Project Deliverable 2

1. Problem statement:

The purpose of this project is to build a Spotify playlist generator model which, given a user's playlist, will generate a new playlist of 50 new songs that should match the user's tastes.

2. Data Preprocessing:

As stated in the first deliverable, we created our own dataset. The Spotify API *spotipy* provides various tools to extract the data from URLs of various playlists and songs. We first created an excel spreadsheet where we put URLs of two playlists for each of Spotify's musical genres (see Spotify_Genres_list_final.xlsx). From there, we've extracted information for each track using methods from the module and wrote the whole data set to an .csv file. Each track in Spotify has an ID and each of these gives access to audio features that we need for our dataset (see MAIS202data.csv). There are a total of 13 features per song. The total size of our data set is currently around 3,800 songs. This suffices to test and train our model, but we will consider enlarging it to a total of 10,000 songs in order to have more options for our recommendations and to have larger clusters. That way, we decrease the chances of having tiny clusters which may cause problems in our algorithm. We hope that enlarging our dataset will also allow the model to make better recommendations. Although, we must take into account the time and space complexity of our algorithms and adjust the size of our dataset later.

Note that some songs are contained in multiple playlists. While preprocessing the data, we deleted the duplicates. While creating the playlist, we also made sure to extract an equal amount of songs from each genre so that our model is not biased towards one genre when making recommendations. We aimed to have around 150 songs per genre for now, but it was difficult to find that many songs for a couple of genres, especially because some genres overlap and have duplicates. We also wanted to have songs that were currently trending or are new so that we can make recommendations that are up to date, but this approach has some limitations for some genres (ex. Classical music or songs from the 80s for instance). We will make sure to keep this in mind when building the larger dataset.

3. Machine learning model:

a. **Specify the framework and tools that you used to implement your model.**

We haven't changed the choice of our model. We implement the k-means algorithm to create clusters of similar songs. We then implemented the K-Nearest_neighbours algorithm to make our playlist predictions. In particular, for every song in the user's playlist, we find which cluster it belongs to, and then we recommend its nearest neighbor within that cluster.
We used several libraries to help us decide on the number of clusters, the model and visualization. We used several modules to visualize and process our data such as pandas, matplotlib.pyplot, seaborn and plotly.graph_objs.

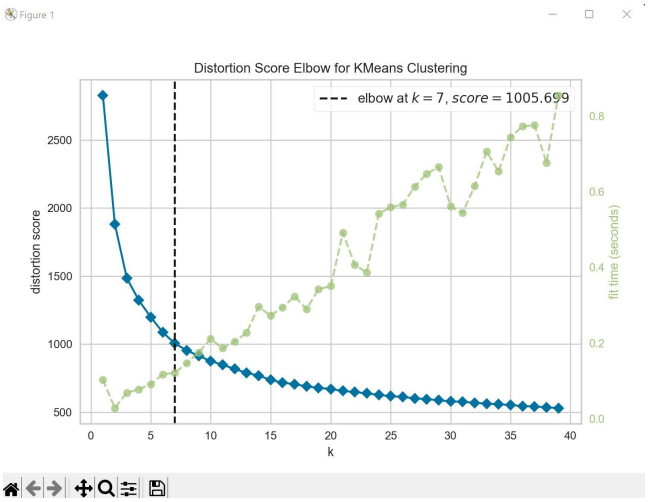
The main libraries we used to build our framework are the following:

- KMeans from sklearn.cluster: k-means learning model, NearestNeighbours model
- PCA from sklearn.decomposition: reduce the dimension of our data for visualisation of clusters
- KElbowVisualiser from yellowbrick.cluster: decide the number of clusters

Then, we create two data frames. One for the main data and one for the songs of the user's playlist. We merge them together and normalize the whole data frame. Afterwards, we execute the KMeans model on it. To provide recommendations, we run the KNN model on it to obtain the songs contained in each cluster that are the nearest neighbors to each song and we output songs based on the ratio of the number of user's songs in that cluster.

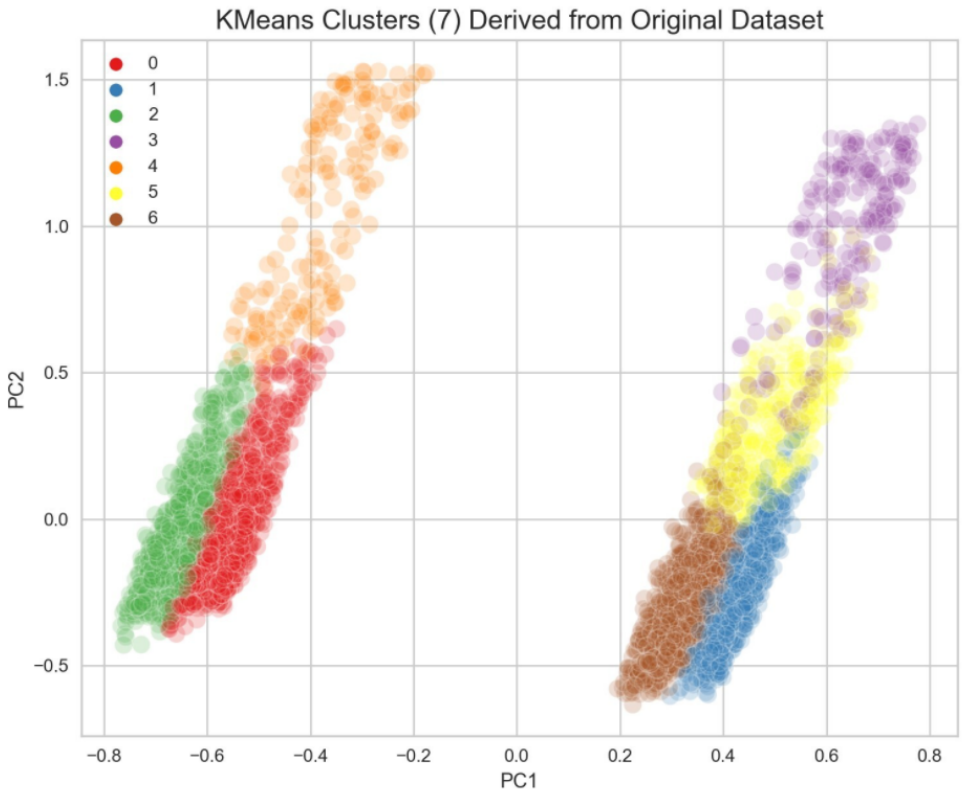
b. **Justify any decision about training/validation/test splits, regularisation techniques, optimization tricks, setting hyper-parameters, etc.**

The first parameter we have to find is the number of clusters. We do this using the Elbow Method. Python yellowbrick.cluster provides a class KElbowVisualizer that does this task by checking a good fit for the number of clusters.



We have found that 7 clusters was the ideal number using the Elbow method, but this number may change as we incorporate the large dataset

After training our k-means model, our second task is to reduce the dimension of our data set because we want to visualize the clusters, but we have 13 features. We use the PCA algorithm from sklearn.decomposition. The following is the visualization of the 7 clusters formed after training the model on the dataset.



We have also normalize our dataset to ensure that our features will all contribute to making the clusters and selecting the closest neighbors.

We will be working on optimizing the model once we are confident that it correctly outputs 50 songs for any input playlist.

c. **Description of validation methods How did you test your model? Is your model overfitting or underfitting?**

So far we just tested our model by creating playlists and taking a quick look at them to make sure our model is doing what it is supposed to be doing. That is, we wanted to make sure that at least a large portion of the recommendations at least came from the same genre/artist/type of songs in the input playlist. We will be using more sophisticated methods to thoroughly test our model soon.

We don't think that our model is overfitting as with our test classical music playlist, most of the outputs are from the classical genre, but some are not. For instance, one of the songs it output was a Kanye West song, which is quite far from the genre of the input playlist.

d. **Did you face any challenges implementing the model?**

One big challenge was that when we tested our model, the song recommendations were not very good. Only the first few recommendations reflected the taste of the user but the rest did not. For example, we tested our model on a premade Spotify playlist containing 50 classical songs, but the recommendations barely contained classical pieces. After consulting a TPM, we learnt that we should normalize our dataset so that the model takes into account all of the features when creating the clusters and finding the nearest neighbors rather than having a feature that has very high values which mostly influences the recommendations. This change was crucial to making our algorithm better as our playlists reflected a lot more what we were going for. Using the same classical playlist, we now had most of the recommendations come from classical pieces.

Our current challenge now is to make sure that our model works for any size playlist and that it outputs 50 songs every time.

4. Preliminary results:

In Deliverable 1 we proposed to evaluate the performance of our model based on the reviews from other people. Since we haven't finalized our web app, this step is yet to be implemented. However, our model gives decent predictions and runs in a relatively short time frame which is around 30 seconds. It is worth noting that since various songs from different genres fall under the same range of certain features, a cluster might contain two songs from completely different genres, for instance a rap song in a classical playlist. As stated above, after the normalization of the dataset, our predictions were much more accurate and the ML part of the project is practically finished. The results of our predictions can be viewed in the links provided above.

5. Next Steps:

We will fine tune our model and test different parameters now that we have a reliable model.

Then we will be working on integrating the model into a webapp.