# CMPT 310 Final Project Write-up
# Music Recommendation System

**Authors:**
**Cyrus Sandhu, Ramtin Rezaei, Jaskamal Chouhan, Varsha Adusumalli**

**System Explanation**

For our project we built a music recommendation system that gives users more accurate recommendations than the current systems offered by streaming services such as Spotify. Our recommendation system takes in a user input of one or more songs and provides recommendations of similar songs based on quantitative audio features.

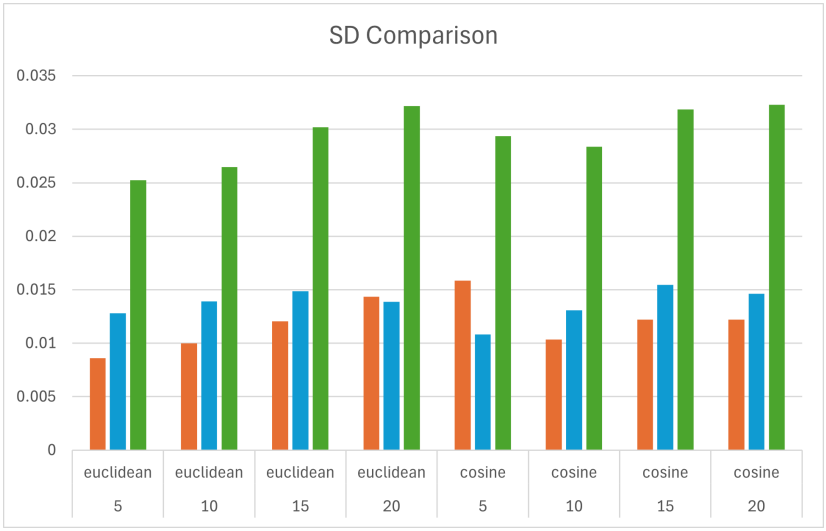
We implement a K-Nearest Neighbors classifier to make recommendations. We did so using the Nearest Neighbors model in the Python sci-kit learn library. The model was trained using a data set featuring over 170,000 songs and their audio features. During the development process of our model, our data pipeline consisted of the following steps.

**Data Pipeline**

- Data Collection
    - Our original intent was to use the Spotify API but due to changes to the accessible endpoints, we had to pivot to using a third party data set instead
- Feature Engineering
    - This is the step where we performed our data cleaning where we incorporated techniques such as normalizing features values via standard scaling and splitting data into training, testing and validation sets.
- Model Training
    - This is where we using our training data set to train a KNN model to make recommendations based off of user inputted songs
- Evaluation
    - We then compared models that featured different hyperparameters to determine the model that was best at making recommendations.
- Deployment
    - Creating a user interface to facilitate the input and out process

**Training Results**

We were able to get the following results during the evaluation state where we compared the effects that different hyperparameters had on the model. Below are plots showing the differences in mean and standard deviation.


From analyzing these results we were able to determine that the optimal model is achieved when we set K=5 and use a euclidean distance metric.

**System Limitations**

The main limitation of the system has to do with the size of the data set we were able to use. When predicting songs we usually are able to get the top recommendation to have a similarity score of around 70% however there are certain cases where this score drops considerably. This is due to the fact that our system is limited by the data set. Since we were not able to access the spotify API due to recent changes made by Spotify. Our entire data set that we had to further split was only 170,000 songs which is less than 1% of the total available songs in the Spotify catalog. With a data set of this size we wanted to make sure our model did not overfit the training data. Which is where the use of cross validation came into play.

**Feature Table**

| Description | Platform | Completeness | Code | Author(s) | Notes |
|---|---|---|---|---|---|
| **Data Preprocessing** | Local | 5 | Python | Cyrus, Ramtin, Jaskamal, Varsha | We cleaned the dataset and ensured that any duplicate songs would be removed. |
| **KNN Model** | Local | 4 | Python | Cyrus, Ramtin | Trained the KNN model by using audio features. Could be improved with a genre feature. |
| **Evaluation Methods** | Local | 5 | Python | Ramtin, Jaskamal | Implemented 5-fold cross validation on different hyperparameters. Also, we tested the accuracy of the model using specific cases and analyzed results based on expectations. |
| **Similarity Based Music Recommendation** | Local | 4 | Python | Cyrus, Ramtin | Uses the KNN Model to create recommendations. Can be improved by using more advanced distance metrics. |
| **GUI Interface** | Local | 4 | Python | Varsha, Ramtin | Works well generally, but improvements could be made in handling the inputs. |
| **Recommendation Logic** | Local | 4 | Python | Cyrus, Jaskamal, Ramtin, Varsha | Designed strategies and logic for the KNN model and feature weighting. This could be improved by using more advanced techniques. |

**External Tools & Libraries:**

- Machine Learning & Data Processing
    - Scikit-Learn – Used for StandardScalar ( feature normalization), Nearest Neighbors (KNN model), train_test_split, and KFold during evaluation.
    - Pandas – Core data manipulation
    - NumPy – Numerical operations and array handling throughout the model and evaluation code.
- GUI Development
    - PyQt6 – The entire desktop user interface is built with PyQt6 ( widgets, layouts, styling via Qt style sheets).
- Dataset

    - This project uses the publicly available Spotify Million Song Dataset ( on Kaggle as “ Spotify Tracks Dataset”. The file data.csv placed in the code file contains ~ 170k tracks with audio features (like danceability, energy, tempo, valence, etc.).
- Platform Dependencies & Installation
    - The application runs on Python and has been tested on Windows 11 and macOS. Required packages installed via “ pip install …package_name…”
    - The dataset file data.csv must be placed in the directory related to the project. No external API keys are needed at runtime.