MAIS 202 - PROJECT DELIVERABLE 2

Noa Kemp (260898203)
Aymen Boustani (260916311)
Sanjeev Lakhwani (260954760)

Due: February 22 nd , 2021 at 11:59PM EST
Over the course of MAIS202, you will be completing a machine learning based project of your choice for the final project. You will also demo your project by integrating it into a webapp (or something more advanced). To conclude, you will be writing a blog post to share your project with others.
Submission

The deliverables may be completed individually or in teams of 2-3. Keep in mind that the project scope and grading will take into consideration the size of the team. All deliverables should be electronically submitted on Github and myCourses. They should also be completed with the same academic integrity and standards expected at McGill University. A quick tip: if you are using results or methods from an academic paper, you can go on their arxiv page and download their citation.

Submit both your well-documented code and report.
Deliverable Description

In this deliverable, you will discuss your progress and report your preliminary results. Be precise in your explanation and report. If you discussed your approach with other students, you should honour them in your report.

1. Problem statement: Restate the initial project that you proposed in deliverable one in 2 - 3 sentences. Be sure to refer back to this problem statement in the following questions.

Build a song recommendation/advisor based on what a user listens to; it would output a recommended playlist specifically built for the user.

2. Data Preprocessing: Confirm the dataset you are working with. State any changes from the initial dataset you chose. Discuss the content of the dataset (number of samples, labels, etc). Describe and justify your data preprocessing methods.

We will be primarily using the Kaggle spotify dataset which contains 174389 samples. We added additional tracks which are also from Kaggle.
From the original labels on the file, we will be using all of them except the key and release date. The key is not relevant except knowing if the song is in a major or minor key; and that is addressed in the ‘mode’ label. The release date is not relevant as we already have the year associated with the song.

We are also converting popularity, which was from 0 to 100; to go from 0 to 1 by dividing everything by 100. We are converting the duration from ms to minutes (dividing by 60000) and just looking at the integer part, we do not think it is relevant to have the exact duration in ms; and just having the minutes is enough to tell the preferences of the user. We also converted the name of the artists from object type to string type.

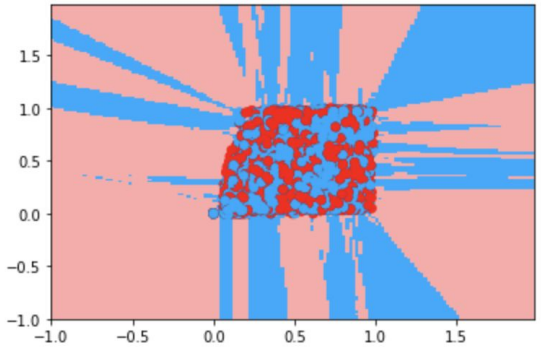
For KNN; we have dropped the names of the songs (we are not going to chose a song which has a similar title; as it is not relevant) and dropping the name of the artists as well, as it might be more interesting to recommend songs independently of the artist, in order to give other artists more visibility.

3. Machine learning model: In the first deliverable, you proposed a model for your project. If you decided to change your model, explain why. Restate your chosen model and elaborate on the design decisions. Report the following:

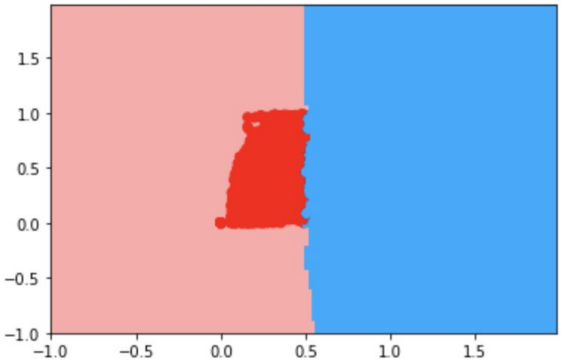
We are looking at how our data behaves using Logistic regression and KNN models.

a. Discuss the framework and tools that you used for your model. Explain your choice. Provide architecture graphs as appropriate. - Justify any decision about training/validation/test splits, regularization techniques, optimization tricks, setting hyper-parameters, etc.

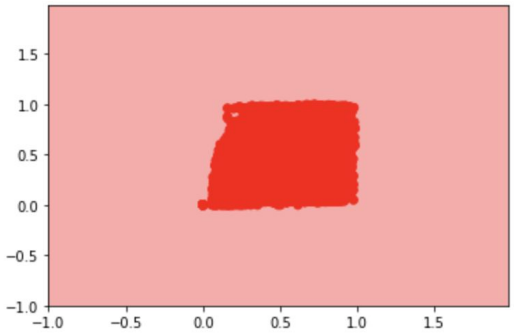
KNN:
By testing, first with a really small portion of the data set (around 4000 songs) and going up to around 50000 songs, we notice that the more songs are used, the better is the accuracy of the model.
But we notice that the worst case scenario is if the user likes really different types of songs, and listens to very different things; meaning that the likings of the songs are random. That yields a really low accuracy (less than 50%).



Compared to having some sort of link between the likings of the user (here we set that the user only likes songs that are at least 50% danceable), which yields a better accuracy (around 70% for test data size being 40% of 50000 samples)

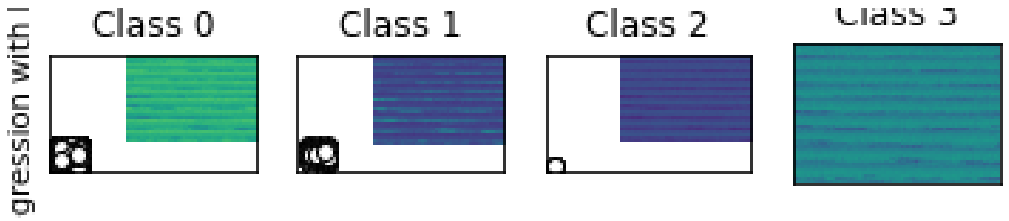


We also set very niche/specific likings (very specific criteria) to represent a user only liking certain types of songs in order to form a cluster of data:



Logistic Regression:

We established a class called ‘taste’ which indicates whether or not the user likes a song. To optimize our results, we subdivided into 4 variables, going from 0 to 1. The greater the value is, the more the user likes the song. This is the value the model is trying to predict. We’ve randomly picked 3000 artists and attributed to their songs a value between 0 and 1 indicating the degree of appreciation. The model relies on 10 features to predict this value. We’ve tried to reduce it but haven’t seen it much change in the accuracy score. That is, the more songs are fitted into the model, the more precise it is. However, we know that overtraining could lead to an overfitting model. We’ve picked the Stochastic Average Gradient solver. The final accuracy was low (around 62%). This is explained by the fact that the model isn’t fitting to what we’re trying to solve. Our training data was also containing too many features and we have to narrow it if we want it to be more efficient.



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

KNN:
We tested for different portions of the dataset; by looking at both random and niched distribution. To avoid underfitting/overfitting, we are using cross validation and randomly choosing each time our test data.
By tuning it (changing the n_neighbors value and the percentage of the data that is used as test set), we see that it changes the accuracy quite a bit; but it always stays around 50% for the random one and 70% for the niched one (and around 100% for the clustered one).

Logistic Regression:
We tested a smaller portion of the dataset to test our model. The model is overfitting.

c. Did you face any challenges implementing the model? If so, how did you solve it? At this point, don’t forget to save your trained weights! You will need them for the integration and/or testing your model!

We had difficulties working on the models all at the same time. We wanted to work on the same notebook through the Live Sharing function of VS Code but there were too many technical difficulties. We ended up sharing most of the tasks to not lose too much time.

4. Preliminary results: In this section, you will focus on the performance of your model. Confirm the metric discussed in Deliverable 1. Present a detailed analysis of your results, providing graphs as appropriate. In addition to an evaluation metric, discuss the overall performance of the model and the feasibility of the project with these results. Remember, graphs are beautiful and we love them!

See for KNN: https://drive.google.com/file/d/16GNqMcO2hSfuj7g1d5rkW0sZ_-l2zRy6/view?usp=sharing

- Currently, we are facing the issue of 'what if the user listens to a broad range of music?', since if that is the case then we get multiple smaller clusters, which right now seems like a challenge (since as the music is broader, the accuracy gets lower).

Logistic Regression: Model.ipynb

5. Next steps: Discuss your next steps. Describe the pros/cons of your approach and future work. Will you be altering your model? For example, will you be fine-tuning it? At this point, if you think that your model is not performing well and/or does not work, please reach out to an exec to see what you can do to improve it.

For KNN we are considering embedding methods such as cosine similarity and vector embedding to better represent our data in a hyperplane such that when applying the neighboring model, we get more accurate and intuitive results.

We're considering to base our Logistic Regression on less features and use it for narrower purposes such as determining to which genre a song belongs. We'll try to fine-tune it and use cross-validation on this model to obtain more accurate results.