Project Deliverable 3

1. Final Training Results

We have implemented the following changes after receiving feedback from deliverable 2:

1. Following a discussion with a TPM, we changed our KMeans model to only train on the dataset songs rather than the dataset with the user's songs. We felt that this made more sense logically because the user's songs act as "testing" data and should be the ones whose cluster is predicted. This significantly decreased the runtime of the whole song generation algorithm since now we don't have to train the model and get the clusters everytime we want to generate a new playlist. Instead we can save the models since the data they use to be trained is always the same dataset. We think that this will make the user experience more enjoyable once we integrate it in the webapp. In fact we managed to decrease the total running time of the algorithm from 28.290 to 0.628 seconds!

2. We also found a small bug in our code where we realized that we were accidentally not taking all of the song features into account when creating the models. Now that this has been fixed, we expect our recommendations to be even better but more testing needs to be done to examine the effect. However, when we tried out a couple different playlists, this actually had almost no impact on the predictions.

Overall, these two fixes did not have a huge effect on our results because the change to the KMeans model only meant that we were removing about 50 songs from the training data, where the training data had around 3800 songs (see figures below for comparison).

In addition, as mentioned in Deliverable 2, we wanted to expand our dataset from 3800 songs to 10 000 songs so that there are a greater variety of generated recommendations. Unfortunately, we did not have time to increase the size of our dataset this week, but that will be our next step.

Finally, we computed the accuracy of our model by taking the mean squared error between the attributes of our songs in the user's playlist and the recommendations. We begin by normalizing our data. The idea was to compute how far we are from an average input feature in our predictions. We used the average of each feature in the user's input to generate the "y_true" values and compared them to the features of the

output songs, or the “y_pred” set. Here is an example of feature data for the classical playlist, and the predictions for old and new models.

Old Model:

	Features	Y_true Mean	Y_pred Mean	Y_true Norm Mean	Y_pred Norm Mean	MSE
0	danceability	0.302178	0.303498	0.127791	0.129181	0.003314
1	energy	0.213662	0.186703	0.119724	0.115753	0.006617
2	key	4.720000	4.700000	0.117560	0.114634	0.006868
3	loudness	-16.757540	-18.160840	-0.136499	-0.135349	0.001682
4	mode	0.660000	0.660000	0.114891	0.114891	0.006800
5	speechiness	0.040494	0.040186	0.139140	0.139984	0.000405
6	acousticness	0.927420	0.918220	0.140837	0.140689	0.000206
7	instrumentalness	0.729223	0.723944	0.131707	0.129568	0.003217
8	liveness	0.143838	0.119072	0.125679	0.132344	0.002529
9	valence	0.219438	0.218000	0.103723	0.106255	0.008716
10	tempo	107.361000	100.507560	0.136799	0.135085	0.001755
11	duration_ms	315879.720000	307178.840000	0.128447	0.129564	0.003214
12	time_signature	3.700000	3.700000	0.138372	0.139153	0.000637

New model:

	Features	Y_true Mean	Y_pred Mean	Y_true Norm Mean	Y_pred Norm Mean	MSE
0	danceability	0.302178	0.301178	0.127791	0.129640	0.003197
1	energy	0.213662	0.188083	0.119724	0.115539	0.006668
2	key	4.720000	4.720000	0.117560	0.114747	0.006841
3	loudness	-16.757540	-18.187820	-0.136499	-0.135423	0.001662
4	mode	0.660000	0.660000	0.114891	0.114891	0.006800
5	speechiness	0.040494	0.040088	0.139140	0.139910	0.000426
6	acousticness	0.927420	0.914120	0.140837	0.140651	0.000217
7	instrumentalness	0.729223	0.725177	0.131707	0.129780	0.003161
8	liveness	0.143838	0.120132	0.125679	0.132209	0.002563
9	valence	0.219438	0.224540	0.103723	0.105193	0.008937
10	tempo	107.361000	100.955760	0.136799	0.135135	0.001741
11	duration_ms	315879.720000	307514.380000	0.128447	0.129631	0.003197
12	time_signature	3.700000	3.700000	0.138372	0.139153	0.000637

The proposed evaluation metric suggests that the quality of our results was almost not impacted by the change of model. This makes sense since we only took away about 50 songs (the size of the input playlist) from our data set of 3800 to train the model. But we will keep working with our improved version of the model as it logically makes more

sense to use. Moreover, this means that our model makes good predictions feature-wise. However, since we have a relatively small data set, there is a feature overlap between songs of different genres for instance, which provides us with seemingly unrelated songs. This issue should be fixed as we increase our data. We made three more tests to compare the MSE here. Overall, the change of model did not significantly impact the predictions, but increased the performance of our model drastically in terms of time complexity.

As mentioned, our next steps are to expand our dataset and test it to make sure it really increases the recommendation quality of our results, as well as to finish polishing the website and model. Furthermore, we must figure out how to make sure our model generates exactly 50 songs each time - sometimes it generates 49 songs, sometimes 51, which is not a big issue, but we wish to keep our results consistent.

Here are more examples of generated playlists.

2. Final demonstration proposal

We have already managed to integrate our first version of our model in a webapp with the help of Shahrad's friend who is very knowledgeable in web development, since we are not very experienced in this field. We focused more on the backend and the machine learning algorithm, while he helped us connect the dots with the frontend. He used React JS to create our webapp, while we provided him with a Django backend.

In our webapp, the user logs into their Spotify account, and the authentication and logging happens thanks to the Spotify API. Then a user can see and select one of their playlists they want to generate new recommendations from, which are displayed on the right hand side. Then our algorithm runs and a new page is displayed which displays the new playlist generated (the first 20 songs). The user can listen to each song in that new playlist individually or open their full playlist on Spotify, and can return to the home page and logout or try it again.

To see what our proud project looks like so far: Mixerify