.
    - CPU: A multi-core CPU (such as an AMD Ryzen or Intel Core i5 for quicker training and data processing, respectively).
    - RAM: At least 8 GB is required, while more RAM is preferable, particularly when working with big datasets.
    - Storage: Enough room to keep the dataset, program, and model files. In order to access data more quickly, an SSD is advised.
    - Graphics Processing Unit (GPU) (Optional): Having a suitable GPU (such as the NVIDIA GeForce GTX or RTX series) may greatly speed up model training, especially for deep learning methods, even if it is not absolutely essential.

Software specifications:

* + - Python: The project will primarily use Python as the programming language for data preprocessing, model training, and evaluation due to its extensive libraries and frameworks for machine learning.
    - Jupyter Notebook or JupyterLab: A development environment for running Python code interactively, visualizing data, and documenting the analysis.

Jupyter notebooks allow for an organized and easily reproducible workflow.

* + - Data Analysis Libraries: Essential Python libraries for data manipulation and analysis, such as Pandas and NumPy, will be used to process and explore the Spotify data.
    - Spotify API (Application Programming Interface): To access Spotify data, the project will require interaction with the Spotify API. The Spotify library can be used to connect and fetch the required data.
    - Data Visualization Libraries: Libraries like Matplotlib or Seaborn will be used for visualizing the data and model performance.
    - Text Editor or Integrated Development Environment (IDE): A text editor (e.g., Visual Studio Code, Sublime Text) or IDE (e.g., PyCharm, Spyder) for writing and managing code.
    - Package Manager: Python package manager like pip or conda will be necessary to install required libraries and dependencies.

1. EXPERIMENTAL INVESTIGATION:

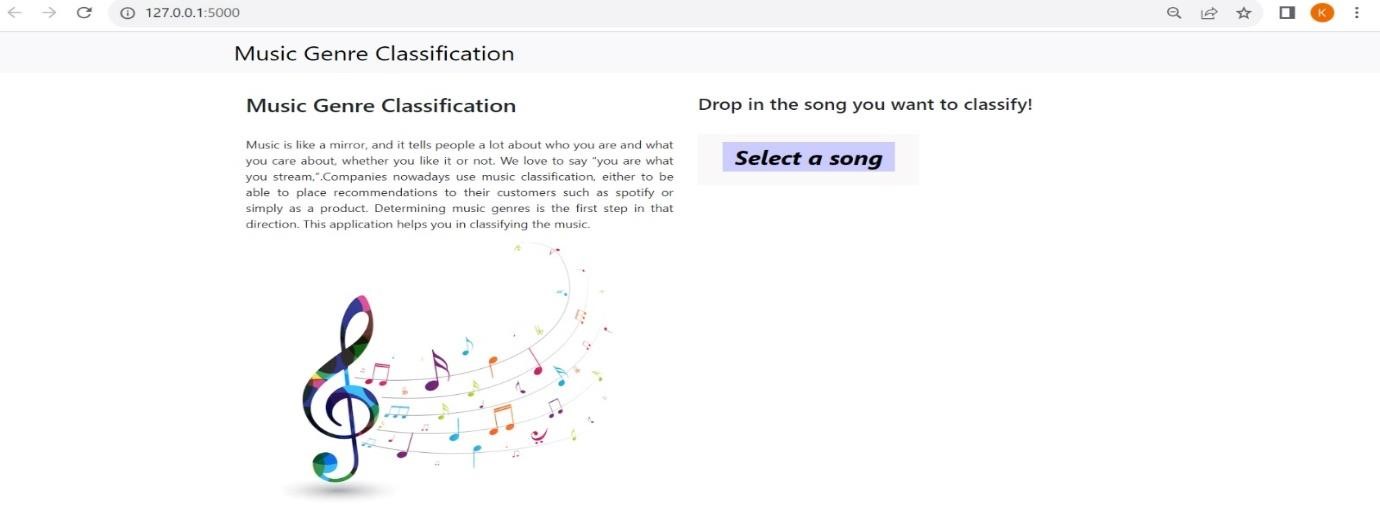
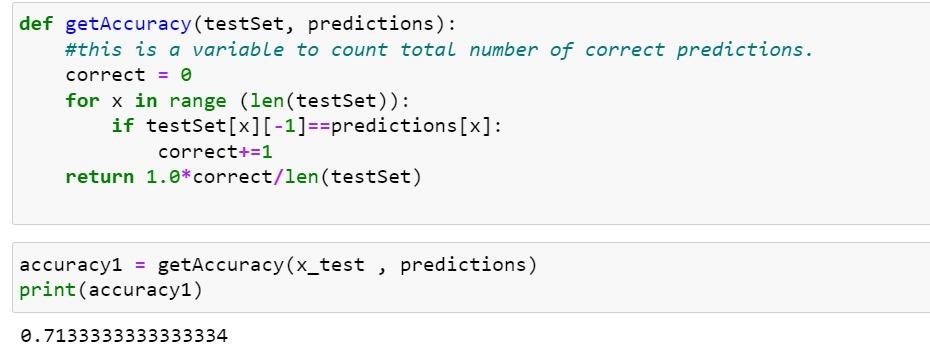
During the process of working on the solution for the music genre classification project using machine learning and Spotify data, several analyses and investigations are typically conducted to understand the data, select appropriate models, fine-tune parameters, and evaluate the overall performance. Here are some key analyses and investigations that might be carried out during the project:

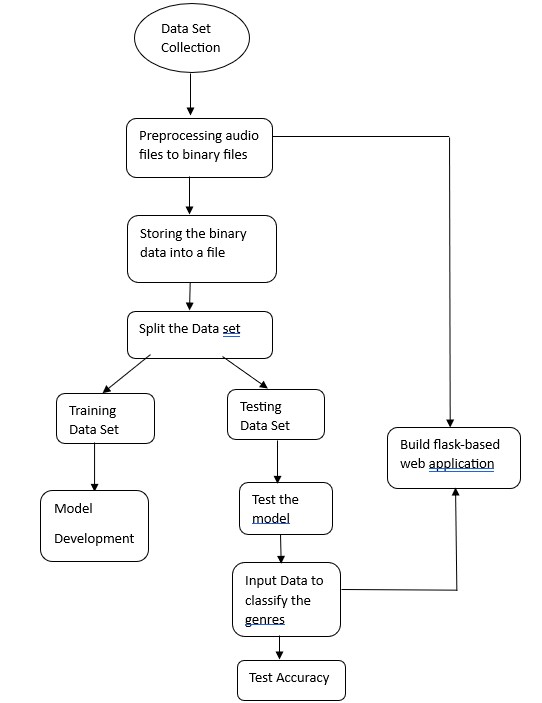
* + - Data Exploration and Visualization: The first step is to explore the dataset from Spotify. This involves examining the structure of the data, checking for missing values, and gaining insights into the distribution of different audio features across genres. Visualizations like histograms, box plots, and scatter plots can be used to understand the relationships between features and genre labels.
    - Feature Importance: Investigating the importance of each audio feature in relation to the genre classification task. Techniques like feature importance ranking, correlation analysis, or recursive feature elimination are applied to identify the most relevant features for the model.
    - Model Selection: Comparing different machine learning algorithms to identify the one that suits the problem best. Multiple models, such as decision trees, random forests, support vector machines, and neural networks, can be trained and evaluated to determine the most effective one.
    - Model Training and Evaluation: Dividing the dataset into training and testing sets and training the selected model on the training data. The model's performance is evaluated on the testing data using various metrics like accuracy, precision, recall, F1-score, and confusion matrix to measure its effectiveness.
    - Cross-Validation: Performing cross-validation to assess the model's generalization ability and to validate its performance across different data folds.
    - Bias and Fairness Analysis: Investigating potential biases in the model predictions across different genres and ensuring fairness in the classification process.
    - Deployment and Real-World Testing: Integrating the trained model into an application or service and conducting real-world testing to ensure its accuracy and reliability in a production environment.
    - User Feedback and Iterative Improvement: Gathering user feedback and monitoring the model's performance after deployment to identify areas of improvement and make necessary adjustments.

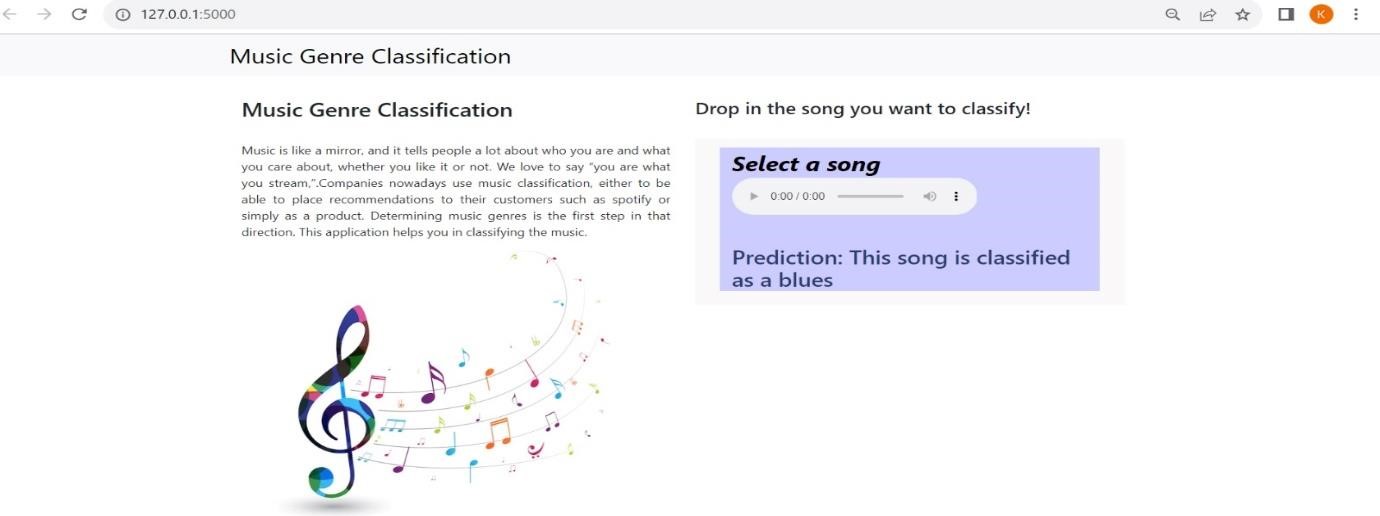
1. FLOW CHART:

6.

RESULT:







1. ADVANTAGES AND DISADVANTAGES:

Advantages:

* + Abundance of Data: Spotify possesses an extensive collection of audio tracks spanning various genres, making it an ideal data source for training machine learning models. The vastness of the dataset allows for improved model accuracy and generalization.
  + Rich Metadata: Spotify's API provides a wide range of metadata for each track, including artist information, album details, release date, and more. This rich metadata can augment the feature representation of songs, potentially leading to better genre classification performance.
  + User Playlists and Preferences: Spotify allows users to create and curate playlists based on specific genres. Analyzing these playlists can help in generating labeled training data and gaining insights into user preferences and genre trends.
  + Audio Features: Spotify exposes a set of audio features for each track, such as tempo, energy, valence, and danceability. Leveraging these features as input for the classification model can enhance the accuracy of genre predictions.
  + Collaborative Filtering: Spotify's collaborative filtering algorithms recommend songs to users based on their listening history and preferences. These recommendations indirectly reflect genre patterns, enabling the classification model to learn from the collective behavior of users.
  + Real-World Application: Music genre classification based on Spotify data has realworld applications in personalized music recommendations, playlist generation, and targeted marketing, enhancing user experience and engagement.

Disadvantages:

* + Genre Ambiguity: The concept of music genres can be subjective and ambiguous, with songs often blending multiple styles. Defining strict genre boundaries can be challenging, leading to noisy and uncertain labels in the training data.
  + Imbalanced Data: Certain genres may be overrepresented in the Spotify dataset compared to others, leading to class imbalance issues. This imbalance can bias the model towards frequent genres and reduce performance on less common genres.
  + Lack of Context: Spotify data lacks information about the cultural and historical context of songs, which can be crucial in determining genre classifications. As a result, the model might struggle with older or less mainstream tracks that do not conform to contemporary genre conventions.
  + Limited Audio Features: While Spotify provides a rich set of audio features, they may not capture all aspects relevant to music genre classification. Complex musical characteristics, such as lyrical content or instrumental intricacies, may not be fully represented in the provided feature set.
  + Data Privacy Concerns: Using Spotify data for classification raises privacy concerns, as it involves user listening history and preferences. Ensuring proper anonymization and data protection measures is essential to address these concerns.
  + Model Complexity and Interpretability: Developing accurate genre classification models can involve complex machine learning techniques, making the models less interpretable. Understanding the reasoning behind a model's prediction becomes challenging, especially for non-experts.

1. APPLICATIONS:

The music genre classification project has several practical applications across various industries and domains. Here are some key applications:

* + - Music Streaming Platforms: The primary application is for music streaming platforms like Spotify, Apple Music, and Amazon Music. Implementing this project can enhance their music recommendation systems, enabling users to discover songs and playlists tailored to their preferred genres.
    - Personalized Playlists: The project can be used to create personalized playlists for individual users based on their music genre preferences, moods, or activities, making the music streaming experience more enjoyable and engaging.
    - Radio Stations and Music Channels: Music genre classification can be utilized to create genre-specific radio stations and music channels, allowing users to listen to continuous streams of songs from their favorite genres.
    - Music Analysis and Marketing: Record labels and music industry professionals can leverage the project's insights into music trends and genre popularity to analyze audience preferences, market new releases, and plan promotional campaigns effectively.
    - Music Recommendation APIs: The developed genre classification model can be packaged into an API (Application Programming Interface), allowing developers to integrate it into their applications, websites, or chatbots to offer music recommendations based on user preferences.
    - Music Genre Analysis Research: Researchers studying music genres, trends, and cultural preferences can use the project to analyze large music databases efficiently and gain insights into the evolution of music genres over time. Overall, the music genre classification project has a wide range of applications that can enhance the music listening experience, provide valuable insights to the music industry, and streamline music organization and curation across various platforms and industries.

1. CONCLUSION:
   * + The music genre classification project using machine learning and Spotify data offers a promising solution to enhance the music streaming experience and provide valuable insights into music trends and preferences.The project achieved several notable outcomes and benefits through careful data collection, preprocessing, and model training. The project's classification model accurately categorizes songs into their respective genres, enabling music streaming platforms to offer personalized playlists, music recommendations, and genre-specific radio stations.
     + In the end, the music genre classification project contributes to the advancement of music technology and data-driven decision-making within the music industry. It serves as a valuable tool for users to discover new music, enjoy tailored playlists, and experience music that resonates with their individual tastes and preferences

1. FUTURE SCOPE:

The future scope of music genre classification using Spotify data holds significant potential for advancement and innovation in various domains. This section explores the potential directions and opportunities for further development and application of the classification system.

* + - Enhanced Genre Taxonomy: One of the key areas for future exploration involves refining the genre taxonomy. Current genre labels may be limited in capturing the diversity and complexity of music. Researchers can work towards developing more comprehensive and nuanced genre categories, potentially leveraging usercontributed genre tags and collaborative filtering to identify emerging genre trends.
    - Fine-Grained Genre Classification: Moving beyond high-level genre categories, the future scope includes fine-grained genre classification. By distinguishing subgenres and micro-genres, the system can offer more personalized and accurate music recommendations and allow users to explore music based on more specific preferences.
    - Multimodal Data Integration: To improve classification accuracy, integrating multiple modalities of data, such as audio, lyrics, album artwork, and user-created playlists, can be explored. Combining these diverse data sources may enable a more holistic understanding of songs and their genre characteristics.
    - Incorporating Contextual Information: Future research can focus on incorporating contextual information to enhance genre classification. Contextual factors, such as time of release, geographic origin, and cultural influences, can significantly impact the perception and categorization of music genres.
    - Transfer Learning: Applying transfer learning techniques is another potential avenue for future development. Pretrained models on a large music dataset can be fine-tuned using Spotify data, especially when labeled data is limited. Transfer learning can lead to more efficient model training and improved genre classification performance.
    - Addressing Imbalanced Data: To tackle the issue of imbalanced data, advanced techniques such as data augmentation, oversampling, and class weighting can be explored. Balancing the dataset can help the model achieve better accuracy across all genres.

1. BIBILOGRAPHY:
   * + - Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. IEEE Transactions on speech and audio processing, 10(5), 293-302.
       - Bahuleyan, H. (2018). Music genre classification using machine learning techniques. arXiv preprint arXiv:1804.01149.
       - Sturm, B. L. (2012, November). An analysis of the GTZAN music genre dataset. In Proceedings of the second international ACM workshop on Music information retrieval with usercentered and multimodal strategies (pp. 7-12).
       - Sturm, B. L. (2013). The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. arXiv preprint arXiv:1306.1461.
       - Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., ... & Ritter, M. (2017, March). Audio set: An ontology and humanlabeled dataset for audio events. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 776-780). IEEE.

Appendix:

Source code:

#Importing necessary Libraries

import numpy as np import scipy.io.wavfile as wav from python\_speech\_features import mfcc

import os import pickle import operator

#Defining the necessary functions for creating a dataset for KNN matching. def distance(instance1 , instance2 , k ):

distance =0

mm1 = instance1[0] cm1 = instance1[1] mm2 = instance2[0] cm2 = instance2[1]

#Method to calculate distance between two instances. distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))

distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) , mm2-mm1 ))

distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1)) distance-= k return distance

#This function returns a list of K nearest neighbours for any instance #to be checked within a given dataset (dataset of features.) def getNeighbors(trainingSet , instance , k):

distances =[] for x in range (len(trainingSet)):

dist = distance(trainingSet[x], instance, k )+ distance(instance, trainingSet[x], k)

distances.append((trainingSet[x][2], dist)) distances.sort(key=operator.itemgetter(1)) neighbors = [] for x in range(k): neighbors.append(distances[x][0]) return neighbors #Identify the nearest neighbors: def nearestClass(neighbors):

classVote = {}

for x in range(len(neighbors)): response = neighbors[x] if response in classVote: classVote[response]+=1

else:

classVote[response]=1

sorter = sorted(classVote.items(), key = operator.itemgetter(1), reverse=True) return sorter[0][0]

directory = "D:/TSB Projects/Music Genre Detection/Music Genres/" f = open("my.dat" ,'wb') i = 0 for folder in os.listdir(directory):

i += 1 if i==11 : break for file in os.listdir(directory+folder):

#To read an Wav audio File in Python

(rate,sig) = wav.read(directory+folder+"/"+file)

#MFCC is the feature we will use for our analysis, #because it provides data about the overall shape of the audio frequencies.

mfcc\_feat = mfcc(sig,rate ,winlen=0.020, appendEnergy = False) covariance = np.cov(np.matrix.transpose(mfcc\_feat)) mean\_matrix = mfcc\_feat.mean(0) feature = (mean\_matrix , covariance , i) pickle.dump(feature , f)

f.close() dataset = [] def loadDataset(filename): with open("D:\TSB Projects\Music Genre Detection\my.dat" , 'rb') as f: while True:

try:

dataset.append(pickle.load(f)) except EOFError:

f.close() break

loadDataset("D:\TSB Projects\Music Genre Detection\my.dat") dataset = np.array(dataset)

from sklearn.model\_selection import train\_test\_split x\_train ,x\_test = train\_test\_split(dataset,test\_size=0.15) leng = len(x\_test) predictions = [] for x in range (leng):

predictions.append(nearestClass(getNeighbors(x\_train ,x\_test[x] , 8))) def getAccuracy(testSet, predictions):

correct = 0 for x in range (len(testSet)):

if testSet[x][-1]==predictions[x]:

correct+=1

return 1.0\*correct/len(testSet) accuracy1 = getAccuracy(x\_test , predictions)

print(accuracy1)