|  |
| --- |
|  |
| The music guru+ |
| an enhanced collaborative/content-based recommendation system! |

|  |
| --- |
| Raveendrababu Pasumarthi  1-21-2021 |

1.0 Introduction

*The music guru+* is an enhanced collaborative/content-based recommendation version of *The music guru* that was originally developed as part of assignment 1 for a company in order to improve their sales. The Company was initially looking for an online intelligent recommendation system and *The music guru* could succeed to some extent of that initial initiative. The(this) enhanced version incorporates new requirements (like top n recommendations) and revamps the existing version in highly modular way by using Object Oriented Programming and advanced Python libraries like numpy, pandas, scipy, scikitlearn and matplotlib. This document briefly discusses about the new requirements understanding, analysis, high level and low-level technical designs, coding & implementation and the limitations of the new version.

The key implementations of the *The music guru+* are:

1. Implementing modules in assignment 1 using Object Oriented Concepts
2. First n most similar artists to the target artist using similarity scores.
3. First n most similar music to the target music using similarity scores.
4. First n most similar music to the target artist using similarity scores.
5. Evaluation of the accuracy of similarity metrics and plotting of the recommendation accuracies of similarity metrics graphs.

The key considerations are.

1. Use OOP, a nice to have interface and fix any existing issues in assignment I part.
2. Use similarity metrics for top n recommendations.
3. A) euclidean B) cosine C) jaccard D) manhattan(cityblock) and E) pearson (correlation)
4. Use distance.cdist from (scipy.spatial library) to generate similarity scores.
5. Use nearest neighbors’ algorithm and accuracy\_score to evaluate and accurately score the metrics.
6. Use top n as top 20 recommendations and visualize all the results in matplotlib graphs.

2.0 Requirements Analysis

The new requirements implementations are analyzed with the appropriate use cases below.



1. High Level Design

3.1 Design considerations:

* Mean of all the observations for each feature was used to calculate the score for the OOP version of assignment I, this is an improvement from previous version in which the very first observation was used.
* Part 2 (Top n recommendations) onwards, completely scikit learn modules were used instead of traditional methods.
* In case of artists, system considers only single artist observations if target artist is matched, otherwise multiple artists observations but not both.

3.2 Class diagram:

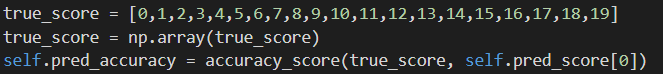


1. Low Level Design/Implementation
   1. Top 20 Artists to a target artist implementation:

* Target artist can have multiple songs.
* Each of those songs will be calculated score against each song of non-target artist.
* Deduped the same score results; Sort the scores and dedup the results by repeated artist with same score; publish top 20 results.
* Numpy transpose and pandas melt feature were used to obtain the scores of multiple songs of an artist’s score, this is effective method to find the artist’s distance.
* module : similarity\_module\_II\_v5;calss:top\_n\_recom;method:top\_n\_artists\_distance()

4.2 Accuracy of metrics calculation:

* Top 20 results and the target from each similarity metrics had been passed to the nearest neighbor’s (NN) algorithm.
* The true score (indices of result from the metric) and predicted score (indices of result from the NN) are passed to the accuracy score.



* All the metrics are matching exactly except the Jaccard distance.

4.3 Best similarity metric:

* Experimental results suggest cosine similarity is the best results retrieving similarity (selecting particular region song and looked any regional songs and cosine retrieved more similar results than any other metric)
* However, from various research papers, its been noticed the Manhattan similarity and cosine similarity are the best choices for music recommendation systems

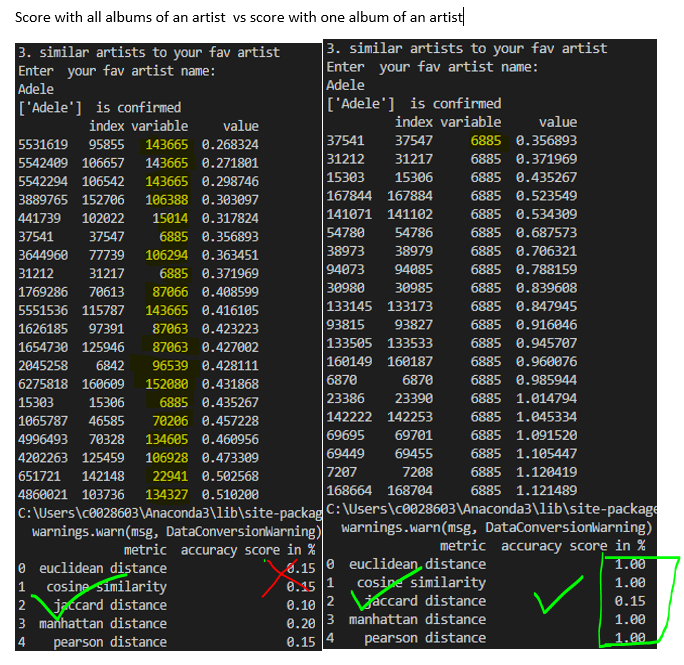
4.0 Testing & Limitations

**Testing:**

* If an artist name is a substring in another artist, then there is chance of another artist result will be displayed (e.g., “Ramchandra Pal” will be displayed for “Ramchandra”) – **Resolved.**
  + This was an issue in the assignment I solution.
* Repetition of same artist in top 20 results – **Resolved**.
  + Used the pandas melt feature to flatten.
  + Deduped the same scores of resulted artists.
  + Sorted the score and captured the top 20 unique artists.
* Artist name n/a(n,a) is removed for user search database
* X,-,A,Z,D,O,M,V,K these are all valid artist names and user can search by that name

**Limitations:**

* Nearest neighbor’s algorithm (used for comparing the chosen algorithm) results in poor accuracy for the top n artists of a target. The chosen algorithm works better and explained in the below image. (zoom the document for better view of image).



* Multiple(duplicate) songs of same name (e.g., there are 17 songs with title **Hello**)
  + The very first result of hello will be the target album.
  + There is a chance of wrong hello for the user. In this case user must choose target album of an artist option
* Multiple songs(duplicate) of **same name of same artist** (e.g., **there are two Hello songs from artist Adele**)
  + The very first result of hello will be the target album.
* Non-English Text on matplotlib graph was not encoded properly.