MINI REPORT

Aim:

The objective of this assignment is to design a recommendation system engine for online music platform using the provided similarity metrics.

Software/libraries used:

Python, Csv library, Numpy library

Python: is a general purpose programming language and one of the most popular language used in data science and machine learning.

Csv: it is a python standard library for working with comma separated values (csv) file.

Numpy: is a python used for working with arrays, it also offers functions for working with linear algebra and matrices

A brief Introduction and exploratory analysis of the dataset.

The provided dataset has 169,909 entries and 19 features. Twelves features were retrieved from the dataset for the analysis as required in the assignment statement. These features are;

Numerical features which include the accoustiness, danceability, energy, liveness, loudness, popularity, speechness, tempo, and valence; Non numerical features like the music name, artist name, and music id.

The music or artist id is unique for each of the entry of the dataset.

Analysis of the problem.

The main objective of this assignment is to design a function that takes two music or artist ids and a similarity metric function, then compute relationship between the two ids using their respective numerical features.

How do we go about this?

I created two python module, load_dataset_module.py and similarity_module.py.

load_dataset_module.py

There three functions defined inside this python module; the load_dataset function, artist_music function and music_features function.

load_dataset function: This function loads the dataset.csv using python standard csv library and stores it in a dictionary. The dataset features and their corresponding values being used as keys-values pair in the dictionary. It then returns a dictionary format of the dataset.

artist_music function: this function takes the dictionary object returned by load_dataset() and returns a dictionary called artist_music_dict dictionary that contains the following features and their corresponding values; the artist name, music name, id, accoustiness, danceability, energy, liveness, loudness, popularity, speechness, tempo, and valence.

music_features function: This function performs the same operation as artist_music() except for the exclusion of music name and artist name in the dictionary it returns.

similarity_module.py.

There are eight functions implemented in this python module; the helper_function, the euclidean_similiarity function, cosine_similarity function, pearson_similarity function, the jaccard_similarity function, the manhattan_similarity function, the comupte_similarity function, and the main function.

Prior to defining all the functions above, artist_music() and music_features() was imported from load_dataset_module.py. Also, numpy was imported for easy mathematical vectorization computation of the arrays. I leverage on numpy inbuilt function for easy calculation of all the metric functions.

(a.) *helper_func function*: This function accepts a dictionary and helps modified it such that the feature "**id"** which is unique for each entry of the dataset is now set as the keys of the dictionary while the corresponding features are now values of the dictionary.

It returns a dictionary where each the "**id**" is the key and respective features set as value.

(b.) *euclidean_similarity function*: This function accepts a data dictionary returned by the *helper_fun()* and two ids, then compute Euclidean similarity between the two ids using their respective numerical features.

It returns the result of the mathematical computation.

```
euc_sim = np.linalg.norm(id1, id2)

return euc_sim
```

np.linalg.norm() compute Euclidean distance for us out of box.

(c.) *cosine_similarity function*: This function accepts a data dictionary returned by the *helper_fun()* and two ids, then compute Cosine similarity between the two ids using their respective numerical features.

It returns the result of the mathematical computation.

```
cos_sim = np.dot(id1, id2) / (np.linalg.norm(id1) * np.linalg.norm(id2))

return cos_sim
```

(d.) *pearson_similarity function*: This function accepts a data dictionary returned by the *helper_fun()* and two ids, then compute Pearson similarity between the two ids using their respective numerical features.

It returns the result of the mathematical computation.

```
ps_sim = np.corrcoef(id1, id2)[0, 1]   #indexing value from on the first row of the second column

return ps_sim
```

(e.) *jaccard_similarity function*: This function accepts a data dictionary returned by the *helper_fun()* and two ids, then compute Jaccard similarity between the two ids using their respective numerical features.

It returns the result of the mathematical computation.

```
intersection = len(list(set(id1).inteersection(id2)))

union = len(id(id1) +  len(id2)) – intersection

return float(intersection) / uion
```

(f.) *manhattan_similarity function*: This function accepts a data dictionary returned by the *helper_fun()* and two ids, then compute Manhattan similarity between the two ids using their respective numerical features.

It returns the result of the mathematical computation.

```
mah_sim = np.abs(id1 – id2).sum()

return mah_sim
```

(f.) *compute_similarity function*: This function accepts three parameters; two ids and a similarity metric function.

It then carries out mathematical computation using the numerical values of each id stored in a numpy array by applying the similarity metric function passed. It the return the out of the calaculation.

```
result = similarity_func(music_feature_dic, id1, id2)

return result
```

(f.) *main function*: This functions runs all the codes.

Inside this function, two ids e.g. id1, id2 accepted from the user, follow by the similarity metric choice of the user, it then calls the necessary functions on these inputs and print/display the result for the user.

Conclusion:

My codes follow the methodology of good software engineering practices with readable comments, easy to read and give the desired output.

All the numpy in-built functions used are gotten from the numpy documentation online.