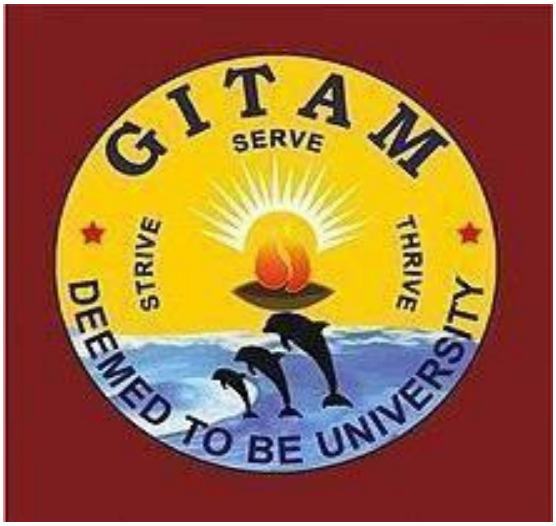
MUSIC RECOMMENDER SYSTEM



TEAM MEMBERS:

121810301006	**SDK Prasad**
121810301031	**S.Sai Sravya**
121810301033	**N.Dheeraj Kumar**
121810301043	**B.Karthikey**

MARCH 2022

MUSIC RECOMMENDER SYSTEM

A Project report submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE

ENGINEERING

Submitted By

SDK Prasad-121810301006

S.Sai Sravya-121810301031

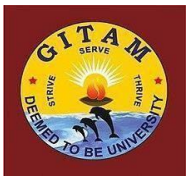
N.Dheeraj Kumar-121810301033

B.Karthikey-121810301043

Under the esteemed guidance of

S. N. V. Jitendra M

Assistant Professor,CSE



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(DEEMED TO BE UNIVERSITY)

VISAKHAPATNA

M MARCH-2022

DEPARTMENT OF COMPUTER SCIENCE

ENGINEERING GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled "**MUSIC RECOMMENDER SYSTEM**" is an original work done in the Department of Computer Science and Engineering,GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in InformationTechnology. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 05-04-2022

Registration No(s).	Name(s)	Signature(s)
121810301006	**SDK Prasad**	
121810301031	**S. Sai Sravya**	
121810301033	**N.Dheeraj Kumar**	
121810301043	**B.Karthikey**	

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING GITAM INSTITUTE OF

TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**MUSIC RECOMMENDER SYSTEM**" is a bonafide record of work carried out by **SDK Prasad(121810301006), S.Sai Sravya(121810301031), N.Dheeraj Kumar(121810301033), B.Karthikey (121810301043)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide

S.N.V. Jitendra M

Assistant Professor

Head of the Department

Head of the Department
Department of Computer Science & Engineering
GITAM Institute of Technology
Gandhi Institute of Technology and Management (GITAM)
(Deemed to be University)
Visakhapatnam-530 045

Dr. R.Sireesha

Professor

TABLE OF CONTENTS

1. ABSTRACT

Music is the art of arranging sounds in time through the elements of melody, harmony, rhythm, and timbre. It is one of the universal cultural aspects of all human societies. In the late centuries, no technology was present for exploring various sounds/genres in the music. This made it very difficult for the people to hear various kinds of music until recommender systems were created.

In the world of e-commerce, recommender systems are crucial. E-commerce is a large market that connects individuals from all over the world together in one location. It is now possible to access and reach the market while sitting at any location. In the field of recommendation, recommender systems play a crucial role. As it is a software tool that assists in showing or displaying information, commerce mobility goes smoothly and efficiently. By assessing the user's taste, things are recommended depending on their preferences.

In this project, we created a music recommender system which predicts the genre according to the user's age and gender. After comparing various prediction classifier algorithms, svc was found to be more accurate. Support vector classifier algorithm was used to accomplish this prediction. On the other hand, a gui using python tkinter was created where the user's details will be asked for. The details are then taken from the gui and predicted to find the most recommended genre for the user's age and gender.

2. INTRODUCTION

Music recommender systems (MRSs) research has recently attracted a lot of attention from both academia and industry. Music fans now have access to tens of millions of songs thanks to music streaming services like Spotify, Pandora, and Apple Music. MRSs are typically quite good at suggesting songs that meet their customers' preferences by filtering this multitude of music items, hence limiting choice overload. Such systems, on the other hand, are far from flawless and frequently produce disappointing recommendations. This is due in part to the fact that users' musical preferences and needs are highly dependent on a variety of factors that are not adequately considered in current MRS approaches, which are typically centered on the core concept of user–item interactions, or occasionally content-based item descriptors. We propose, on the other hand, that meeting users' musical enjoyment needs necessitates taking into consideration intrinsic, extrinsic, and contextual elements of listeners, as well as better interaction information. For example, musical tastes and needs are known to be influenced by the listeners' personality and emotional state (intrinsic) as well as their activities (extrinsic). Users' contextual elements, such as weather, social surroundings, and points of interest, are also taken into account. A music playlist or listening session's composition and annotation also convey information about which songs go well together or are appropriate for a specific occasion. As a result, MRS researchers and designers should think about their users holistically in order to create systems that are suited to each user's unique needs.

In light of this, we expand on what we perceive to be among the most significant contemporary difficulties in MRS research in this trends and survey piece, by examining the present state of the art and its limitations (Sect. 2). Because we won't be able to address all of the issues in depth, we'll concentrate on cold start, automated playlist continuation, and MRS evaluation. While these issues exist in other recommendation domains to some level, certain aspects of music present unique obstacles in these situations. The short duration of items (in comparison to movies), the high emotional connotation of music, and users' acceptance of duplicate recommendations are only a few of them. We provide our visions for future directions in MRS research in the second section (Sect. 3). We go over psychologically inspired music suggestion (taking human personality and emotion into account), situation-aware music recommendation, and culture-aware music recommendation in more detail. We wrap up this article with a summary and identification of prospective beginning places for researchers interested in tackling the issues raised (Sect. 4).

The authors' composition permits them to consider both academic and industry perspectives, which are both expressed in this essay. Furthermore, we'd like to point out that the Challenge 2: Automatic

playlist continuation principles described in Sect. 2 play a key role in the task description, planning, and execution of the ACM Recommender Systems Challenge 2018, which focuses on this use case. As a result, this article may also serve as an entry point for potential challenge participants.

3. LITERATURE REVIEW

In the last few years, computer networks (the World Wide Web) have seen a massive increase in data and the number of online users, particularly after the introduction of the Internet two.0. Customers may access databases of statistics and information in the initial model of the internet (Web1.0), but they were limited in terms of contribution and records accessibility. Over the other hand, Web two.0, the second generation of the WWW|World Wide Web|WWW|web|computer network, can be defined as a worldview that supports communication, ability, user-targeted arrangement, knowledge exchange, and collaboration on the internet. Furthermore, we are witnessing a noticeable shift from solitary and native to global collaboration and contribution in this age. Upgrades to the Internet two.0 have also changed the technique for creating and planning knowledge. Web 1.0, web 2.0 data management, and accessibility, on the other hand, are allocated and completed by implies that of a person's collaborations as hostile accessing, producing, and managing data on a specific laptop or browser. For example, Wikipedia is an obvious example of this collaboration in that anybody may access its content and edit it, as well as add new resources to it. Different models are uniting records and images sharing locations, sights, and everyone the other vital informal communities that are growing in popularity day by day. Recommender frameworks differ from Information Retrieval (IR) in several ways, although both focus on providing users with more information. It's possible that it also suggests systems based on Information Retrieval (IR) and Immediate Frequency (IF); additionally, IR has been compared to Recommender System (RS) in relevant and useful situations. Every performs a variety of duties and activities. There are three fundamental classifications for these distinctions. One is concerned with objects, while another is concerned with the problems of users, and the last is concerned with the general atmosphere. As a result, it is clear that RS is fundamentally the same because the IR system is consistent with Baeza-Yates (1999) in terms of aims and functions, but the operating methods are different. We tend to survey the state of acquisition in advised frameworks calculations and systems that are unit critical to distinguish the holes and improvement zones during this activity. With suggested frameworks and talking about recommender frameworks assessment methodologies, we tend to propose probable answers for beat shortcomings and proverbial problems.

EMBEDDING EMOTIONAL CONTEXT IN RECOMMENDATION SYSTEM (2005)

- In 2005, Gonzalez proposed a first model based on psychological aspects
- He uses Emotional Intelligence

● This is to improve music recommendations.

RECOMMENDER SYSTEM BASED ON PERSONALITY TRAITS (2008)

● The system used 30 facets of big 5 personality traits and only big 5 personality traits as the psychological measures of the users .

IMPROVING MUSIC RECOMMENDER SYSTEMS: WHAT CAN WE LEARN FROM RESEARCH ON MUSIC TASTES? (2014)

● It was published which discuss about the music tastes from a psychological point of view

● The proposal offers a great insight into how a recommendation engine can be improved with the personality via the series of steps.

A COMPARATIVE ANALYSIS OF PERSONALITY BASED MUSIC RECOMMENDER SYSTEMS (2016)

● It was distributed which depicts a fundamental report on considering data about the objective client's character in the music recommendation system.

● It proposes five diverse sorts of models for the character-based music suggestion system.

SMART STRESS RELIEVING MUSIC PLAYER USING INTELLIGENT SENTIMENT ANALYSIS (2018)

● Sayali Chavan has proposed a framework "XBeats-An Emotion Based Music Player" which utilizes Viola-Jones Algorithm for face discovery and Support Vector Machine for feeling recognition.

● Dolly Reney and Dr. Neeta Tripaathi made "An Efficient Method to Face and Emotion Detection" have recognized countenances from the information picture utilizing Viola-Jones face discovery calculation and assessed the face and feeling location utilizing K-Nearest Neighbour (KNN) classifier.

MUSIC RECOMMENDATION SYSTEM (2019)

● We are understanding the implementations of various models like popularity, collaboration, classification and trying to find the model which is best suitable as per cost, maintenance and user requirements.

● Models like popularity and classification can be implemented easily but the efficiency of collaborative is very high when compared to other models.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

A music recommendation system, should provide excellent suggestions to users in order to solve the data difficulties. However, it has become evident that the music recommendation algorithm does not meet the clients' needs. The question is, what is causing this problem? There are bound to be some flaws in the current music recommendation algorithm. A music recommender system comprises of a couple of various parts, for example,

- The recommendations cannot be accessed easily.
- The recommendations are not suitable for a person of certain age.
- For example, in general, people who are above 50 and more age in females in India want to listen to regional music or devotional music.
- Many music websites/applications have no information if the user is new and they just suggest the most popular songs as a default.
- It would be more helpful if the websites/apps take in the user's details that might be correlated to the genre they prefer.
- This would help the user to find the songs preferred by people of their age and gender, which might also be their preferred genre easily.

5. OBJECTIVES

The goal of the music recommendation system engine is to determine how accurate it is. The goal is to locate listeners looking for surprising song suggestions. The value of a machine is determined by how well it can generate fantastic recommendations that the user enjoys.

The objective of the system is to suggest songs of the genre that the user might like by taking in their age and gender. There are many websites and applications that do not work on getting the user's information to suggest the songs of their preferred genre in the very beginning. This might be a huge problem and an inconvenience to the users to find their songs when they first start using the music player.

6. SCOPE :

Musical genre also has significant importance beyond simply its utility in organizing and exploring music, and should not be evaluated solely in terms of commercial applicability. Many individuals actively identify culturally with certain genres of music easily. Genre is so important to listeners, additionally found that categorization in general plays an essential role in music appreciation. So, it has to be added as a feature to every music application in order to make it user friendly and also make users happy.

7. SYSTEM METHODOLOGY:

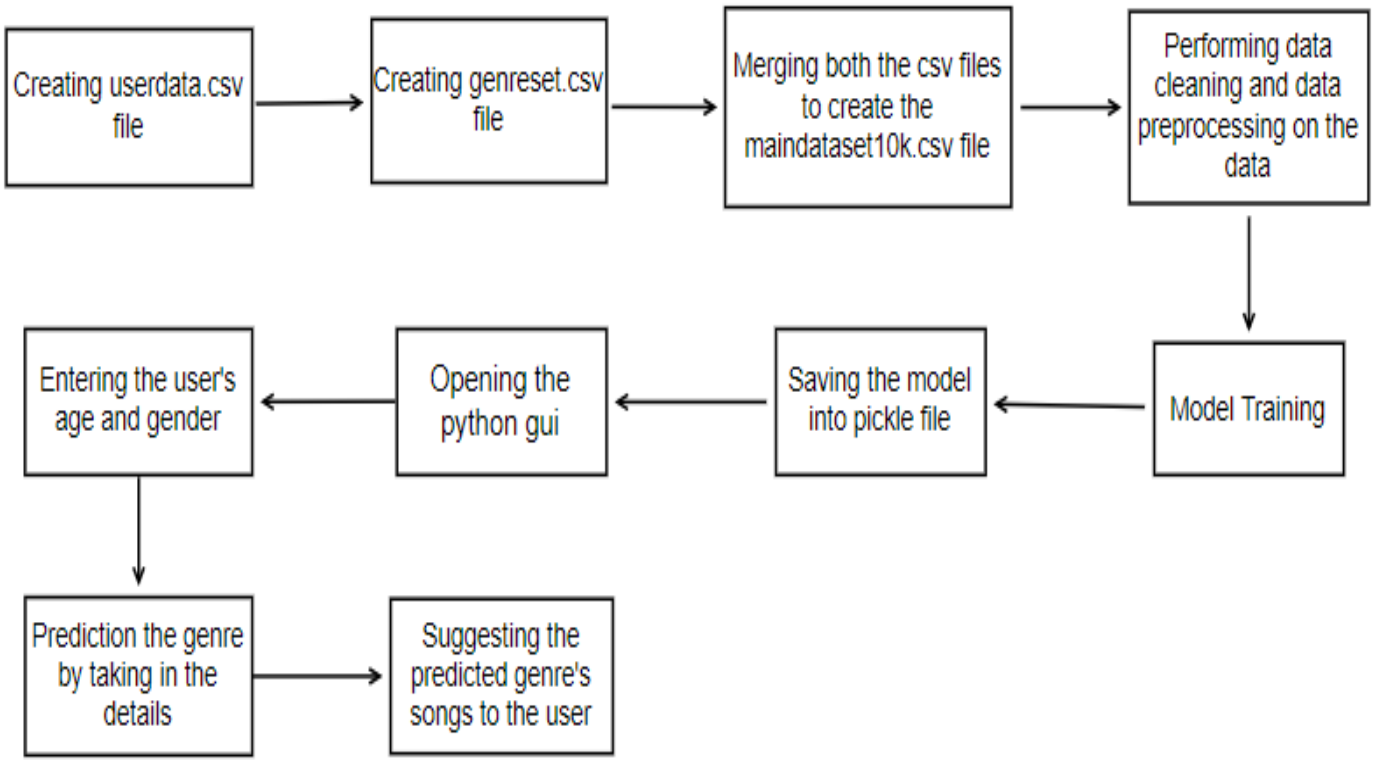


Fig 1: Major steps involved in this project

Support vector classifier is one of the most successful and efficient algorithms to predict user interest and recommend highly personalized content to users in recommender systems. For our project, we have predicted the user genre by taking in his age and gender. After the comparison between various algorithms SVC algorithm was more accurate for our data.

8. OVERVIEW OF TECHNOLOGIES

Information of all imported packages:

(i) Pandas:

Pandas is a Python toolkit for data manipulation that provides versatile and expressive data structures (such as dataframes and series). Pandas, which is built on top of numpy, is just as quick as numpy but much easier to use.

(ii) Numpy:

Numpy is a data management library that allows us to work with massive multi-dimensional arrays as well as a vast number of mathematical operations. Here's a quick demonstration of numpy in action.

(iii) Scikit-learn - SKlearn:

Scikit-learn is undoubtedly Python's most helpful machine learning library. Classification, regression, clustering, and dimensionality reduction are just a few of the useful capabilities in the sklearn toolkit for machine learning and statistical modelling. Machine learning models are built with sklearn. It should not be used for data reading, manipulation, or summarization. There are libraries that are better for that (e.g. NumPy, Pandas etc.).

(iv) Matplotlib:

Matplotlib is a Python package that allows you to create static, animated, and interactive visualizations. Matplotlib is a data visualization and graphical plotting package for Python and its numerical extension NumPy that runs on all platforms. As a result, it provides an open source alternative to MATLAB. Matplotlib's APIs (Application Programming Interfaces) can also be used to incorporate charts in graphical user interfaces.

(v) Decision Tree:

Decision Tree is a supervised learning technique that may be used to solve both classification and regression problems, however it is most commonly employed to solve

classification issues. Internal nodes represent dataset attributes, branches represent decision rules, and each leaf node provides the conclusion in this tree-structured classifier.

(vi) Random Forest:

Decision Tree is a supervised learning technique that may be used to solve both classification and regression problems, however it is most commonly employed to solve classification issues. Internal nodes represent dataset attributes, branches represent decision rules, and each leaf node provides the conclusion in this tree-structured classifier.

(vii) KNN :

The K-Nearest Neighbor method is one of the most fundamental Machine Learning algorithms. It is based on the Supervised Learning technique. The K-NN approach assumes that the new case/data and old cases are similar and places the new case in the most similar category to the existing categories.

The K-NN approach saves all available data and categorises new data points depending on how similar they are to the current data. This means that utilising the K-NN approach, fresh data can be swiftly sorted into a well-defined category. Although the K-NN method can be used for both regression and classification, it is most typically employed for classification.

(viii) SVM (SVC):

The Support Vector Machine, or SVM, is a popular Supervised Learning technique that may be used to solve both classification and regression issues. However, it is mostly utilised in Machine Learning for Classification difficulties.

The SVM algorithm's purpose is to find the optimum line or decision boundary for categorising n-dimensional space into classes so that additional data points can be readily placed in the correct category in the future. A hyperplane is the name for the optimal choice boundary.

(ix) XGBoost Classifier:

In the scikit-learn framework, XGBoost provides a wrapper class that allows models to be treated as classifiers or regressors. This means that XGBoost models can use the entire scikit-learn library. XGBClassifier is the XGBoost model for classification. We can make it and then fit it to our training data.

(x) MLP Classifier :

MLPClassifier stands for Multi-layer Perceptron Classifier, which is linked to a Neural Network by its name. Unlike other classification methods such as Support Vectors or Naive Bayes Classifier, MLPClassifier does classification using an underlying Neural Network.

A neural network is a set of algorithms that attempts to recognise underlying relationships in a batch of data using a method that mimics how the human brain works. Neural networks, in this context, refer to systems of neurons that can be organic or artificial in nature.

9. Machine learning:

Machine learning is significant because it allows businesses to see trends in customer behaviour and business operating patterns while also assisting in the development of new goods. Machine learning is at the heart of many of today's most successful businesses, like Facebook, Google, and Uber. For many businesses, machine learning has become a crucial competitive differentiation.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

1. Supervised Machine Learning:

Supervised machine learning, as the name implies, is based on supervision. It means that in the supervised learning technique, we train the machines with a "labelled" dataset, and the machine guesses the output based on the training. Some of the inputs are already mapped to the output, as indicated by the labelled data. We may put it another way: first, we train the machine using the input and output, and then we ask it to predict the outcome using the test dataset.

2. Unsupervised Machine Learning:

Unsupervised learning differs from supervised learning in that there is no requirement for supervision, as the term implies. It indicates that the system is trained with an unlabeled dataset and predicts the output without any supervision in unsupervised machine learning.

Models are trained with data that is neither classified nor labelled in unsupervised learning, and the model operates on the data without any supervision.

3. Semi-Supervised Learning:

Semi-supervised learning is a machine learning algorithm that falls between between supervised and unsupervised learning. It is the middle ground between Supervised (With) and Unsupervised

(Without).

During the training period, supervised learning (with labelled training data) and unsupervised learning (without labelled training data) techniques are used, using a combination of labelled and unlabeled datasets.

Although semi-supervised learning acts on data with a few labels and is the middle ground between supervised and unsupervised learning, it largely consists of unlabeled data. Labels are expensive, thus they may only have a handful for corporate purposes. It is distinct from supervised and unsupervised learning, which are differentiated by the presence or absence of labels.

4. Reinforcement Learning:

Reinforcement learning is a feedback-based process in which an AI agent (a software component) explores its surroundings automatically by striking and trailing, taking action, learning from its experiences, and improving its performance. The purpose of a reinforcement learning agent is to maximise the rewards for each good behaviour and to minimise the punishments for each negative activity.

There is no labelled data in reinforcement learning, unlike supervised learning, and agents learn solely from their experiences.

10. MLP(Multiple Layer Perceptron):

A feedforward artificial neural network called a multilayer perceptron (MLP) generates a set of outputs from a collection of inputs. Several layers of input nodes are connected as a directed graph between the input and output layers of an MLP. Backpropagation is used by MLP to train the network. MLP is a method of deep learning.

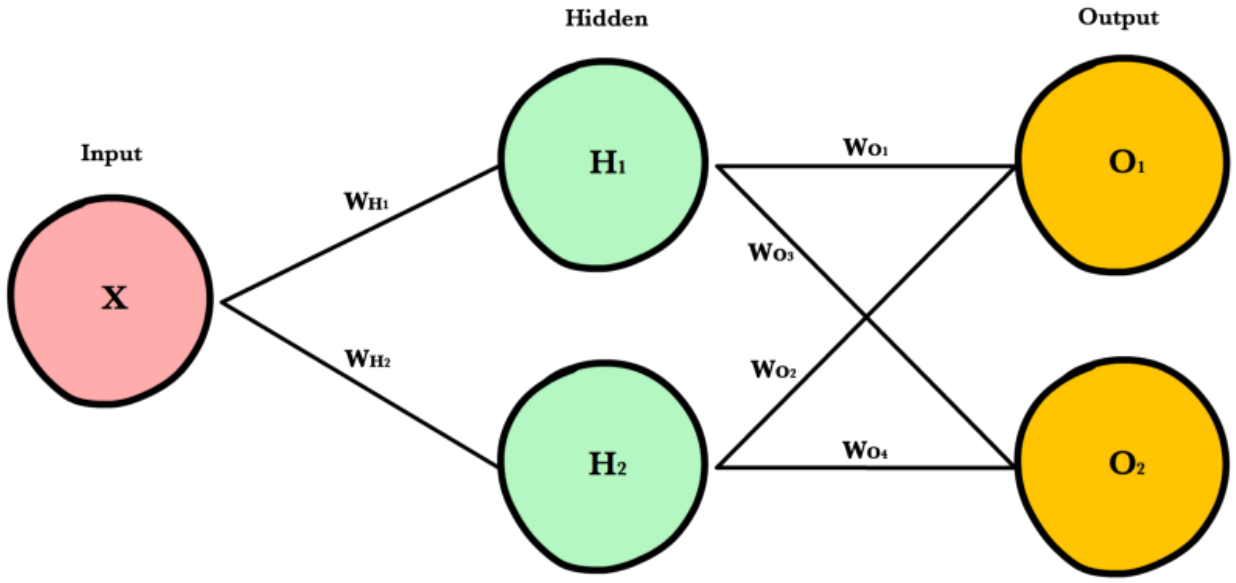
A multilayer perceptron is a neural network that connects multiple layers in a directed graph, which implies that the signal only goes one direction through the nodes. Aside from the input nodes,

each node has a nonlinear activation function. Backpropagation is a supervised learning technique used by an MLP. MLP is a deep learning technique since it uses numerous layers of neurons.

MLP is frequently used in supervised learning issues, as well as in computational neuroscience and parallel distributed processing research. Speech recognition, picture recognition, and machine translation are examples of applications.

11. Neural Networks

Neural networks are a type of machine learning technique that uses numerous hidden layers and non-linear activation functions to describe complicated patterns in datasets. A neural network takes an input, runs it through multiple layers of hidden neurons (mini-functions with unique coefficients that must be learned), and then outputs a prediction that is the sum of all the neurons' inputs.



Iterative optimization techniques such as gradient descent are used to train neural networks. An error measure is calculated after each training cycle based on the difference between prediction and target. Backpropagation is a technique that calculates the derivatives of this error metric and propagates them back across the network. The coefficients (weights) of each neuron are then changed based on how much they contributed to the total mistake. This technique is done until the network error falls below a threshold that is acceptable.

Advantage of Using Artificial Neural Networks:

- In ANNs, a problem might have numerous instances, each of which is represented by a set of attribute-value pairs.
- ANNs used to solve issues with a target function output can be discrete, real, or a vector of

many real or discrete-valued properties.

- Noise in the training data is not a problem for ANN learning algorithms. There may be faults in the training samples, but they will have no effect on the final output.
- It is commonly utilised in situations where a quick evaluation of the learned target function is necessary.
- The number of weights in the network, the number of training instances considered, and the settings of various learning algorithm parameters can all contribute to long training durations for ANNs.

The McCulloch-Pitts Model of Neuron:

Warren McCulloch and Walter Pitts presented the first artificial neuron model in 1943. The linear threshold gate is another name for the McCulloch-Pitts neural model. It is a neuron with a single output y and a set of inputs I1, I2,..., Im. The linear threshold gate simply divides a collection of inputs into two categories. As a result, the output y is binary. These equations can be used to explain such a function mathematically:

$$Sum = \sum_{i=1}^{N} I_i W_i,$$
$$y = f(Sum)$$

W1,W2,W3…. Wn are weight values that are associated with each input line and are normalised in the range of (0,1) or (-1,1). Sum is a threshold constant that represents the weighted sum. At the threshold, the function f is a linear step function.

Single-layer Neural Networks (Perceptrons)

The input is multi-dimensional (i.e., it can be a vector): input x = input y = input z = input z = input y = input z = input z = input z = input (I1, I2, .., In)

Input nodes (or units) are normally fully connected to a node (or multiple nodes) in the following tier. The weighted sum of all the inputs is taken by a node in the next layer:

$$SummedInput = \sum_i w_i I_i$$

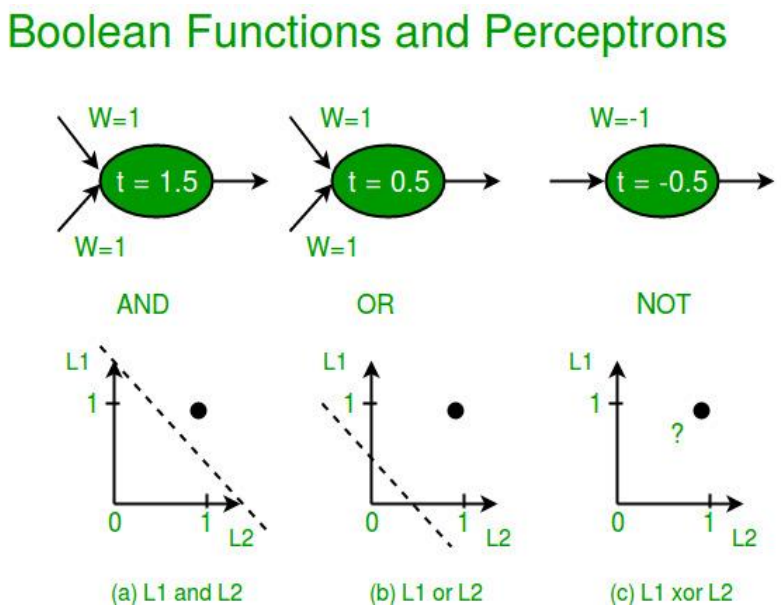
The rule:

There is a "threshold" t on the output node.

If the sum of the inputs is less than t, it "fires" (output y = 1). Otherwise, it doesn't fire (output y = 0) (summed input t).

$$\begin{aligned} &if \sum_i w_i I_i \geqslant t \\ &\quad \text{then } y{=}1 \\ &\text{else (if } \sum_i w_i I_i < t) \\ &\quad \text{then } y{=}0 \end{aligned}$$

which



Limitations of Perceptrons:

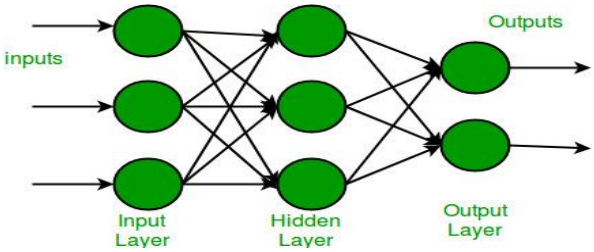
I Due to the hard-limit transfer function, a perceptron's output values can only take one of two values (0 or 1).

(ii) Perceptrons can only classify sets of vectors that are linearly separable. The input vectors are linearly separable if they can be separated into their right categories using a straight line or a plane. Learning will never reach a point where all vectors are correctly classified if the vectors are not linearly separable.

The Boolean function is a type of logic. XOR cannot be separated in a linear manner (Its positive and negative instances cannot be separated by a line or hyperplane). As a result, a single layer perceptron will never be able to compute the XOR function. This is a significant flaw that has previously caused the field of neural networks to stagnate. Multi-layer, on the other hand, has solved this problem.

Multi-layer Neural Networks

One or more hidden layers make up a Multi-Layer Perceptron (MLP) or Multi-Layer Neural Network (apart from one input and one output layer). A multi-layer perceptron can learn non-linear functions as well as linear functions, but a single layer perceptron can only learn linear functions.



This neuron receives x1,x2,....,x3 (plus a +1 bias term) as input and produces f(summed inputs+bias), where f(.) is the activation function. The fundamental purpose of Bias is to offer a trainable constant value to each node (in addition to the normal inputs that the node receives). Every activation function (or non-linearity) starts with a single number and applies a predetermined mathematical operation to it. In practise, you might come across the following activation functions:

Sigmoid:takes real-valued input and squashes it to range between 0 and 1.

$$\sigma(x) = \frac{1}{(1+exp(-x))}$$

tanh:takes real-valued input and squashes it to the range [-1, 1].

$$tanh(x) = 2\sigma(2x) - 1$$

ReLu: ReLu stands for Rectified Linear Units. It takes real-valued input and thresholds it to 0 (replaces negative values to 0).

$$f(x) = max(0, x)$$

12. DATASET INFORMATION

The dataset, our approach to modelling music preferences on a user level, and how we analyse a user group's preferences and homogeneity of these choices are all described in the following sections.

Dataset

The data was gathered from the Department of Computational Perception's webpage at http://www.cp.jku.at/datasets/LFM-1b/. The study Online Music Listening Culture of Kids and Adolescents Listening, Analysis, and Music Recommendation Tailored to the Young utilises the dataset's implementation. The collection is massive and contains numerous files.

We use the LFM-1b dataset, which consists of 1,088,161,692 individual listening events created by 120,175 Last.fm users who listened to 585,095 unique artists after data purification. 46,120 (38.4%) of the 120,175 individuals disclose age information on their profiles. Only those who give their age are counted, and there are 5,953 users aged 6 to 18 (inclusive) (12.9 percent). When users under the age of 25 are included, the figure rises to over two-thirds of the population (30,404 users or 65.9 percent).

Creating the main dataset

From the dataset, we retrieved two files: one with user id, age, and gender, and the other with genres and play counts for each user. The user id was used to join the two files. There are 15,000 rows in the dataset. User id, age, gender, and genre are the four columns. The users' ages, which range from 13 to 70, were taken into consideration.

13. IMPLEMENTATION

13.1 Coding

13.1.1 Visual Studio Code: Gui Using python tkinter

```
#! /usr/bin/env python3
from datetime import datetime
from typing import Container
from numpy import roots
import pygame as pg
import tkinter as tk
from tkinter import Entry
from tkinter import filedialog, messagebox
from os import chdir, listdir, sys
from functools import partial

import pickle
loaded_model = pickle.load(open(r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\filename_model_mlp.sav', 'rb'))

PAUSE = pg.USEREVENT+1
TRACK_END = pg.USEREVENT+1



class MusicFolder:
    def __init__(self):
        self.folder = None
```

```

    def get_files(self, tracklist, *args, **kwargs):
        self.folder = kwargs['folder']
        tracklist.delete(0, tk.END)
        chdir(self.folder)
        tracks = []
        formats = ['mp3', 'wav', 'ogg']
        playlist = listdir()
        for track in playlist:
            if track[-3:] in formats:
                tracklist.insert(tk.END, track)
        tracklist.select_set(0)

class Controls:
    def __init__(self):
        pass

    def play(self, *args, **kwargs):
        file = f'{kwargs["folder"]}/{kwargs["active"]}'
        pg.mixer.music.set_endevent(TRACK_END)
        try :
            pg.mixer.music.load(file)
            pg.mixer.music.play()
```

```
        except Exception:
            messagebox.showerror(title='No folder selected', \
            message='You must select a music folder to load the play list. The program will exit now.')
            sys.exit()

    def pause(self):
        pg.mixer.music.pause()

    def unpause(self):
        pg.mixer.music.unpause()

    def next(self, *args, **kwargs):
        pass

    def prev(self, *args, **kwargs):
        pass

    def stop(self, *args, **kwargs):
        pg.mixer.music.stop()
        pg.mixer.music.set_endevent()
        file = f'{kwargs["folder"]}/{kwargs["active"]}'
        pg.mixer.music.set_endevent(TRACK_END)
        pg.mixer.music.load(file)
```

```


class Player:
    def __init__(self, parent):
        self.parent = parent
        self.parent.update()
        self.width = self.parent.winfo_width()
        self.height = self.parent.winfo_height()
        self.parent.grid_columnconfigure(0, weight=1)
        self.parent.grid_rowconfigure(0, weight=1)
        bgcolor = 'light blue'
        fgcolor = 'navy'
        self.folder = None
        pg.init()
        pg.mixer.init()


        pg.mixer.music.set_endevent(TRACK_END)

        self.control = Controls()
        self.music = MusicFolder()

        # Container
```

```
        container = tk.Frame(self.parent)
        container.grid(column=0, row=0, sticky='news')

        # Contains the header image/canvas
        headerframe = tk.Frame(container)
        headerframe.grid(column=0, row=0, sticky='new', pady=2, padx=2)

        # Contains 3 columns of info - status/track/choose folder button
        frame1 = tk.Frame(container)
        frame1.grid(column=0, row=1, sticky='new', pady=2)

        # Status Label
        self.status_label = tk.Label(frame1, padx=8, bg=bgcolor, fg=fgcolor, \
        width=17, text='No status', anchor='w')
        self.status_label['bd'] = 1
        self.status_label['relief'] = 'ridge'
        self.status_label.grid(column=0, row=0, sticky='news', padx=2)

        # Track playing label
        self.track_label = tk.Label(frame1, padx=8, bg=bgcolor, fg=fgcolor, \
        width=70, text='No track is playing', anchor='w')
        self.track_label['bd'] = 1
        self.track_label['relief'] = 'ridge'
        self.track_label.grid(column=1, row=0, sticky='news', padx=4)


        # Button for populating our listbox with tracks
        self.button = tk.Button(frame1, text='Enter Age and Gender', \
        fg='navy', bg='lightsteelblue')
        self.button['command'] = partial(self.get_music)
        self.button.grid(column=2, row=0, sticky='new', padx=2)
        self.button.bind('<Enter>', partial(self.on_enter, self.button))
        self.button.bind('<Leave>', partial(self.on_exit, self.button))

        # Contains 3 columns - spacer/listbox/scrollbar
        frame2 = tk.Frame(container)
        frame2.grid(column=0, row=2, sticky='new', pady=2)
        frame2.grid_columnconfigure(0, weight=3)

        # Just a spacer label. May use to show album image?
        spacer_label = tk.Label(frame2, bg='silver', bd=1, relief='ridge')
        spacer_label['height'] = 15
        spacer_label['width'] = 30
        spacer_label.grid(column=0, row=0, sticky='news', padx=2)

        # Frame for listbox to give appearence of text not against side
```

```
        padframe = tk.Frame(frame2, bd=1, relief='ridge', bg='aliceblue', padx=8, \
        pady=5)
        padframe['highlightcolor'] = '#999999'
        padframe['highlightbackground'] = '#999999'
        padframe['highlightthickness'] = 1
        padframe.grid(column=2, row=0, sticky='news', padx=2)
        padframe.grid_rowconfigure(0, weight=3)
        padframe.grid_columnconfigure(0, weight=3)

        # Listbox and scrollbar
        self.scrollbar = tk.Scrollbar(frame2, orient='vertical')
        self.playlist = tk.Listbox(padframe, width=70, bd=0, bg='aliceblue')
        self.playlist['yscrollcommand'] = self.scrollbar.set
        self.playlist['selectmode'] = 'single'
        self.playlist['selectbackground'] = 'lightsteelblue'
        self.playlist['selectforeground'] = 'navy'
        self.playlist['highlightcolor'] = 'white'
        self.playlist['highlightbackground'] = 'white'
        self.playlist['highlightthickness'] = 0
        self.playlist['bd'] = 0
        self.playlist.grid(column=0, row=0, sticky='news')
        self.scrollbar.grid(column=3, row=0, sticky='ns', padx=2)
```

```
        # Contains the control buttons - play/stop/next/prev
        frame3 = tk.Frame(container)
        frame3.grid(column=0, row=3, sticky='new', pady=2)
        for i in range(4):
            frame3.grid_columnconfigure(i, weight=3, uniform='control_btns')

        # The buttons - play/stop/next/prev
        # play button will double as a pause button
        self.play_btn = tk.Button(frame3, text='Play', fg='navy', bg='lightsteelblue')
        self.play_btn.grid(column=0, row=0, sticky='new', padx=2)
        self.play_btn['command'] = partial(self.play, state='play')
        self.play_btn.bind('<Enter>', partial(self.on_enter, self.play_btn))
        self.play_btn.bind('<Leave>', partial(self.on_exit, self.play_btn))


        self.stop_btn = tk.Button(frame3, text='Stop', fg='navy', bg='lightsteelblue')
        self.stop_btn.grid(column=1, row=0, sticky='new', padx=2)
        self.stop_btn['command'] = partial(self.play, state='stop')
        self.stop_btn.bind('<Enter>', partial(self.on_enter, self.stop_btn))
        self.stop_btn.bind('<Leave>', partial(self.on_exit, self.stop_btn))
```

```
183         self.next_btn = tk.Button(frame3, text='Next', fg='navy', bg='lightsteelblue')
184         self.next_btn.grid(column=2, row=0, sticky='new', padx=2)
185         self.next_btn['command'] = partial(self.next)
186         self.next_btn.bind('<Enter>', partial(self.on_enter, self.next_btn))
187         self.next_btn.bind('<Leave>', partial(self.on_exit, self.next_btn))
188 
189         self.back_btn = tk.Button(frame3, text='Prev', fg='navy', bg='lightsteelblue')
190         self.back_btn.grid(column=3, row=0, sticky='new', padx=2)
191         self.back_btn['command'] = partial(self.prev)
192         self.back_btn.bind('<Enter>', partial(self.on_enter, self.back_btn))
193         self.back_btn.bind('<Leave>', partial(self.on_exit, self.back_btn))
194 
195 
196 
197     def get_music(self):
198         #06
199         def age():
200             if my_entry.get():
201                 # Get the current year
202 
203                 # Calculate The Age
204                 your_age = my_entry.get()
205                 your_gender = my_entry2.get()
206                 # Show age in message box
207 
208                 genderenumber = 0
209 
210                 if your_gender == 'male' or 'Male' or 'MALE' or 'M':
211                     genderenumber = 1
212 
213 
214                 result = loaded_model.predict([[your_age, genderenumber]])
215                 string_result = str(result)[2:-2]
216 
217                 messagebox.showinfo("Confirmation", f"Your age is {your_age}, your gender is {your_gender}
                    recommeded genre is {string_result}")
218                 if string_result == 'rock':
219                     self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\rock'
220 
221                 elif string_result == 'electronic':
222                     self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\electronic'
223 
224                 elif string_result == 'pop':
225                     self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\pop'
226 
227                 elif string_result == 'reggae':
```

```
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\reggae'

            elif string_result == 'rap':
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\rap'

            elif string_result == 'punk':
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\punk'

            elif string_result == 'blues':
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\blues'

            elif string_result == 'classical':
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\classical'

            elif string_result == 'folk':
                self.folder = r'C:\Users\SRAVYA\Documents\major_proj\music-recommender\folk'
            self.music.get_files(self.playlist, folder=self.folder)
            popup.destroy()

        else:
            # Show Error Message
            messagebox.showerror("Error", "You forgot to enter your age!")

```

```
    popup = tk.Tk()
    popup.title('Details')

    popup.geometry("500x300")

    my_label = tk.Label(popup, text="Enter Your Details", font=("Helvetica", 18))
    my_label.pack(pady=20)

    my_label1 = tk.Label(popup, text="Age", font=("Helvetica", 15))
    my_label1.pack(pady=0)
    my_entry = Entry(popup, font=("Helvetica", 12))
    my_entry.pack(pady=10)

    my_label2 = tk.Label(popup, text="Gender", font=("Helvetica", 15))
    my_label2.pack(pady=0)
    my_entry2 = Entry(popup, font=("Helvetica", 12))
    my_entry2.pack(pady=10)

    my_button = tk.Button(popup, text="OK!", font=("Helvetica", 18), command=age)
    my_button.pack(pady=20)

    popup.mainloop()
```

```
273
274
275         # self.folder = filedialog.askdirectory()
276         # self.music.get_files(self.playlist, folder=self.folder)
277
278     # Define some button animations
279     def on_enter(self, btn, event):
280         btn['bg'] = 'powderblue'
281         btn['fg'] = 'navy'
282         btn['cursor'] = 'hand2'
283
284     def on_exit(self, btn, event):
285         btn['bg'] = 'lightsteelblue'
286         btn['fg'] = 'navy'
287
288     def next(self):
289         pg.mixer.music.stop()
290         index = self.playlist.curselection()
291         if index:
292             next_index = 0
293             if len(index) > 0:
294                 last_index = int(index[-1])
295                 self.playlist.selection_clear(index)
296
297                 if last_index < self.playlist.size()-1:
298                     next_index = last_index + 1
299             self.playlist.activate(next_index)
300             self.playlist.selection_set(next_index)
301             self.status_label['text'] = 'Now Playing here'
302             self.play(state='play')
303         else:
304             pass
305
306     def prev(self):
307         try:
308             pg.mixer.music.stop()
309             index = self.playlist.curselection()
310             last_index = int(index[-1])
311             if last_index == 0:
312                 last_index = self.playlist.size()
313             self.playlist.selection_clear(index)
314             last_index = last_index - 1
315             self.playlist.activate(last_index)
316             self.playlist.selection_set(last_index)
317             self.play(state='play')
318         except Exception:
```

```
318             except Exception:
319                 pass
320 
321         def play(self, *args, **kwargs):
322             if self.playlist.get(tk.ACTIVE):
323                 state = kwargs['state']
324                 self.track_label['text'] = self.playlist.get(tk.ACTIVE)[:-4]
325 
326                 if state == 'play':
327                     self.play_btn['text'] = 'Pause'
328                     self.play_btn['command'] = partial(self.play, state='pause')
329                     self.status_label['text'] = 'Now Playing'
330                     self.control.play(active=self.playlist.get(tk.ACTIVE), folder=self.folder)
331 
332 
333                 elif state == 'pause':
334                     self.play_btn['text'] = 'Resume'
335                     self.play_btn['command'] = partial(self.play, state='unpause')
336                     self.status_label['text'] = 'Paused'
337                     self.control.pause()
338 
339                 elif state == 'unpause':
340                     self.play_btn['text'] = 'Pause'
```

```
341                     self.play_btn['command'] = partial(self.play, state='pause')
342                     self.status_label['text'] = 'Now Playing'
343                     self.control.unpause()
344 
345                 else:
346                     try:
347                         index = self.playlist.curselection()
348                         self.play_btn['text'] = 'Play'
349                         self.play_btn['command'] = partial(self.play, state='play')
350                         self.status_label['text'] = 'No status'
351                         self.track_label['text'] = 'No track is playing'
352                         self.playlist.selection_clear(index)
353                         self.playlist.select_set(0)
354                         self.playlist.activate(0)
355                         self.control.stop(folder=self.folder, active=self.playlist.get(tk.ACTIVE))
356                     except Exception:
357                         pass
358             else:
359                 messagebox.showerror(title='No folder selected.', message='Please choose a folder with music
                    files.')
360                 pass
361 
362 
```

```
363
364
365   def main():
366       root = tk.Tk()
367       root.title('Tkinter Music Player')
368       root.geometry('805x315+250+250')
369       root.resizable(0, 0)
370       root['padx'] = 10
371       root['pady'] = 5
372       Player(root)
373       root.mainloop()
374
375   if __name__ == "__main__":
376       main()
```

13.1.2 Jupyter notebook code: MLP algorithm Using Python

In [4]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\LFM-1b_users.txt",delimiter='\t')
df.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\LFM-1b_users.csv",index=None)
df1=df.drop(columns=['country','registered_unixtime','playcount'])
df1=df1.drop(df1.index[15000:])
df1=df1[df1.age!=-1]
df1=df1[~(df1.age <13)]
df1=df1[~(df1.age >70)]
df1=df1[df1.gender!='n']
df1.dropna(inplace=True)
df1["gender"].replace({"m": "1", "f": "0"}, inplace=True)
df1.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\userdata.csv",index=None)
df1
```

Out[4]:

	user_id	age	gender
0	384	35	1
4	3653	31	1
5	4813	43	1
7	5069	30	1
10	6958	36	1
...	...	...	...
14991	7706337	28	0
14993	7707627	30	1
14995	7708223	23	1
14997	7709118	22	1
14999	7709567	20	1

10137 rows × 3 columns

In [20]:

```
#cutting the original dataset, making a genre df with max value of each row and naming it with its respective column, appendi
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b_UGP\LFM-1b_UGP_weightedPC_allmusic.txt",delimiter='\t')
df.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b_UGP\LFM-1b_UGP_weightedPC_allmusic.csv",index=None)
df=df.drop(df.index[15000:])
df8=pd.DataFrame(df.iloc[:,1:].idxmax(axis=1).to_frame())
df8["genre"]=df.iloc[:,1:].idxmax(axis=1).to_frame()
df9=df["user_id"].to_frame()
genres=df9.join(df8)
genres=genres.drop(columns=0)
genres.dropna(inplace=True)
genres.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\genreset.csv",index=None)
genres
```

Out[20]:

	user_id	genre
0	384	rock
1	1206	alternative
2	2622	pop
3	2732	alternative
4	3653	rock
...	...	...
14995	7710411	alternative
14996	7710701	pop
14997	7711214	alternative
14998	7711719	alternative
14999	7712328	pop

15000 rows × 2 columns

In [11]:
```
import pandas as pd
df1=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\userdata.csv")
genres=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\genreset.csv")
dfinal = df1.merge(genres, on="user_id", how = 'inner')
dfinal.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv",index=None)
dfinal
```

Out[11]:

	user_id	age	gender	genre
0	384	35	1	rock
1	3653	31	1	rock
2	4813	43	1	alternative
3	5069	30	1	alternative
4	6958	36	1	alternative
...	...	...	...	...
10128	7706337	28	0	rock
10129	7707627	30	1	alternative
10130	7708223	23	1	electronic
10131	7709118	22	1	rock
10132	7709567	20	1	rock

In [12]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
df["genre"].replace({"alternative": "rock", "heavy metal": "rock", "new age":"reggae", "easy listening":"pop", "world":"pop",
df.to_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv",index=None)
df.genre.unique()
```

Out[12]:
```
array(['rock', 'electronic', 'pop', 'reggae', 'rap', 'punk', 'blues',
       'classical', 'folk'], dtype=object)
```

In [1]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10133 entries, 0 to 10132
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   user_id  10133 non-null  int64
 1   age      10133 non-null  int64
 2   gender   10133 non-null  int64
 3   genre    10133 non-null  object
dtypes: int64(3), object(1)
memory usage: 316.8+ KB
```

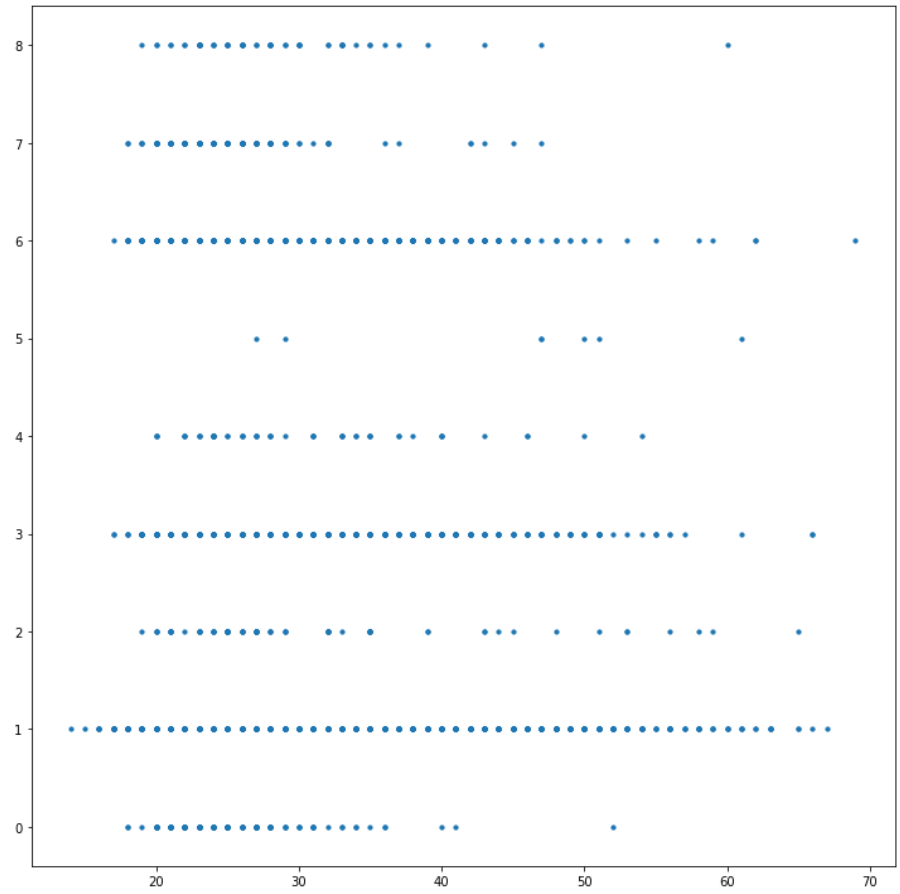
In [3]:

```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b_UGP\LFM-1b_UGP_weightedPC_allmusic.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120175 entries, 0 to 120174
Data columns (total 21 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   user_id         120175 non-null  int64
 1   rnb             120175 non-null  int64
 2   rap             120175 non-null  int64
 3   electronic      120175 non-null  int64
 4   rock            120175 non-null  int64
 5   new age         120175 non-null  int64
 6   classical       120175 non-null  int64
 7   reggae          120175 non-null  int64
 8   blues           120175 non-null  int64
 9   country         120175 non-null  int64
 10  world           120175 non-null  int64
 11  folk            120175 non-null  int64
 12  easy listening  120175 non-null  int64
 13  jazz            120175 non-null  int64
 14  vocal           120175 non-null  int64
 15  children's      120175 non-null  int64
 16  punk            120175 non-null  int64
 17  alternative     120175 non-null  int64
 18  spoken word     120175 non-null  int64
 19  pop             120175 non-null  int64
 20  heavy metal     120175 non-null  int64
dtypes: int64(21)
memory usage: 19.3 MB
```

In [5]:

```
import pandas as pd
from matplotlib import pyplot as plt
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
types = set(df['genre'])
listu = list(enumerate(types))
listu2 = []
for i in listu:
    listu2.append(i[::-1])
magic = dict(listu2)
x = list(df['genre'])
x1 = []
for i in x:
    x1.append(magic[i])
y = list(df['age'])
plt.rcParams['figure.figsize'] = [10, 10]
plt.scatter(y,x1,s = 10)
genre_labels = list(types)

plt.tight_layout()
plt.show()
```



In [13]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
df.age.unique()
```

Out[13]:
```
array([35, 31, 43, 30, 36, 51, 38, 45, 29, 33, 48, 28, 27, 37, 22, 40, 25,
       53, 39, 26, 24, 32, 21, 23, 47, 42, 34, 50, 52, 41, 20, 44, 46, 57,
       56, 58, 19, 54, 59, 18, 55, 61, 63, 62, 49, 65, 69, 66, 17, 14, 16,
       60, 67, 15], dtype=int64)
```

In [14]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
df.gender.unique()
```

Out[14]:
```
array([1, 0], dtype=int64)
```

In [15]:
```
import pandas as pd
df=pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
df.genre.unique()
```

Out[15]:
```
array(['rock', 'electronic', 'pop', 'reggae', 'rap', 'punk', 'blues',
       'classical', 'folk'], dtype=object)
```

In [16]:
```
import pandas as pd
import numpy as np

# create a dataframe with one column
df = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
# counting unique items
item_counts = df["genre"].value_counts()
print(item_counts)
```

```
rock          7922
electronic    1098
pop            698
rap            120
punk           111
blues           73
folk            54
classical       50
reggae           7
Name: genre, dtype: int64
```

In [17]:
```
##we need to separate the data in two parts
##One part to train the model and the other for testing
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']

model = DecisionTreeClassifier()
model.fit(x, y)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

Out[17]:
```
array(['rock', 'electronic'], dtype=object)
```

In [21]:
```
#training the model and calculating accuracy
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,random_state=10)

model = DecisionTreeClassifier()
model.fit(x_train, y_train)
predictions = model.predict(x_test)
score = accuracy_score(y_test, predictions)
score*100
```

Out[21]:
```
79.13172175629009
```

In [22]:
```
##we need to separate the data in two parts
##One part to train the model and the other for testing
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

classifier = KNeighborsClassifier(n_neighbors=5)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

Out[22]:
```
array(['rock', 'electronic'], dtype=object)
```

In [23]:
```
#training the model and calculating accuracy KNN
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,random_state=10)

scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)
predictions = classifier.predict(x_test)
score = accuracy_score(y_test, predictions)
score*100
```

Out[23]:
```
78.73704982733103
```

In [24]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

# Create the model with 100 trees
model = RandomForestClassifier(n_estimators=100, bootstrap = True, max_features = 'sqrt')
# Fit on training data
model.fit(x_train, y_train)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

Out[24]:
```
array(['rock', 'electronic'], dtype=object)
```

In [25]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,random_state=10)

# Create the model with 100 trees
model = RandomForestClassifier(n_estimators=100, bootstrap = True, max_features = 'sqrt')
# Fit on training data
model.fit(x_train, y_train)
predictions = model.predict(x_test)
score = accuracy_score(y_test, predictions)
score*100
```

Out[25]:
```
79.13172175629009
```

In [26]:
```
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

# Create the model with 100 trees
model = SVC()
# Fit on training data
model.fit(x_train, y_train)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

Out[26]:
```
array(['rock', 'rock'], dtype=object)
```

In [27]:
```
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=10)

# Create the model with 100 trees
model = SVC()
# Fit on training data
model.fit(x_train, y_train)
predictions = model.predict(x_test)
score = accuracy_score(y_test, predictions)
score*100
```

Out[27]:
```
79.18105574740997
```

In [28]:
```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

model=MLPClassifier()
model.fit(x_train,y_train)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

Out[28]:
```
array(['rock', 'rock'], dtype='<U10')
```

In [29]:
```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=10)

model=MLPClassifier()
model.fit(x_train,y_train)
predictions=model.predict(x_test)
score = accuracy_score(y_test, predictions)
score*100
```

Out[29]:
```
79.18105574740997
```

In [31]:
```
import xgboost as xgb
import pandas as pd
from sklearn.model_selection import train_test_split

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x.values, y.values, test_size = 0.2, random_state=10)

model= xgb.XGBClassifier()
model.fit(x_train,y_train)
predictions = model.predict([[63, 1], [63, 0]])
predictions
```

```
C:\Users\SRAVYA\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder i
is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass opt
coder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with
..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[16:20:39] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starti
3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'ml
tly set eval_metric if you'd like to restore the old behavior.
```

Out[31]:
```
array(['rock', 'rock'], dtype=object)
```

In [32]:
```
import xgboost as xgb
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=10)

model= xgb.XGBClassifier(n_estimators = 10)
model.fit(x_train,y_train)
predictions = model.predict(x_test)
score =accuracy_score(y_test, predictions)
score*100
```

```
[16:20:42] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: St
3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to
tly set eval_metric if you'd like to restore the old behavior.
```

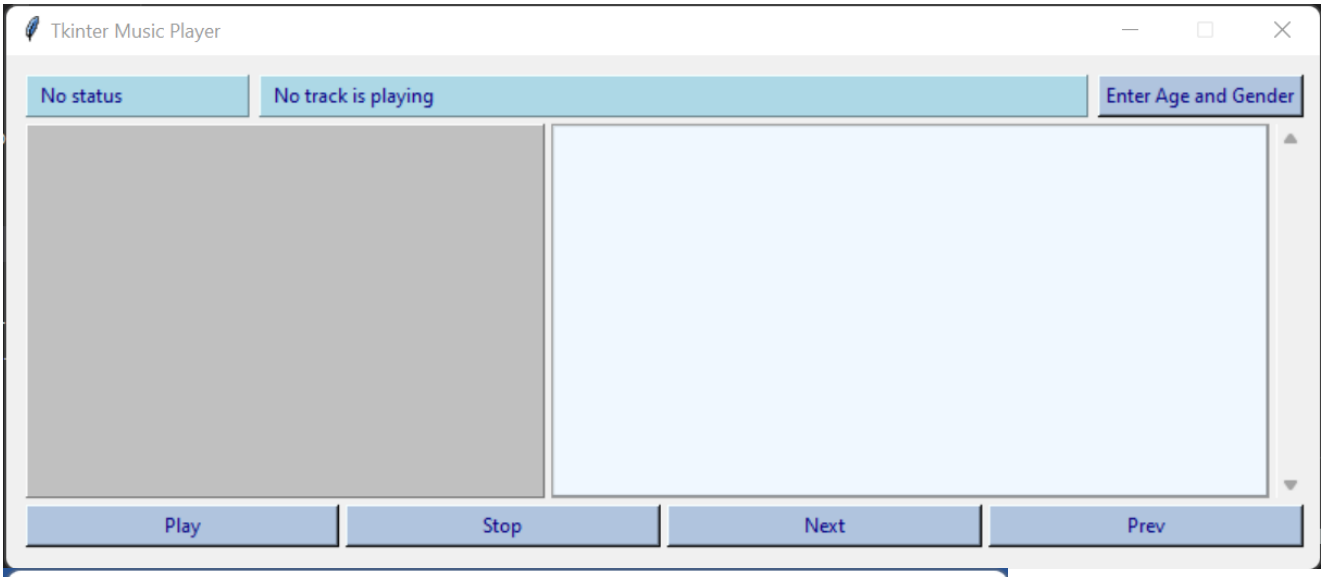
Out[32]:
```
79.13172175629009
```

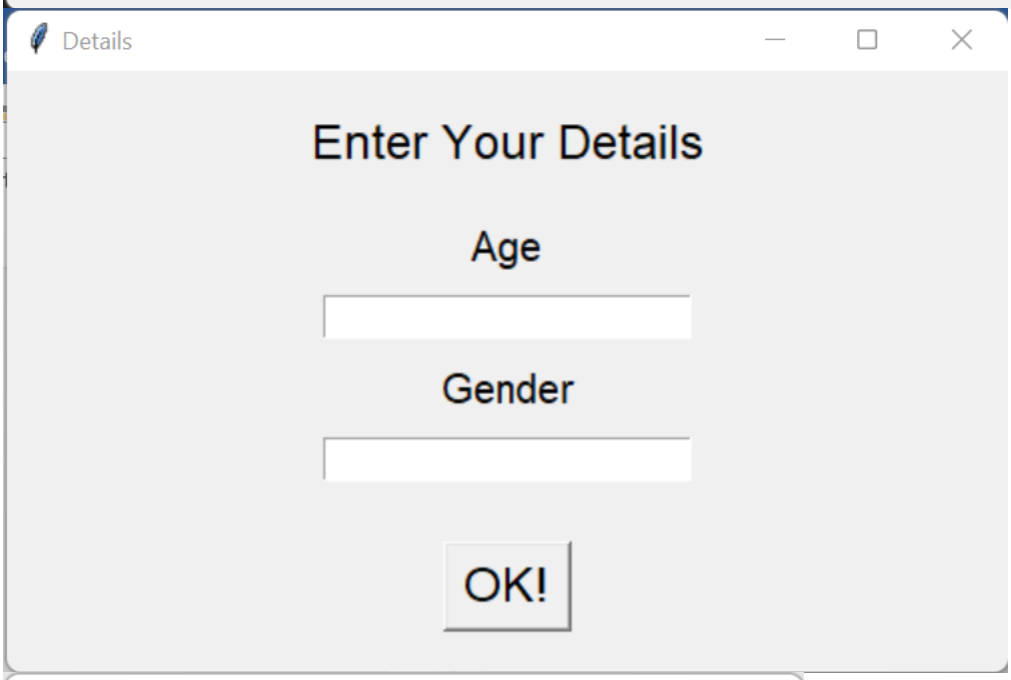
In [33]:

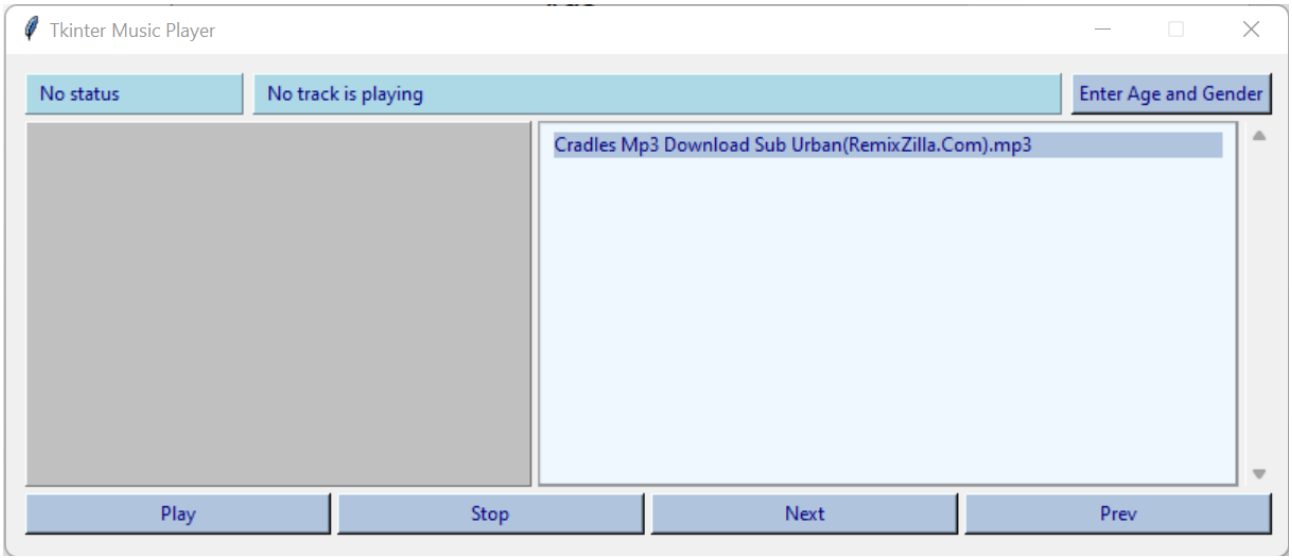
```
# the libraries we need
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('always')  # "error", "ignore", "always", "default", "module" or "once"

# separating data into training and test
music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

# first, initialize the classificators
tree= DecisionTreeClassifier(random_state=10) # using the random state for reproducibility
forest= RandomForestClassifier(random_state=10)
knn= KNeighborsClassifier()
svm= SVC(random_state=10)
mlp=MLPClassifier(random_state=10)
xg=xgb.XGBClassifier(n_estimators = 10)

# now, create a list with the objects
models= [tree, forest, knn,svm,mlp,xg]

for model in models:
    model.fit(x_train, y_train) # fit the model
    y_pred= model.predict(x_test) # then predict on the test set
    accuracy= accuracy_score(y_test, y_pred) # this gives us how often the algorithm predicted correctly
    clf_report= classification_report(y_test, y_pred) # with the report, we have a bigger picture, with precision and recall
    print(f"The accuracy of model {type(model).__name__} is {accuracy:.6f}")
    print(clf_report)
    print("\n")
```

14. RESULTS

Python tkinter gui:







Classification report:

The accuracy of model DecisionTreeClassifier is 0.783917

	precision	recall	f1-score	support
blues	0.00	0.00	0.00	11
classical	0.00	0.00	0.00	13
electronic	0.00	0.00	0.00	225
folk	0.00	0.00	0.00	8
pop	0.00	0.00	0.00	128
punk	0.00	0.00	0.00	24
rap	0.00	0.00	0.00	23
reggae	0.00	0.00	0.00	2
rock	0.79	1.00	0.88	1593
accuracy			0.78	2027
macro avg	0.09	0.11	0.10	2027
weighted avg	0.62	0.78	0.69	2027

The accuracy of model RandomForestClassifier is 0.784410

	precision	recall	f1-score	support
blues	0.00	0.00	0.00	11
classical	0.00	0.00	0.00	13
electronic	0.00	0.00	0.00	225
folk	0.00	0.00	0.00	8
pop	0.00	0.00	0.00	128
punk	0.00	0.00	0.00	24
rap	0.00	0.00	0.00	23
reggae	0.00	0.00	0.00	2
rock	0.79	1.00	0.88	1593
accuracy			0.78	2027
macro avg	0.09	0.11	0.10	2027
weighted avg	0.62	0.78	0.69	2027

```
The accuracy of model KNeighborsClassifier is 0.780957
              precision    recall  f1-score   support

       blues       0.00      0.00      0.00        11
   classical       0.00      0.00      0.00        13
  electronic       0.00      0.00      0.00       225
        folk       0.00      0.00      0.00         8
         pop       0.29      0.02      0.03       128
        punk       0.00      0.00      0.00        24
         rap       0.00      0.00      0.00        23
      reggae       0.00      0.00      0.00         2
        rock       0.79      0.99      0.88      1593

    accuracy                           0.78      2027
   macro avg       0.12      0.11      0.10      2027
weighted avg       0.64      0.78      0.69      2027

The accuracy of model SVC is 0.785890
              precision    recall  f1-score   support

       blues       0.00      0.00      0.00        11
   classical       0.00      0.00      0.00        13
  electronic       0.00      0.00      0.00       225
        folk       0.00      0.00      0.00         8
         pop       0.00      0.00      0.00       128
        punk       0.00      0.00      0.00        24
         rap       0.00      0.00      0.00        23
      reggae       0.00      0.00      0.00         2
        rock       0.79      1.00      0.88      1593

    accuracy                           0.79      2027
   macro avg       0.09      0.11      0.10      2027
weighted avg       0.62      0.79      0.69      2027

The accuracy of model MLPClassifier is 0.785890
              precision    recall  f1-score   support

       blues       0.00      0.00      0.00        11
   classical       0.00      0.00      0.00        13
  electronic       0.00      0.00      0.00       225
        folk       0.00      0.00      0.00         8
         pop       0.00      0.00      0.00       128
        punk       0.00      0.00      0.00        24
         rap       0.00      0.00      0.00        23
      reggae       0.00      0.00      0.00         2
        rock       0.79      1.00      0.88      1593

    accuracy                           0.79      2027
   macro avg       0.09      0.11      0.10      2027
weighted avg       0.62      0.79      0.69      2027

The accuracy of model XGBClassifier is 0.785397
              precision    recall  f1-score   support

       blues       0.00      0.00      0.00        11
   classical       0.00      0.00      0.00        13
  electronic       0.00      0.00      0.00       225
        folk       0.00      0.00      0.00         8
         pop       0.00      0.00      0.00       128
        punk       0.00      0.00      0.00        24
         rap       0.00      0.00      0.00        23
      reggae       0.00      0.00      0.00         2
        rock       0.79      1.00      0.88      1593

    accuracy                           0.79      2027
   macro avg       0.09      0.11      0.10      2027
weighted avg       0.62      0.79      0.69      2027
```

Saving the MLP model:

In [19]:
```
#persisting a model and saving it to a file
import pandas as pd
from sklearn.neural_network import MLPClassifier
import pickle

music_data = pd.read_csv(r"C:\Users\SRAVYA\OneDrive\Documents\major_proj\LFM-1b\maindataset10k.csv")
x = music_data.drop(columns=['user_id','genre'])
y = music_data['genre']


model = MLPClassifier()
model.fit(x, y)

filename='filename_model_mlp.sav'
pickle.dump(model,open(filename,'wb'))
```

In [20]:
```
#using the model created
import pandas as pd
from sklearn.neural_network import MLPClassifier

predictions = model.predict([[21,0]])
predictions[0]
```

Out[20]: 'rock'

Test Accuracy:

78.50%

15. CONCLUSION AND FUTURE SCOPE

By using training data sets, classification has become one of the most essential technologies and problem-solving strategies in machine learning. Almost everyone on the planet enjoys listening to music, and numerous music industries and businesses produce between 24,000 and 40,000 song tracks per day. Human life is intertwined with music. As a result, it serves numerous purposes and has an impact on human psychology.

This project taught us about the importance of machine learning in the music industry, as well as recommender systems. Python tkinter is also something we're familiar with. We gained enough experience and confidence from this project to create or work on another model.

REFERENCES

[1] Rajeeva Shreedhara Bhat#1 ,Rohit B. R.#2 ,Mamatha K. R.#3 3Assistant Professor Information Science and Engineering, B M S College Of Engineering, Bengaluru, India

[2] Lam Hoang. 2018. Literature Review about Music Genre Classification. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/ 1122445.1122456

[3] 1M.D.Nevetha, 2A.Nithyasree, 3A.Parveenbanu, 4Mrs.Jetlin CP 1Student, 2Student, 3Student, 4Assistant Professor Agni College of Technology

[4] Allamy, S., Koerich, A.L.: 1D CNN Architectures for Music Genre Classification. arXiv preprint arXiv:210507302 (2021).

[5] Bleeck, S., Ives, T., Patterson, R.: Aim-mat: the auditory image model in matlab. Acta Acust. Acust. 90, 781–787 (2004)

[6] Cano, P., Gômez, E., Gouyon, F., Herrera, P., Koppenberger, M., Ong, B., Serra, X., Streich, S., Wack, N.: ISMIR 2004 Audio Description Contest. Technical Report. Music Technology Group, Bracelona

[7] Castillo, J.R., Flores, M.J.: Web-based music genre classification for timeline song visualization and analysis. IEEE Access 9, 18801–18816

[8] Chaki, J.: Pattern analysis based acoustic signal processing: a survey of the state-of-art. Int. J. Speech Technol.

[9] Chan, W.C., Liang, P.H., Shih, Y.P., Yang, U.C., Chang Lin, W., Hsu, C.N.: Learning to predict expression efficacy of vectors in recombinant protein production. BMC Bioinform. 11(1), 1–12.

[10] http://www.cp.jku.at/datasets/LFM-1b/

[11] https://arxiv.org/pdf/1912.11564.pdf

[12] Divya Sardana "RecommenderSystems_PyData"

[13] https://towardsdatascience.com/how-to-build-a-simple-song-recommender-296fcbc8c85 "RecommenderSystems PyData",2016

[14] Kavin Kumar.V, Rachamalla Rahul Reddy, Rohit Balasubramanian, Sridhar.M, Sridharan.K Dr.D.Venkataraman," A Hybrid Approach for Recommendation System with Added Feedback Component"

[15] G. Gonzalez and M. Miquel. (2017). Embedding Emotional Context in Recommendation System. 20th International Florida Artifical Intelligence Research Society ConferenceFLAIRS.

[16] Nunes, M.A.S.N. (2008). Recommender system based on personality traits