MUSIC RECOMMENDATION SYSTEM
CODE NAME: SPOOFY

Project Contributors

This project is a joint effort between the college students FILIPPOU GEORGIOS, PETROGLOY SPYROS, and MAVRODI DIMITRA.

Introduction

All too often we and many of the people we know end up listening to the same songs over and over again, simply because the process of discovering new music is too inconsistent and time-consuming. To add to this, more often than not, the songs that actually end up adhering to our taste and become a staple in our playlists, are discovered through word of mouth and from other's recommendations. Specifically, recommendations made by friends, family and generally from people that know us closely. Actually searching for music, one personally enjoys, rarely results in any substantial finds. As a result, the process can be very random, inefficient, time consuming and frustrating.

But let's consider why recommendations from people close to us are so effective at choosing the songs we are most likely to enjoy. Such people have come to know us more in-depth, they know our likes and dislikes, our opinions, our personality as well us the things that interest us. This, obviously, also includes our taste in music. Knowing and processing all of this information unconsciously, they are immediately able to eliminate large swaths of music that we wouldn't enjoy, with a large degree of certainty. They can then go on to further narrow down their options using all the aforementioned personal information. The final selection is by no means infallible, however it is highly likely to be accurate, although this also scales with how well they know us.

However, the draw backs of this method are also significant and would be near impossible to escape as well. Firstly, they only know so many songs themselves to recommend from and searching to find new ones wouldn't be any more effective than if the person itself searched for them. Secondly, they are not always available to offer this help, as they have their own lives to live. Thirdly, they are not obligated to put in the effort to make these recommendations in the first place.

As can be seen in the graph below, despite these heavy restrictions, recommendations from friends and family still rank as the third most likely source of new songs for people around the world.

Recommendations based on personal information are clearly effective, however the above hurdle largely limit their capabilities. So, we endeavor to create a way

	Europe	US	Mexico	UAE	APAC
Radio/satellite radio	40	33	26	26	24
Music apps	33	25	53	32	45
Recommendations from friends and family	25	25	36	29	29
Movies/TV	24	21	32	31	34
Social media	23	21	49	55	45
Commercials	10	9	19	21	16
Websites/blogs	9	9	14	25	19
Video games	6	8	11	12	11
Podcasts	6	6	12	13	10

to make accurate recommendations without being held back by the human element. The way we believe that is achievable it through the use of algorithms.

A well-designed model would have access to potentially hundreds of thousands of songs, if not more and with the right feature handling it would also know them intimately. Furthermore, a machine designed to make these recommendations would always be available regardless of time of day and its judgement would be unaffected from any environmental factor such as fatigue, sleep deprivation and mental overload. Lastly a machine would be obligated to always answer, answer to the best of its ability and without any falsehood.

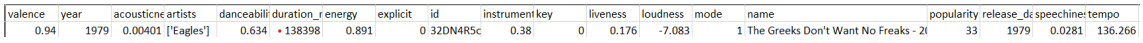
In other words a machine would make the process of discovering new music easy, quick and reliable. In order to make this possible we will use a large database of songs that have been analyzed and had their features extracted. We will then take a list of songs a user already likes as the only required input. This list will be used in order to train an algorithm to discover patterns inside the use's choices. Having found the characteristics the user is most interested in, the algorithm will then go through its extensive database to find the songs that most closely resemble the use's interests. It will then output the top songs it believes are most likely to appeal to the user.

Database Information

The first step in the creation process is the acquisition or creation of a database with as many different songs as possible. It's important to note that the diversity of the songs is especially important in this case, since the more quantitative and diverse our database is, the more people and tastes it will be able to effectively accommodate.

Each song in the database we will use has been dissected and analyzed in order to determine its specific characteristics. The ones we are interested in, and that could be of interest to the user, are as follows: release year, duration, acousticness, danceability, liveness, loudness, energy, instrumentalness, bpm, speechiness, lyric inclution, explicitness, voice pitch and popularity. The database contains 170653 different songs from all categories, most cultures, many time periods and multiple languages. Compared to its original state some categories of less importance have been removed and some categories have had name changes in order to make their

purpose clearer. Every row represents a different song and each column contains one of the above mentioned characteristics.



valence	year	acousticne	artists	danceabili	duration_r	energy	explicit	id	instrument	key	liveness	loudness	mode	name	popularity	release_da	speechine	tempo
0.94	1979	0.00401	['Eagles']	0.634	138398	0.891	0	32DN4R5c	0.38	0	0.176	-7.083	1	The Greeks Don't Want No Freaks - 2(	33	1979	0.0281	136.266

The creation of such a database, while possible, would be very time consuming, difficult and is beyond the scope of this project. Therefore, after contacting multiple organizations that specialize in data collection and categorization, we have managed to acquire a couple of different databases. Although none had the exact parameters we desired, after some editing and combining we managed to create our ideal database.

One of the issues in the previous milestone was the lack of user data. With the assistance of K we acquired some usable data that we could use for the early stages of the training. However, if we wanted to train a more accurate and versatile model we would require a lot more data than we could acquire on our own through surveys, at least not with the resources and the time limit we had.

To try and remedy this situation, we found a user matrix that contains the personal choice of multiple users, as well as their specific rating for these choices. This is an improvement compared to previous milestone where every choice held the same weight. Additionally this matrix also contains some personal information about each user account such as id, name, screen-name, number of followers etc. As mentioned above, all this information will assist in the training of the algorithm and allow it to have a more "intimate" view of each user, thus allowing it to make more accurate and personal recommendations.

We combined these files and made a ratings.csv file. The columns, or features, of this file are user id, song id and rate. The rate field represents the rate that a specific user gave to one song. We can have multiple entries of a specific user and multiple entries of a song, but we can't have an entry with the same combination of a specific user and a specific song twice.

Now that we have the above datasets, we managed to create a sparse array. In this array, each column stands for the ids of the songs and each row for the ids of the users. In each index, we have the rate that a specific user gave to a specific song. If a user hasn't given a rate to a song, initially we had NaN values but we converted them to 0.0. The name of this array is df-filled.

Now we have all the information that we require in order to create and prototype some initial algorithms.

Data Analysis and Algorithms

At first we tried to organize our data. So, we created a file named dataset information.html which contains useful information about our features. To extract this file, we used the ProfileReport command which is located in ydata-profiling library. We also made a histogram with respect to duration in millisecond to see how many songs have the same duration. We found that most songs last 192000ms (specifically 56). Finally, we plotted the correlations of our features using seaborn library. We passed our music dataset through a Principal Component Analysis (PCA). Through this process we managed to identify the most important and influential features of the songs in the database. Doing this allows us to reduce the dimensionality of

our database and therefore its complexity as well. By keeping a total of around 80 percent of the relevant features we minimize the risk of overfitting and reduce the presence of noise in the data, resulting in a more accurate and capable model. As a result, the starting 15 features were narrowed down to 8 principal components.

Having reduced the dimensionality of the dataset, we then used linear regression. To do this we first had to pass the data through a train-test-split function. Although we wanted to use the result of the PCA for the input variable and the list of chosen songs as an output variable, the length for input and output had to be equal. So, we created a new array with the same number of rows and a single column: "chosen". We then used a for loop in order to simulate a user choosing songs they like and created an array with 50000 such choices. Afterwards we went through the entire database and each song that wasn't in the list would be marked as 0 in the new array, and each song that was on the list would be marked as 1. Afterwards it was in an acceptable state and we ran it through the train-test-split and the linear-regression function following that.

In this way we basically trained the model to be able to predict whether a song would be a 1 or a 0. To measure the accuracy of our model we calculated its RMSE which came out to be around 0.4357304, which we deemed acceptable. The resulting evaluations were then sorted in descending order, with the ones at the top being the most likely to appeal to the simulated user.

We also trained this algorithm with the real user K, who chose around 70 songs. He was then offered the top 10 choices to rate from a minimum of 0 to a maximum of 10. He very unhesitatingly gave them and overall score of 5/100. This seems to contradict the result of the RMSE which scored fairly high. This could be the result of a number of causes such as the model simply being incapable of predicting the choices of K specifically, RMSE not being the best method to ascertain its accuracy, with the real value being much lower than was calculated, K's judgment being affected by outside factors such as high stress at the time, among others. In any case the results are inconclusive though we suspected linear regression not being the best option since the beginning, so we moved on.

Having acquired a user matrix, we then decided to perform the SVD method using the df-filled array. The algorithm performs the factorization of this array and then, it multiplies its components to fill in the zero values. Then, we find the indices with the largest values for a specific user (5 in default). Songs with the largest values in this array, are having more chances to be liked by the user.

We then tried to implement two deep learning methods. Firstly, we implement a Neural Network using only the numpy library. Our NN had 1 hidden layer with 5 neurons. The input array was the output of our PCA algorithm and the output is a weighted matrix, where bigger weights correspond to the songs that a user may like. Then, we converted this matrix to a binary matrix, where 1 corresponds to the indices with values greater than zero and 0 to the values less than zero. And then, we used MSE to compare the binary matrix with the initial matrix. Our MSE is 0.254504 which is fairly low.

Our second Neural Net was made with the keras library. Here we have an input layer with 12 neurons and relu as an activation function, one hidden layer with 8 neurons and again relu, and finally, one output layer with 1 neuron and sigmoid function as an activation function. We run this bunch of code and we find the predictions.

And finally, we used the k-nearest neighbors algorithm in order to find the nearest songs in the multidimensional space. We represent each song as a vector and we try to find the rest k vectors with the smallest distance from the initial, and by the meaning "distance", we mean the cosine similarity of the vectors. Then, we print the initial song (obviously with distance 0), and then the 2 nearest songs (with the smallest distance). These 2 songs are being recommended to the user.

We also trained the K-nearest neighbor using the data of K. The songs chosen were the same as for the linear regression. The algorithm would then go on to recommend 10 songs that it believes would appeal to K. Once again K was presented with these songs and asked to rate them in the same manner. The resulting score this time was an 52/100. A very significant improvement over linear regression and a decent score in general. Though once again it should be noted that this is simply one user's experience and it is possible that another user with a different taste would find less success.

Conclusion

To sum up with the goal of creating a music recommendation system to simplify and expedite the process of song discovery, we tested out different methods and algorithms. We tried PCA followed by linear regression, SVD, numpy-based Neural Network, keras-based Neural Network and k-Nearest Neighbor, clustering and other kinds of NN. Out of the ones we tried we rejected the last two and from the ones that remain, we believe that k-NN is the most reliable and accurate. That being said we had some trouble producing proper evaluations from some of the algorithms and any at all for others.

We believe the best algorithm for a music recommendation system (MRS) to be among the ones mentioned above. However not in their current state. Ideally we would like to have a large number of users that could rate the recommendations from each algorithm. Using methods like RMSE, simulated users and reviews are adequate for the moment, however the closer we are to actual users, who are the target consumers of a MRS, the better the algorithms can be trained and the better the recommendations they can make.

Also considering that the market for MRSs is currently oversaturated, with apps like Spotify, youtube music, apple music, amazon music, deezer, soundcloud etc, we are considering approaching the market from a different angle, as competing directly against them would be unrealistic. Most if not all of the above mentioned apps use an algorithm which is trained with hundreds of thousands of songs, that is able to make general recommendations. What we are thinking of doing is offering a more personal service. Each user of this service would send their playlists the songs they like the most and we would train the appropriate algorithm specifically for that user. This would theoretically allow for even greater accuracy than the general purpose algorithms used by bigger companies. With the number of users such companies try to accommodate it would be infeasible for them to attempt to employ such a model, as they would have to train millions of different models at any given time. But for a smaller business, such as ours, it would be possible. Of course the problem with this method remains that it doesn't scale well to more customers, however this also means that it is mostly safe from being overshadowed by pre-existing massive cooperations, as there will always be a steady flow of dissatisfied customers we could

accommodate.

Bibliography

- https://medium.com/artificialis/music-recommendation-system-with-scikit-learn-30f4d07c60b3
- Music Database: https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset/input
- User Database: https://www.kaggle.com/datasets/whoseaspects/genuinefake-user-profile-dataset/code

Acknowledgments

Special thanks to Kapakos Giorgios as K.