**1 Overview**

I was inspired to do my AI project on a problem that I face very frequently. As an avid music lover, I find myself listening to music daily, whether it be in the car or while I am taking a shower or doing homework. The problem that I am faced with is that Apple Music only suggests songs to me that are based on the genre of the song I am listening to. However, I prefer to listen to music based on the mood that I am currently in. I decided that I would make my AI project into something that could help this problem. I have created an AI that will suggest three songs of the same mood that the user input.

**2 Methodology**

To create my project, I decided that I would use Python, as it is the programming language that I am most familiar with to write artificial intelligence code. The modules that I imported are pandas, random, sklearn, tkinter, numpy, and matploblib. I used pandas to import the CSV file. I used random to help me shuffle the list of songs so that it wouldn’t output the same songs in alphabetical order. I used sklearn to create a label encoder and create the nearest neighbors model. I used tkinter to build my GUI. I used numpy to help me reshape and concatenate my data. Finally, I used matploblib to create the graph of my K Nearest Neighbors model.

**2.1 Collecting Data**

I knew I had to find a database of songs. I tried searching for databases of songs based on mood, but I did not find anything. I also realized that whatever I would have found would not be accurate because songs can convey different moods to different individuals, so I decided to make one of my own. I went through all my songs on my Apple Music and put them into eight different playlists that include these moods: Angry, Breakup, Happy, Lonely, Love, Party, Sad, and Vibey. Apple Music has an option to “Export Playlist” to a text file. So, I did that, and then I used openAI to help me format this data into a CSV file. The CSV file has 1526 columns and 6 rows. The rows include Title, Artist, Genre, Mood, Number of each Genre, and Number of each Mood.

**2.2 Performing Experiments**

To perform experiments after I wrote the Label Encoding code, I used the code below to help test and see if this code worked.

# Print the encoded data

for i in range (len(titleAndArtist)):

print("Title:", titleAndArtist[i][0])

print("Artist:", titleAndArtist[i][1])

print("Genre:", genreAndMood[i][0])

print("Mood:", genreAndMood[i][1])

This code worked and it showed me that each column’s label was encoded to a number in alphabetical order. There is a commented section in my code after the references in label encoding that describes what number each mood and genre label got encoded to.

**2.3 Testing the System**

When testing the overall functionality of the code, I looked at the output in the GUI text boxes and compared them to my original CSV file to make sure that the songs that it output were in the specific mood that I inputted. Overall, it was an easy way to test.

**3 What I Did and Didn’t Do**

While creating the label encoding section of my code, the majority of this code came from the project that I submitted for Assignment #7 (Plotting and classification). It came specifically from the classRoster.py file. However, in the assignment I submitted, I used .loc and .tolist() instead of .iloc. The difference between these two functions is that .loc locates data by labels, while .iloc locates data based on an integer index. Something that I did not do in that assignment was combine two separate arrays into one 2D array. This code came from a source online. Their code looked like:

>>> a **=** np**.**array**((1,2,3))**

>>> b **=** np**.**array**((2,3,4))**

>>> np**.**column\_stack**((**a**,**b**))**

*array([[1, 2],*

*[2, 3],*

*[3, 4]])*

This is the code I wrote that combined the separate arrays for genre and mood:

genreAndMood = np.column\_stack((genre, mood))

When creating the recommend section that created the K Nearest Neighbor Clusters, most of this code came from the in-class Lesson 13, k\_nearest\_neighbors.py file. This helped me create the model for the k nearest neighbor algorithm. However, I did have to tweak this a little bit. Since it was outputting songs that were not in the same user selected mood, I had to write code that helped me only recommend songs with the same mood.

I did this by first creating an empty list called sameMoodAndGenre. I used a for loop to loop through all the indices from the k nearest neighbor model to check and see which indices matched the user’s mood and genre input as the original genre and mood array. If it matched up, then those indices would be placed into the list that I created called sameMoodAndGenre that is used to hold the indices of the songs that had the same mood and genre as the user’s input. After this, I needed to figure out how many songs were left to generate the same mood, so I subtracted the number of clusters from the length of the sameMoodandGenre list. If this proved to be greater than 0, then I had to go through the songs to find ones that had the same mood, but different genres. To do this, I had to iterate through the songs to find the ones with the same mood, but different genres. I needed to check and make sure the indices from sameMoodAndGenre were not included so I used the code below from an online source to figure out how to check if an item was not in a list.

lis = ['t','u','t','o','r','i','a','l']

if 'p' not in lis:

print ("List is not having element with value p")

else :

print ("List is having element with value p")

This is the code I wrote: if moodGenre[1] == userMood[0][0] and i not in sameMoodAndGenre: I would have been unable to write this block of code if it wasn’t for the basic tutorials and assignments in Python that were given in the beginning of this course.

When creating the GUI code, some old code I had from Mrs. Etheredge’s CS101 class helped me. However, I did have to use some online sources to help me understand how to create a spin box, list box, and scroll bar. They also helped me learn how to get the user’s input. Most of this code came from geeksfromgeeks. It was an amazing source that aided in helping me learn how to do these things. Finally, I needed to learn how to use sticky to get the objects in my GUI to be aligned correctly.

**4 Final Results**

When beginning the code, I tried using the kmeans algorithm, but it did not give the output I desired. However, after completing the code with the k nearest neighbor algorithm, I accomplished having an AI that will always output a song from my CSV file based on the mood that the user inputs. My original idea was to have the program output 3 songs of the same mood AND genre from the user input. If there weren't 3 songs based on the user selected mood and genre then it would just output 3 songs of the same mood. I tried my best to write a block of code that would do this; however, it did not work out as planned. The obstacles I faced were that every time I output the songs; they were in alphabetical order. Since they were in alphabetical order, only the same 3 songs were being output every time and started with the letter A. Obviously that is not what I was trying to do, so I had to shuffle the list of songs randomly. I think this is what hindered my block of code that tried to output the songs with the same mood and genre. The code does have the potential to output songs of the same user selected mood and genre, but it is random, so it will not do this every time. After doing this project, I learned that creating AI code is not as easy as I thought it would be. I learned that it takes time to do a project to this extent and if I were to have waited to the last minute to do this code that I would have never gotten it halfway done. In the future, maybe over the summer, I want to keep working on this code and maybe use a different approach to achieve the output that I desired originally. I feel as though I could use the label encoder to help encode the labels to numbers, but the rest I think could possibly be achieved without machine learning models.

Overall, I feel like my code is well documented and structured. It is split into three different parts, Label encoding, Recommend Function & Nearest Neighbor Clustering, and GUI. Each part has a comment that marks the beginning and end of each. At the end of each part, I put references of the websites that I used for each part and those sources have been included in this paper. After label encoding, there is even a commented section that tells what number each individual mood and genre gets encoded to, and how many of each song and genre there is in my CSV file. One would be able to see this if you were to look at my code. It is unfortunate that it did not end up printing the way I originally wanted it to. However, it does output songs to the user based on the mood the user selected so I think that a win is a win.

**5 References**

This is how I learned to stack two arrays into one:

<https://numpy.org/doc/stable/reference/generated/numpy.column_stack.html>

This is how I learned to reshape the user input data into a 2D array

<https://saturncloud.io/blog/understanding-the-differences-between-numpy-reshape1-1-and-reshape1-1/>

This is how I learned to concatenate data to combine two inputs into one 2D array:

<https://www.geeksforgeeks.org/numpy-concatenate-function-python/>

This is how I learned to check if data was not in a list:

<https://www.tutorialspoint.com/list-methods-in-python-in-not-in-len-min-max>

This is how I learned to create a spin box, list box, and scroll bar:

<https://www.geeksforgeeks.org/python-tkinter-spinbox/?ref=lbp> (If you were to click on this link, there is a scroll bar on the left that gives tutorials on how to create each of these.)

This website also aided in assistance for the scroll bar: <https://www.tutorialspoint.com/python/tk_scrollbar.htm>.

These websites helped me learn how to use sticky to get the objects in my GUI aligned correctly:

<https://www.pythonguis.com/tutorials/create-ui-with-tkinter-grid-layout-manager/#:~:text=columnspan%20%2C%20rowspan%20%2D%2D%20specifies%20how,spacing%20right%20for%20your%20widgets.&text=sticky%20%2D%2D%20specifies%20a%20value%20of,NE%20%2C%20SW%20%2C%20or%20SE%20> and <https://www.oreilly.com/library/view/python-in-a/0596001886/re701.html>