**Music Recommendation System Draft**

**Step 1: Choose the most similar song to user interaction history**

* A screenshot of a computer screen

  Description automatically generatedCalculate a target value based on user interaction history. Then, select the most similar song based on the target value.

**Figure 1: Top 5 Similarity based on user preferences**

**Similarity method**: **Manhattan Distance** convert to % difference.

**Artist**: 1 if the same as the target value, otherwise 0 **(10% weight)**

**Genres**: 1 if the same as the target value, otherwise 0 **(10% weight)**

**Decades**: Similarity in the range of 50 years **(10% weight)**

**Formula: MAX(1 – DIF / 50, 0)**

**Acoustics**: Similarity in the range of 0.0 to 1.0 **(5% weight)**

**Formula: 1 – DIF**

**Valence**: Similarity in the range of 0.0 to 1.0 **(15% weight)**

**Formula: 1 – DIF**

**Energy**: Similarity in the range of 0.0 to 1.0 **(15% weight)**

**Formula: 1 – DIF**

**Danceability**: Similarity in the range of 0.0 to 1.0 **(15% weight)**

**Formula: 1 – DIF**

**Speechiness**: Similarity in the range of 0.0 to 0.5 **(5% weight)**

**Formula: 1 - DIF / 0.5**

**Tempo**: Similarity in the range of 0.0 to 250.0 **(15% weight)**

**Formula: 1 – DIF / 250**

**DIF** is the absolute value of subtraction between target and track values.’

* Now the most similar track will be chosen as the “**center**” track of the playlist.

**Step 2: Find the nearest neighbors of the chosen track based on playlist types**

The difference between each type of playlist is the level of exploration. The more exploration the playlist is, the more likely a new track with a different genre but the same audio features appear.

**Type 1: Discovery Mix (High Exploration)**

* Compare **valence, energy, danceability**

**Type 2: Exploration Mix (Medium Exploration)**

* Compare **valence, energy, danceability, and tempo**. Only selected tracks with **75 or higher popularity**

**Type 3: Personalized Hit Mixes (No Exploration)**

* Compare **acousticness, valence, energy, danceability, speechiness, and tempo**. Only selected tracks with **75 or higher popularity and the same genre**

**Similarity Method: Euclidean Distance**

**Type 1 Example:**

**Chosen Track: “**Save Your Tears“ in **Figure 1**.

A graph with numbers and points

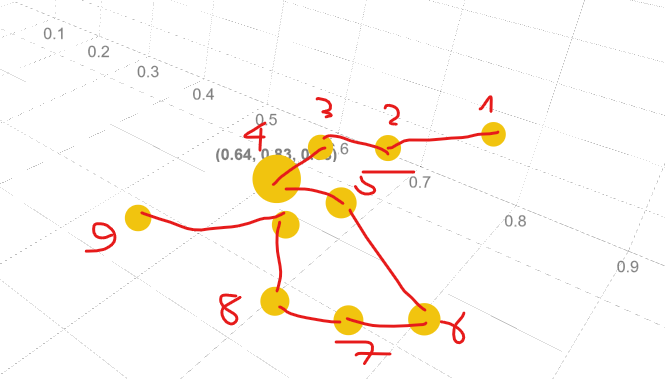
Description automatically generated

**Figure 2: Top 10 similar tracks to the chosen track on 3D plot**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Song** | **Valence** | **Energy** | **Danceability** | **Euclidean Distance** |
| Save Your Tears (Chosen) | 0.64 | 0.83 | 0.68 | 0 |
| Rap God | 0.63 | 0.84 | 0.71 | 0.0331 |
| Just A Little Step | 0.67 | 0.85 | 0.69 | 0.0374 |
| So Much Better | 0.62 | 0.86 | 0.72 | 0.0538 |
| Love Letter | 0.69 | 0.86 | 0.66 | 0.0616 |
| Nothing But Trouble | 0.7 | 0.81 | 0.68 | 0.0632 |
| Don’t Tread On Me | 0.68 | 0.88 | 0.66 | 0.067 |
| Killing Boys | 0.68 | 0.9 | 0.69 | 0.0812 |
| Berzerk | 0.68 | 0.87 | 0.74 | 0.0824 |
| Unity | 0.57 | 0.88 | 0.71 | 0.0911 |

**Step 3: Reorder these tracks to create a smooth transition playlist**

Making a smooth-transition playlist is curating a playlist that transitions smoothly from one song to the next based on the similarity between consecutive songs. The smoothest transition playlist will be the playlist with the lowest total distance between two consecutive songs in the playlist. For example, to make a smooth transition playlist from **Figure 2**, the playlist will have the order like this:



**Figure 3: Example of a smooth-transition playlist**

* Before having an in-depth explanation about algorithms can be used to solve this problem. I would like to introduce a well-known problem that is highly relevant to the smooth-transition problem:

**Travelling salesman problem (TSP)**

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

* The only difference between the TSP problem and creating a smooth-transition playlist here is that there is no need to return to the first (origin) song. Therefore, we just need to remove the longest distance between two songs in the playlist and select either side as the first song.
* As this is a combinatorial optimization problem, we could use either greedy or optimization algorithms & methods.
* **Algorithms & Methods:**
  + **Nearest Neighbour (NN) algorithm:**
    - The nearest neighbor algorithm is a greedy algorithm created to solve this problem with a good enough answer.
    - **Advantage**: This algorithm is very easy to implement and can be executed quickly. **Worst-case Time Complexity**: **O(n2)**
    - **Disadvantage:** Due to its “greedy” approach, this algorithm is not a suitable solution as it frequently produces an answer in which the total distance is much higher than the lowest possible distance.
  + **Dynamic Programming:**
    - Dynamic Programming is a mathematical optimization method that aims to produce globally optimized answers.
    - Dynamic Programming is an optimization of a plain recursion. The main difference is that while a recursive solution has to repeatedly call for the same inputs, dynamic programming stores the results of subproblems to solve the main problem so that it doesn’t need to compute again. In other words, it is recursion with “memoization”. **A clear example of this difference is the Fibonacci Numbers problem: (Appendices)**
      * **Normal Recursion:** returns f(n – 1) + f(n – 2).
* When f(n – 1) is called, it calculates f(n – 2) again.
  + - * **Dynamic Programming:** dp[n] = dp[n – 1] + dp[n – 2] with a loop iterates from 1 to n. During the loop, dp[n – 1] and dp[n – 2] value has been calculated and stored before.
    - **Advantage:** This method produces the exact solution.
    - **Disadvantage:** This algorithm is computationally expensive, as its **Time Complexity is O(n22n).**
* **Not suitable for a playlist consisting of more than 20 songs.**
  + **Genetic Algorithm:**
    - Genetic Algorithm is a metaheuristic inspired by the process of natural selection. A metaheuristic is a higher-level procedure designed to provide a sufficiently good solution to an optimization problem.
    - **Advantage**: This algorithm is not computationally expensive since the overall **Time Complexity** is **O(Generations \* (N \* Population + N2)**
* **Faster than dynamic programming, but slower than Nearest Neighbor Algorithm.** 
  + - **Disadvantage**: This algorithm is not guaranteed an exact solution
* **Provides better solutions than the Nearest Neighbor Algorithm, but doesn’t guarantee the correct answer as Dynamic Programming**.
* I have chosen to implement a **Genetic Algorithm** for the smooth-transition playlist problem since it provides a good solution that is suitable for the smooth-transition problem since minor differences are not humanly noticeable.