Name: Borbála Jakab
Neptun ID: YUBYNR

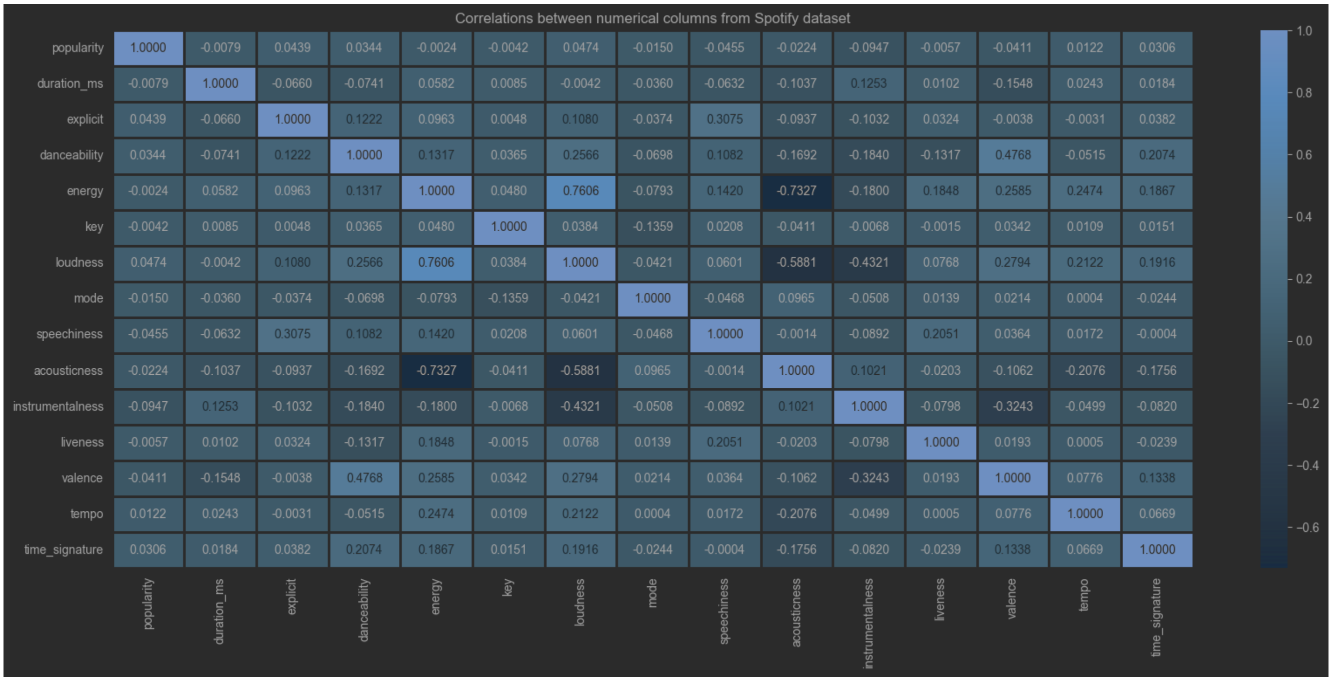
Data Mining and Machine Learning MSc Course

Multi-faceted Analysis and Classification of Spotify Tracks

Data Exploration: I downloaded the dataset from the website (https://huggingface.co/datasets/maharshipandya/spotify-tracks-dataset). The dataset I am working with is a Spotify track dataset which according to the website was uploaded 4 months ago (2023.06.). After downloading it I opened it in PyCharm and started to collect more information about what my dataset contains and what I am going to work with.

The data I downloaded was a 'csv' file which means it is a coma separated dataset. At first glance after opening the dataset with Pandas I could tell that it contains 114000 rows and 20 columns. To be able to be more familiar with the dataset I check the beginning and the end of my dataset. Also, I check all the column names and the data in the table. Overall, my table contains a few columns with string type of data but most of my columns contains numerical values (integers and floating numbers). After I get a little more knowledge about the nature of the data I checked for missing values. My data only contains 3 missing values, and all these values are in the same row. So, after careful consideration I decided that my data is big enough to be able to delete the row with missing values without significantly changing the results of any calculations with my dataset. After this change I have 113999 rows. Just be to be sure I check the number of missing values again. After this change I decided to check if there is any duplication in my data and I did delete those rows which had an identical twin. After deleting them my dataset contains 113549 rows. As I was not exactly sure at the beginning which columns, I am going to us from now on I only deleted one column ('track_id') which contained only a bunch of numbers and letter. To collect more information about my data I used the describe() function and the info() function on my data.

I also did some correlation analysis with a heatmap between the columns which contains numeric values to have a better understanding which parameters have significant correlations between them (*Figure 1.*).



	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature
popularity	1.0000	-0.0079	0.0439	0.0344	-0.0024	-0.0042	0.0474	-0.0150	-0.0455	-0.0224	-0.0947	-0.0057	-0.0411	0.0122	0.0306
duration_ms	-0.0079	1.0000	-0.0660	-0.0741	0.0582	0.0085	-0.0042	-0.0360	-0.0632	-0.1037	0.1253	0.0102	-0.1548	0.0243	0.0184
explicit	0.0439	-0.0660	1.0000	0.1222	0.0963	0.0048	0.1080	-0.0374	0.3075	-0.0937	-0.1032	0.0324	-0.0038	-0.0031	0.0382
danceability	0.0344	-0.0741	0.1222	1.0000	0.1317	0.0365	0.2566	-0.0698	0.1082	-0.1692	-0.1840	-0.1317	0.4768	-0.0515	0.2074
energy	-0.0024	0.0582	0.0963	0.1317	1.0000	0.0480	0.7606	-0.0793	0.1420	-0.7327	-0.1800	0.1848	0.2585	0.2474	0.1867
key	-0.0042	0.0085	0.0048	0.0365	0.0480	1.0000	0.0384	-0.1359	0.0208	-0.0411	-0.0068	-0.0015	0.0342	0.0109	0.0151
loudness	0.0474	-0.0042	0.1080	0.2566	0.7606	0.0384	1.0000	-0.0421	0.0601	-0.5881	-0.4321	0.0768	0.2794	0.2122	0.1916
mode	-0.0150	-0.0360	-0.0374	-0.0698	-0.0793	-0.1359	-0.0421	1.0000	-0.0468	0.0965	-0.0508	0.0139	0.0214	0.0004	-0.0244
speechiness	-0.0455	-0.0632	0.3075	0.1082	0.1420	0.0208	0.0601	-0.0468	1.0000	-0.0014	-0.0892	0.2051	0.0364	0.0172	-0.0004
acousticness	-0.0224	-0.1037	-0.0937	-0.1692	-0.7327	-0.0411	-0.5881	0.0965	-0.0014	1.0000	0.1021	-0.0203	-0.1062	-0.2076	-0.1756
instrumentalness	-0.0947	0.1253	-0.1032	-0.1840	-0.1800	-0.0068	-0.4321	-0.0508	-0.0892	0.1021	1.0000	-0.0798	-0.3243	-0.0499	-0.0820
liveness	-0.0057	0.0102	0.0324	-0.1317	0.1848	-0.0015	0.0768	0.0139	0.2051	-0.0203	-0.0798	1.0000	0.0193	0.0005	-0.0239
valence	-0.0411	-0.1548	-0.0038	0.4768	0.2585	0.0342	0.2794	0.0214	0.0364	-0.1062	-0.3243	0.0193	1.0000	0.0776	0.1338
tempo	0.0122	0.0243	-0.0031	-0.0515	0.2474	0.0109	0.2122	0.0004	0.0172	-0.2076	-0.0499	0.0005	0.0776	1.0000	0.0669
time_signature	0.0306	0.0184	0.0382	0.2074	0.1867	0.0151	0.1916	-0.0244	-0.0004	-0.1756	-0.0820	-0.0239	0.1338	0.0669	1.0000

Figure 1.: Correlations between the numerical columns from the Spotify dataset. Overall, we can see significant positive correlation between 'loudness' and 'energy' and significant negative correlation between 'acousticness' and 'energy'.

Then to not just try to image but be able to see the distribution of different parameters in my dataset I did the visualization on the features in my dataset (*Figure 2.*). Most of the distribution of the different parameters are pretty useful to have a better understanding of the dataset. But for example, the 'explicit' and 'mode' column is not really helpful in this kind of visualization as these are binary variables. I did a separated visualization about the genres in my dataset (*Figure 3.*). Honestly, I personally do not think that is significantly helpful as most of the genres are moving around 1000. There are only a few genres (romance, classical, dance, german) whit a noticeably decreas in their numbers but this is because these genres had rows which were totally identically. And, in 'k-pop' genre there is one row which I deleted because it contained missing values.

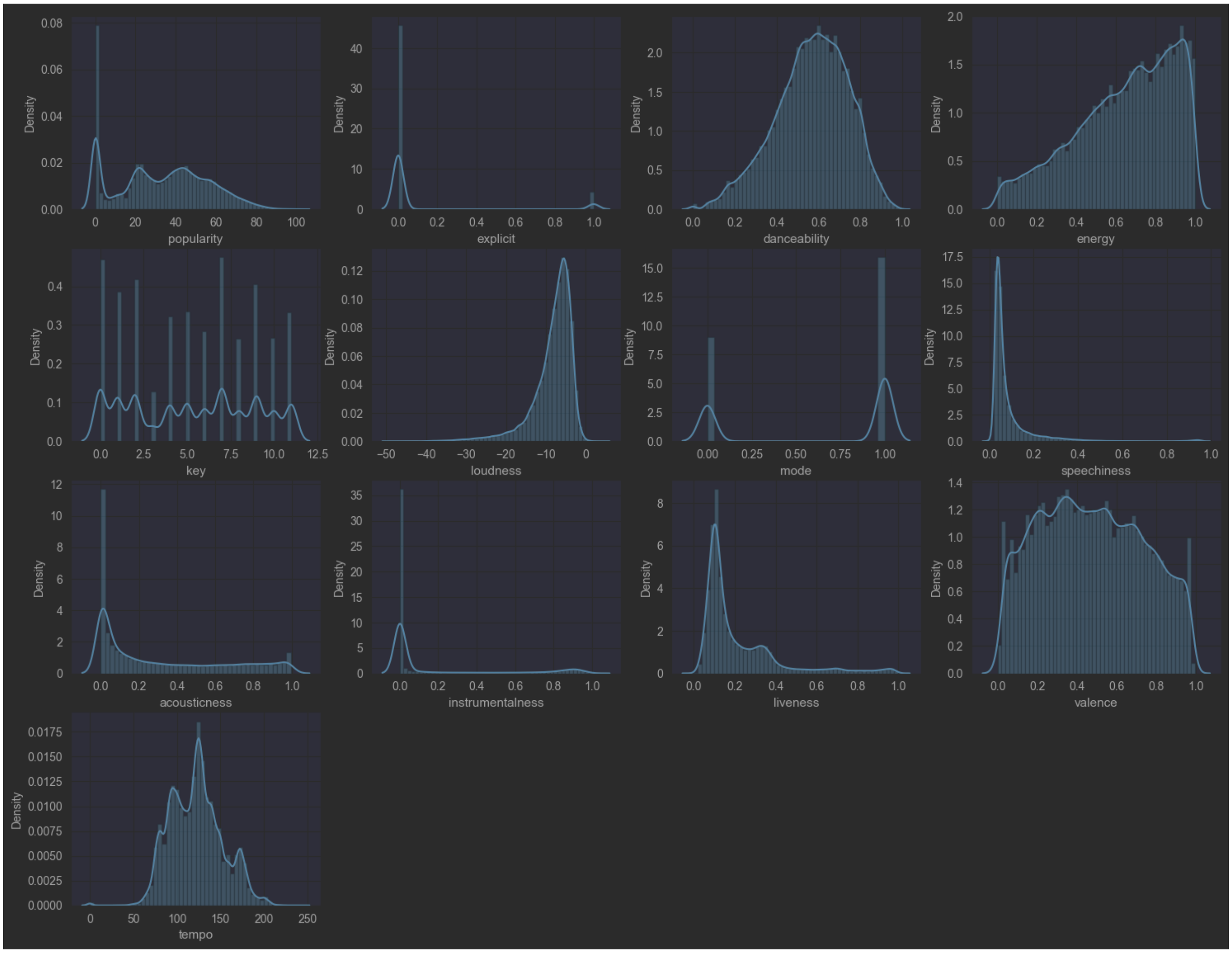


Figure 2.: Distribution of the numerical parameters in the dataset

Name: Borbála Jakab
Neptun ID: YUBYNR

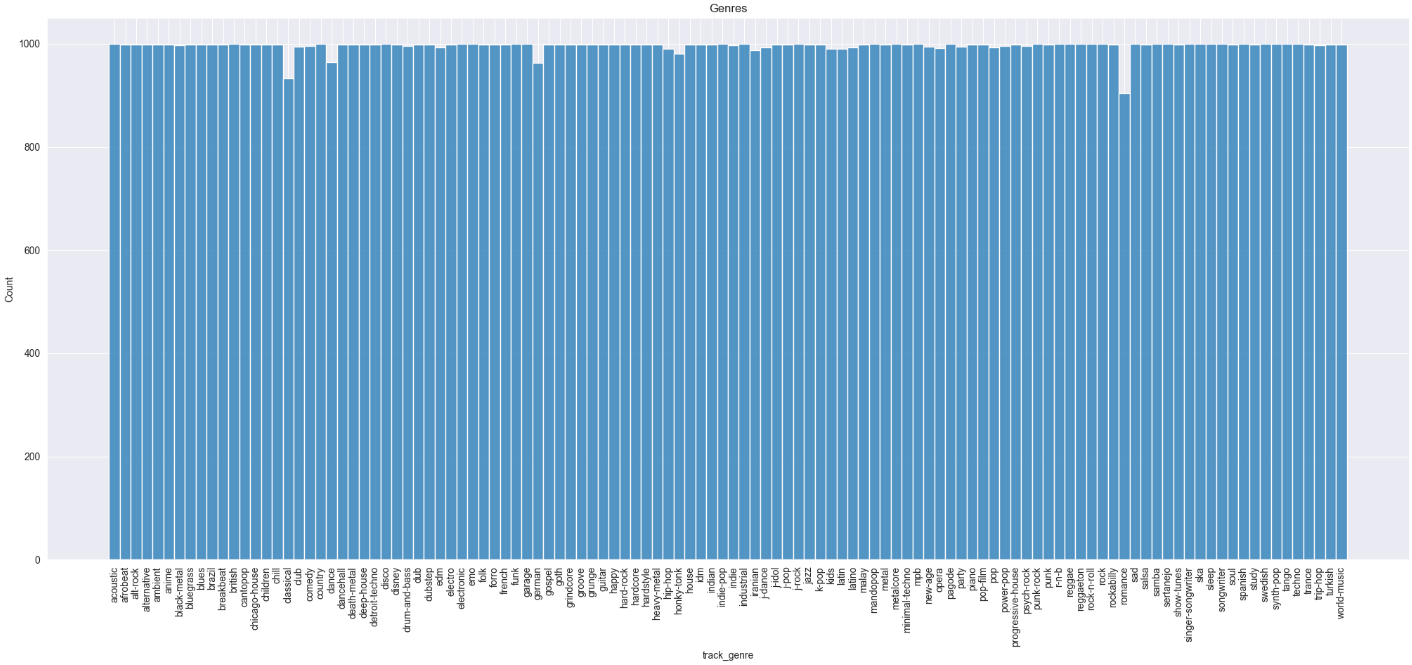


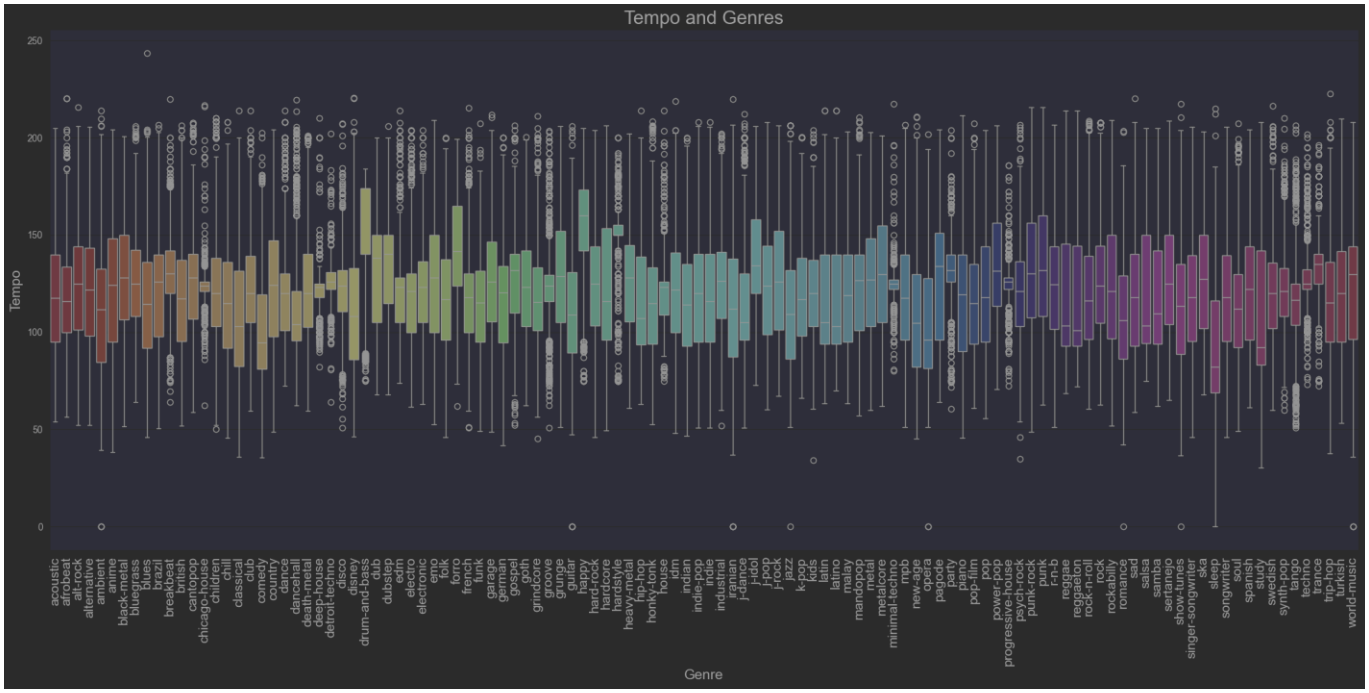
Figure 3.: Distribution of the Genres in the dataset.

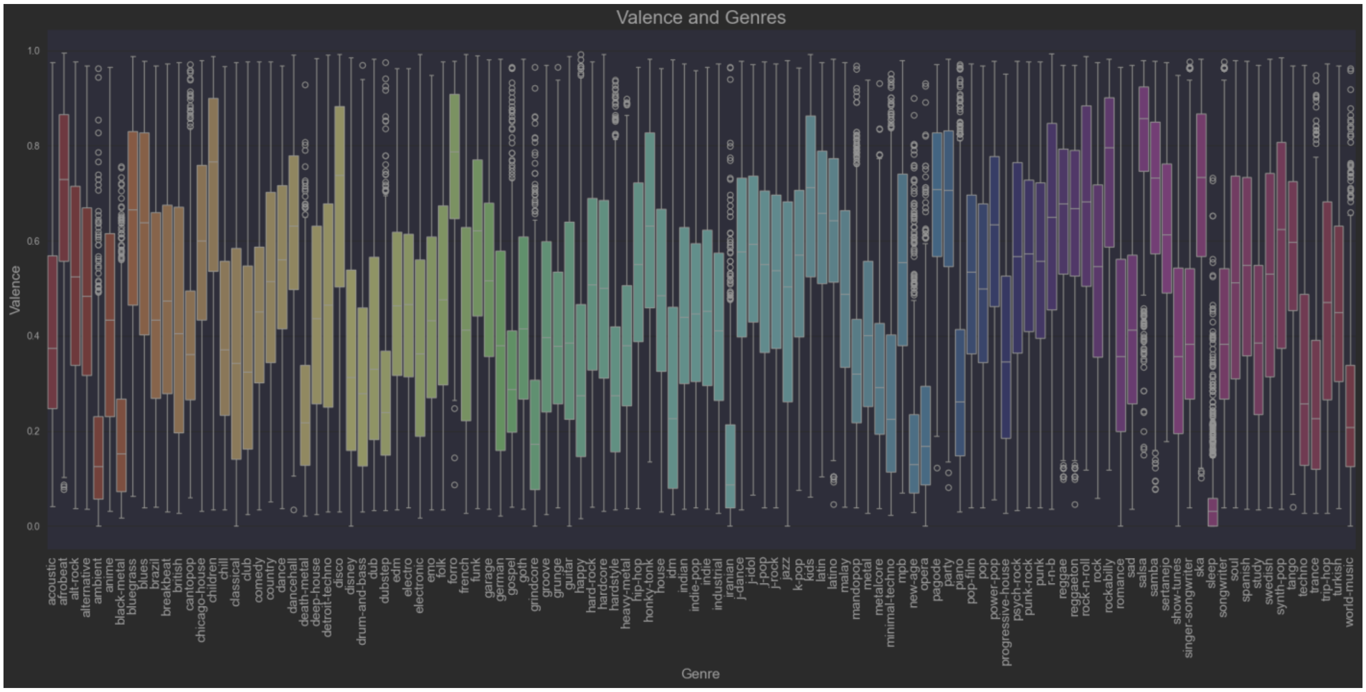
Feature Engineering: In this part as my task asked, I only kept the primary artist from the 'artist' column (the artist names were separated by semicolon), and I also changed the 'explicit' column from True or False into binary format: 0 and 1. (1 = True and 0 = False).

After this I saved my dataset into a new 'csv' because from this point, I am not going to manipulate my dataset permanently so I wanted to have it basically as check point which I can use from now on. I saved it as '**spotify_data.csv**'.

Name: Borbála Jakab
Neptun ID: YUBYNR

Data Visualization: After getting a better visualization about the distribution of some parameters I wanted to see the distribution of them across the different genres in my dataset (*Figure 3.*). I did not visualize all the audio features across all the genres just a few one I thought would be interesting.





Name: Borbála Jakab
Neptun ID: YUBYNR

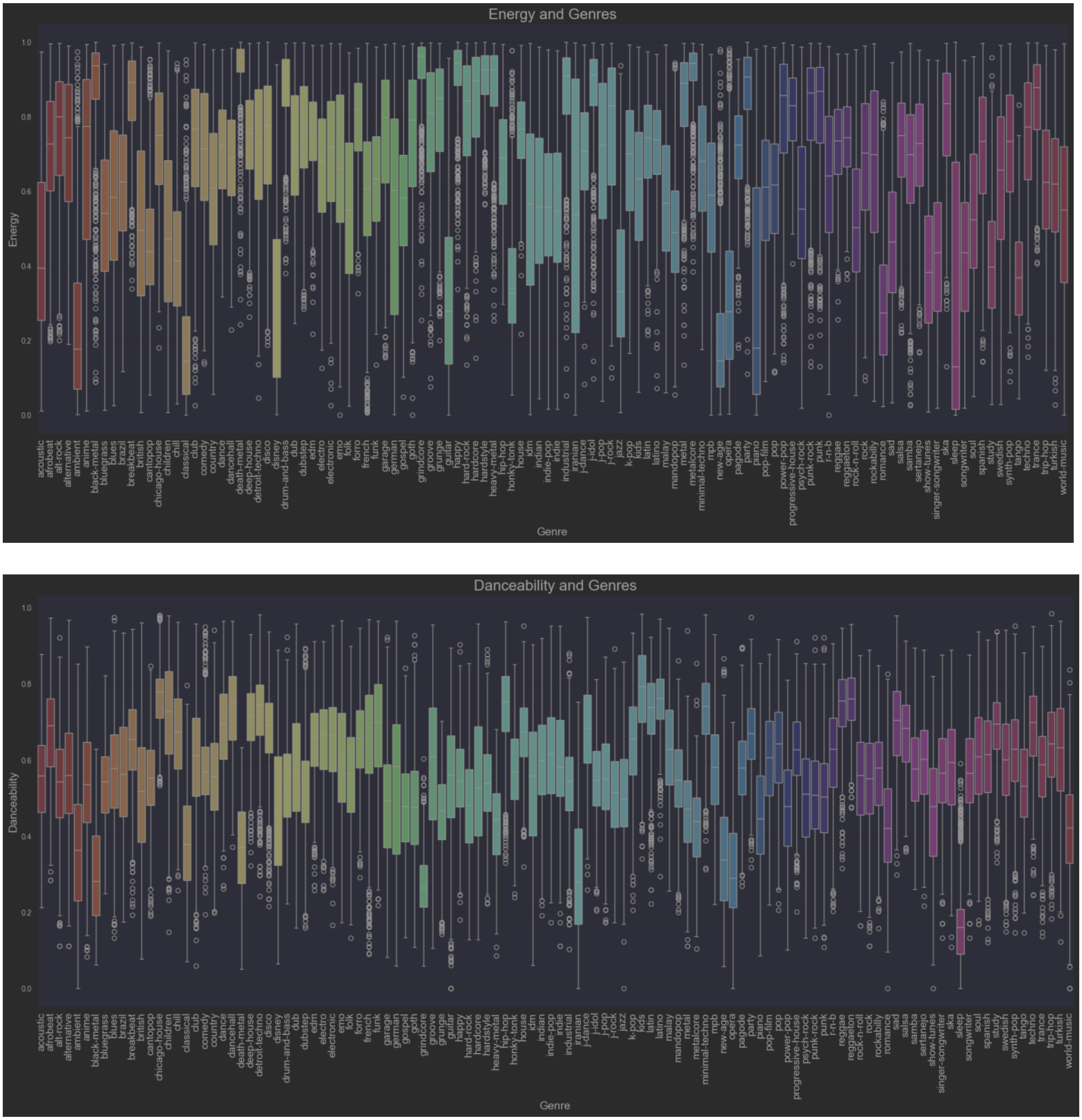


Figure 3.: Distribution of various audio features across different genres

As we can see on the Figures in these 4 features the distribution is highly variable across different genres. From these 4 features I would say that the biggest difference can be observed in 'Energy' in different genres. In 'Tempo' these are a few outliers in genres but mostly the values are closer to each other. 'Danceability' is also highly distributed but in my opinion not as much as 'Energy'. In my Python file I visualized more features but the distribution there was quite similar across the genres. For example, if we take the feature 'Key' or 'Mode' they are not really helpful because different genres are distributed similarly to each other.

After this I Visualize the correlation between features like danceability, energy, and valence (*Figure 4.*). On the correlation between 'Energy' and 'Loudness' we can clearly see the positive correlation. While 'Accountancies' and 'Energy' shows a negative correlation. Also, in these figures as a third element we can see in blue (False) and orange (True) which song is explicit. And I would say explicit songs cluster a little bit when the 'Energy' is higher. While 'Energy' and 'Danceability' gives us a little bit strange curve. Also 'Danceability' and 'Valence' also gives us a positive correlation but not as positive as the first one I mentioned.

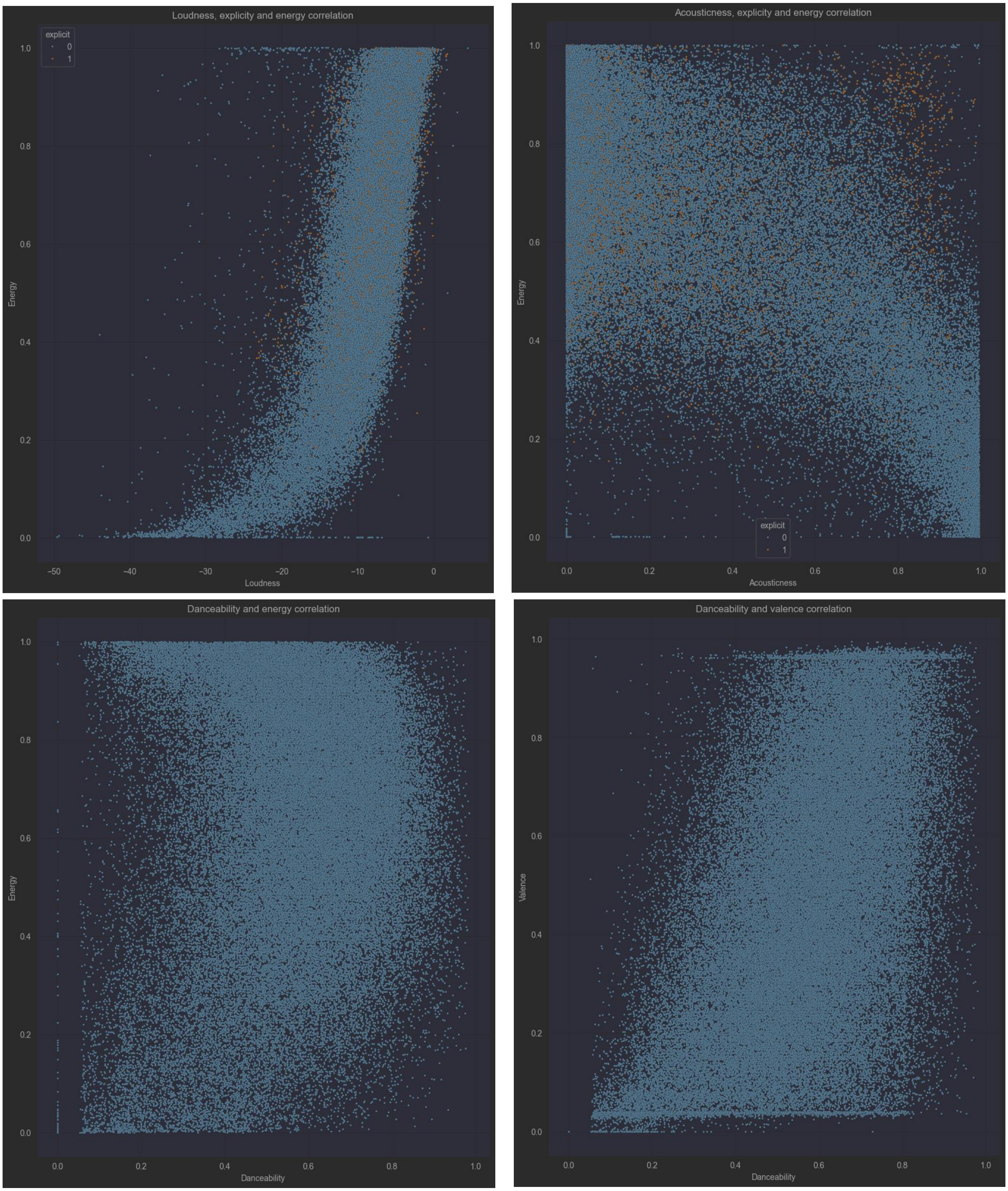


Figure 4.: Correlation between features like danceability, energy, explicit and valence.

Then, I used dimensionality reduction techniques (PCA) to visualize clusters of songs in a 2D space. The dataset contains a lot of variables. So, plotting it is not feasible this is why I will reduce the features or dimensionality of the dataset to 2 as it can be visualized using 2D graphs (*Figure 5*).

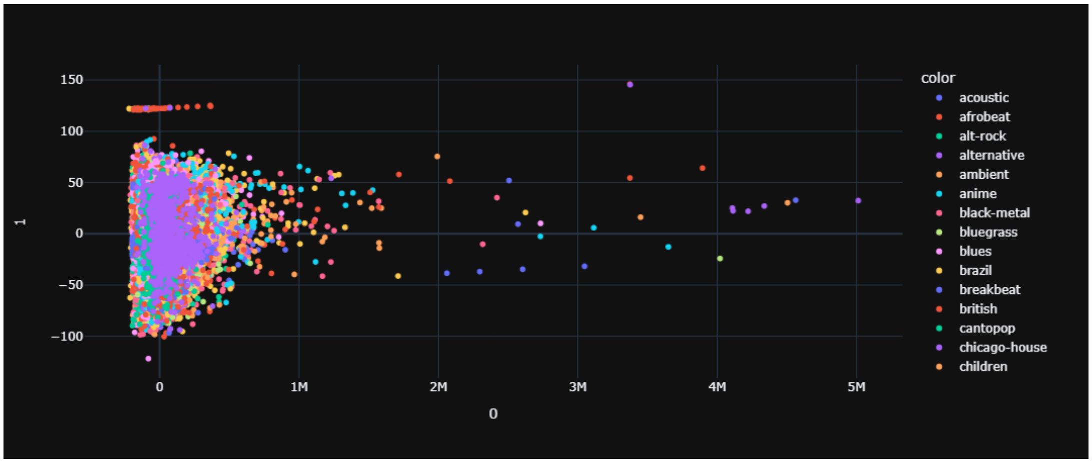


Figure 5.: Dimensionality reduction techniques (PCA) to visualize clusters of songs in a 2D space. (The colors represent the different genres)

We can see on Figure 5 that most of my datapoint cluster on one side of the figure. The reason for this is that I did not normalize my dataset beforehand. Probably the variance is all on just one or two variables with a bigger scale. So, a redid my analysis with normalized data and I got totally different outcome (*Figure 6*.). We can see 2 main cluster on the figure and the scaling of the axles does not show that high range.

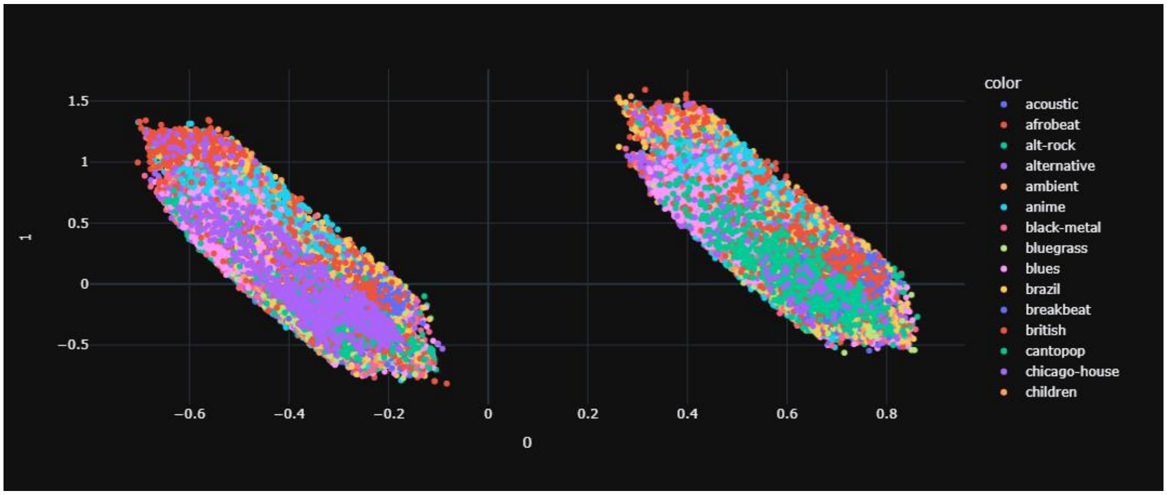


Figure 6.: PCA with normalized dataset. (The colors represent the different genres)

Name: Borbála Jakab
Neptun ID: YUBYNR

Recommendation System: I build a content-based recommendation system that suggests songs similar to a given song based on audio features. For this I used cosine similarity. What was interesting in this part that there are a lot of songs in my dataset that are basically duplicate only their genre differs. So, I decided to only keep one of the songs because their audio features are identical and, in this part, I did not use their genre for the prediction. So, I dropped almost 40.000 rows from my dataset for this task.

I tried out my recommendation system and I think it works acceptable. There are a few times when it is not giving the same music vibe for me, but I always check, and the audio features of the songs are quite similar to each other every time.

Classification Based on Audio Features: And as I just mentioned a few lines before I found it really interesting that the same song with similar audio features is in different genres. Because here in this part I am going to train various classifiers to predict the genre of a track based on its audio features. But there are a lot of songs (almost 40.000) with identical audio features but with different genres.

Because of there I decided to not just use the full dataset but to use my original dataset but with some manipulation. Overall, I used 4 different datasets, and I trained all my classifiers with all of my separate datasets. And in the end, I compared them to each other based on their accuracy score (*Figure 7.*).

The 4 different datasets are the so called '**full dataset**' where I basically use the whole dataset, I got at the beginning of the analysis only a few rows are dropped where the rows are totally identical or there are missing values. The second dataset is the '**filtered dataset**'. It is the dataset where I dropped almost 40.000 rows where the songs were identical only the genre were different. The third dataset '**less genre**' is a dataset which contains the same number of rows as the first one, but I manually reduced the number of genres. Using my best knowledge, I merged genres together so my classifier can work with less genre. The last dataset '**popular genre**' only contains the 20 most popular genre based on the averaged popularity score. This contains the least data compared to the other datasets.

For these 4 datasets I used 4 different classification method. I used Dummy Classifier, Decision Tree, Random Forest, and Gradient Boosting. Overall neither of these methods worked well on my dataset. For the Decision Tree and Random Forest, the main problem was that almost every time it overfitted and the accuracy score was pretty bad. Using my best knowledge, I tried to tune the hyperparameters manually because RandomSearcCV and the GridSearcCV did not worked for me well. The best model I get as results from these always had way worse accuracy score than the default model and it was still overfitting. So, using my best knowledge I tried to tune them manually and a few times I could actually get better results than with the default parameters. But the results are still pretty bad.

While with Gradient Boosting it was the opposite. The model tends to underfit. But I had the same problem with tuning the parameters. It was always wors then the default one, so I left them with the default parameters.

Also, unfortunately my own laptop is not strong enough to tune the hyperparameters accurately. A lot of times I got 'Memory Error' during tuning the hyperparameters. I tried it with different methods but only the manually tuning version worked for me. I am sure that there are way better ways to tune them, but I did not have the equipment and capacity to do it in another way.

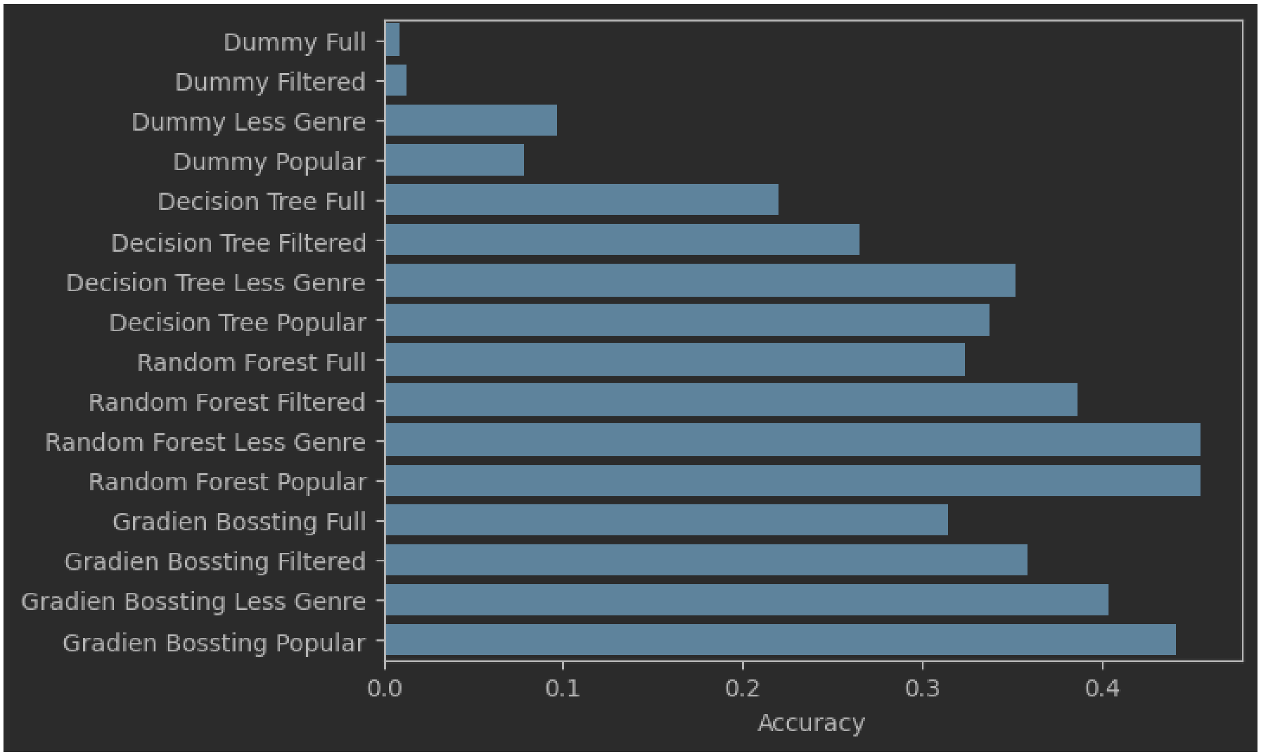


Figure 7.: Accuracy scores of the classification models

From the results in the figure above I can tell that my models are not working well unfortunately. But if we compare the datasets and the classification models to each other we can see some improvements in them. I used the Dummy Classifier just to have a basic view what should I expect from the other classification models.

Overall, the 'Less Genre' and the 'Popular Genre' dataset have better accuracy score then the other two dataset in every classification models. The reason for this can be that the 'Less Genre' dataset only works with less than half of the genres compared to the original dataset. So, it is somewhat easier to classify the genre because there are less. For the 'Popular Genre' dataset the above mentioned also true but it also contains way less rows than the original dataset.

From all the classification models I would say that the Random Forest and the Gradient Boosting worked the best with my dataset.

Analysis of Popularity: During this I also used the dataset where the totally identical songs were dropped from my dataset. Also, as I am trying to predict the popularity score based on its features, so I had to delete those rows where the popularity score was only available once in the dataset. So, I had to delete two rows with the popularity score 100 and 99 because with this popularity scores only one row was findable. After this as my target variable is the popularity score, I made a distribution about it (*Figure 8.*)

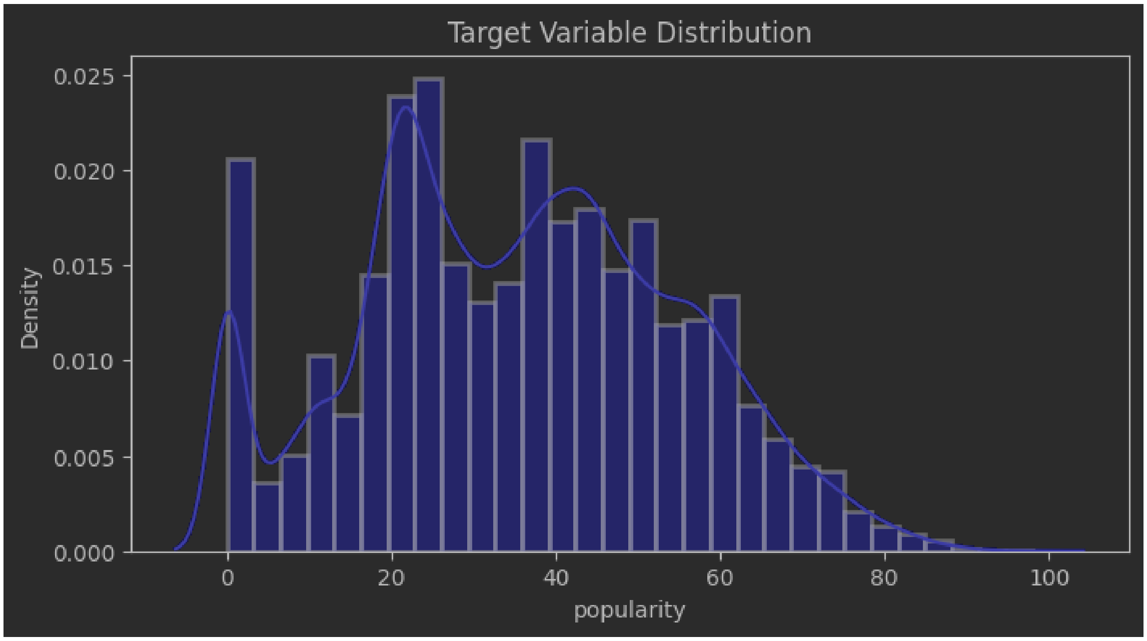


Figure 8.: Distribution of my target variable

Overall based on the figure above I can tell that here are a few pop-ups interval. Like at the beginning and around 20. But the distribution of the popularity scores is quite different.

Then I compared the popularity scores to the other features (*Figure 9*.).

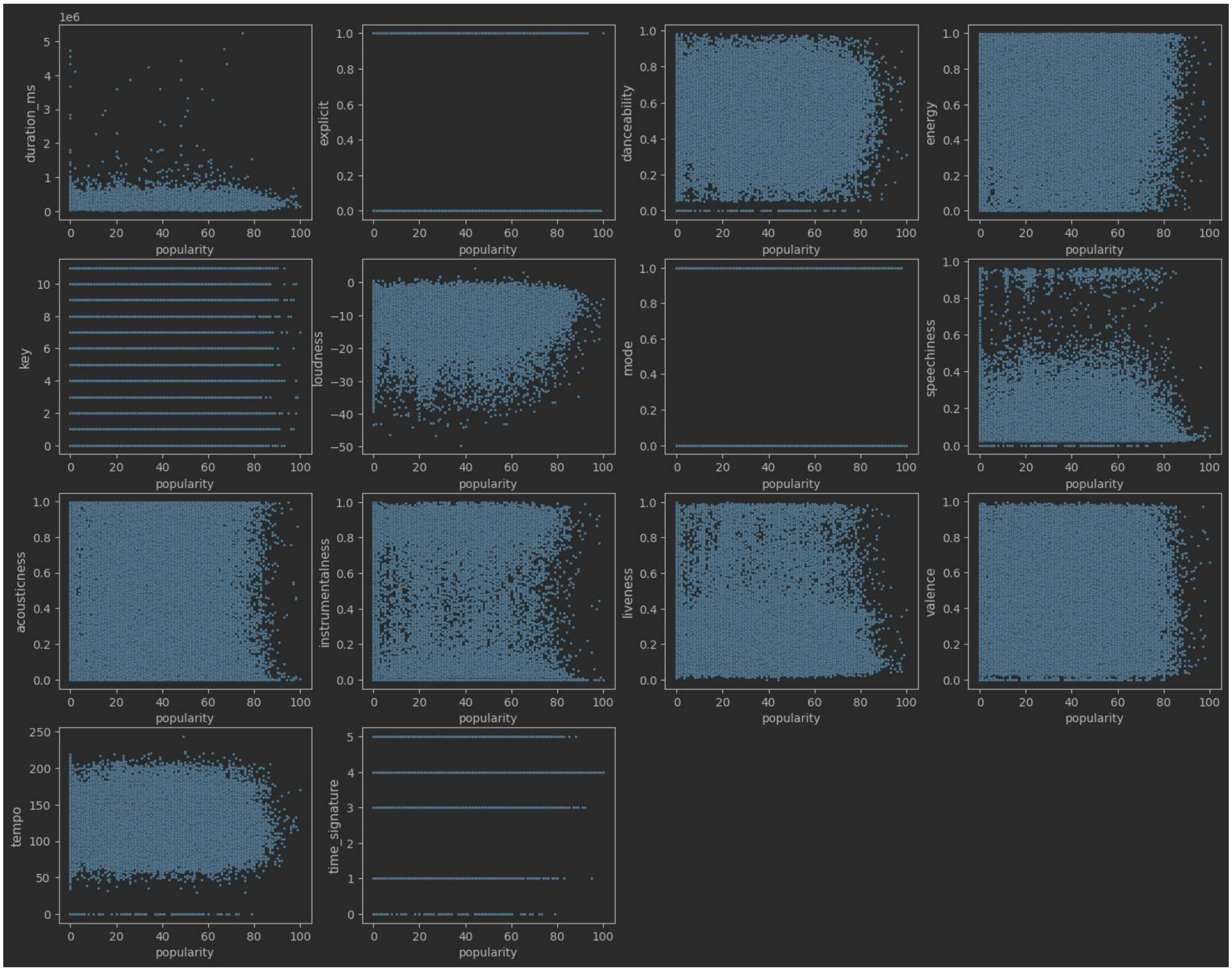


Figure 9.: Target variable correlation with other features

Based on this and the correlation matrix I did before I genuinely can not tell that popularity has a significant positive or negative correlation with any other features. Maybe loudness has the most significant and seeable positive correlation with popularity.

As my data is quite big and the distribution of the popularity scores are quite different for an easier version, I did a new column in my dataset where I made 10 classes based on the popularity scores in the dataset (you can see the terms and conditions of the classes in the python file).

So, I did a Linear Regression and Polynomial Regression for the original dataset popularity scores and for my new dataset where it contains these 10 classes based on the popularity score (*Figure 10.*).

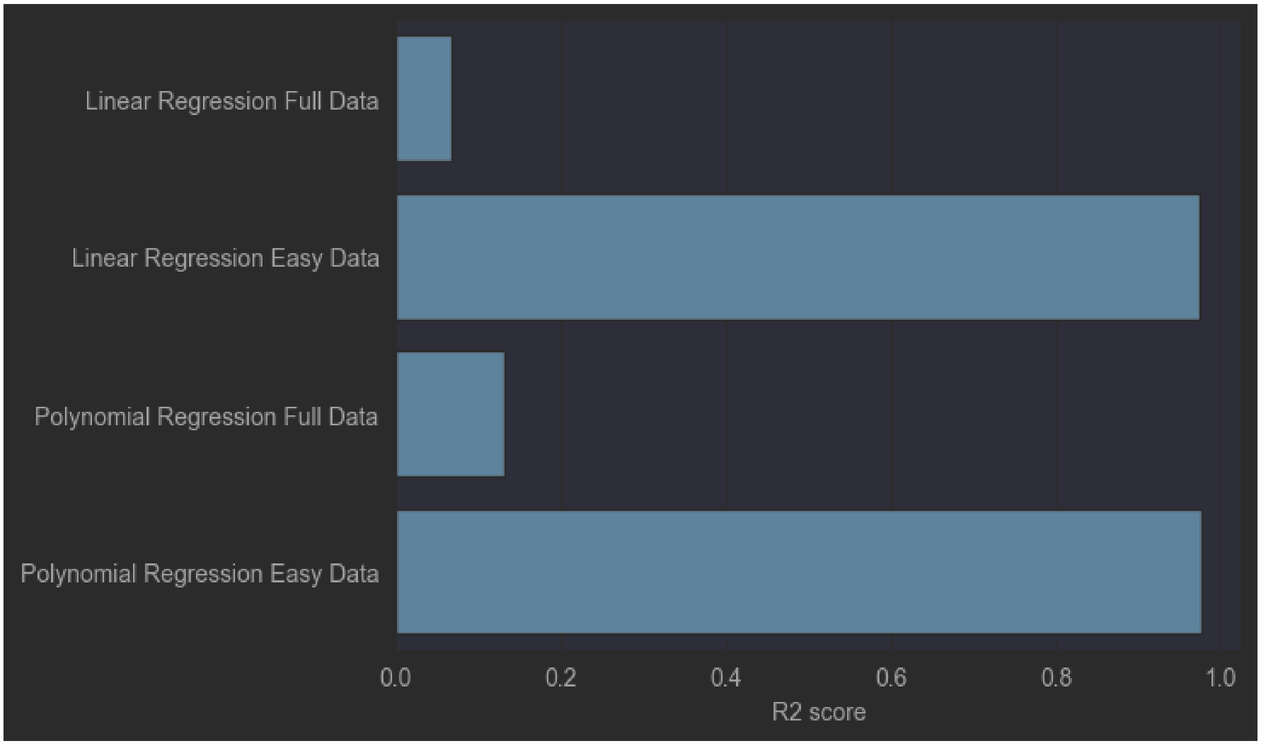


Figure 10.: R2 Scores of the models

As a result, we can definitely tell that the easier version I did for the dataset works way better than the original dataset popularity scores. Both for the Linear Regression and the Polynomial Regression has better R2 scores. The reason for this is my models in this way have to classify the test data into way less option then in the original dataset.

Name: Borbála Jakab
Neptun ID: YUBYNR

Genre-based Analysis: I made a dataframe where I collected the average values for each column for each genre (‘average’ dataframe). As it is a pandas dataframe I could easily open the whole datafram to have a look at the results.

For example, looking at the ‘danceability’ feature I could tell that most of the genres moves between 4-6 but there are a few outliers. For my surprise based on the average value ‘kids’ music is the most danceable despite the fact that there is a genre called ‘dance’ in the dataframe. However not so surprisingly ‘sleep’ genre has the lowest score (~1.7) in this feature (*Figure 11.*).

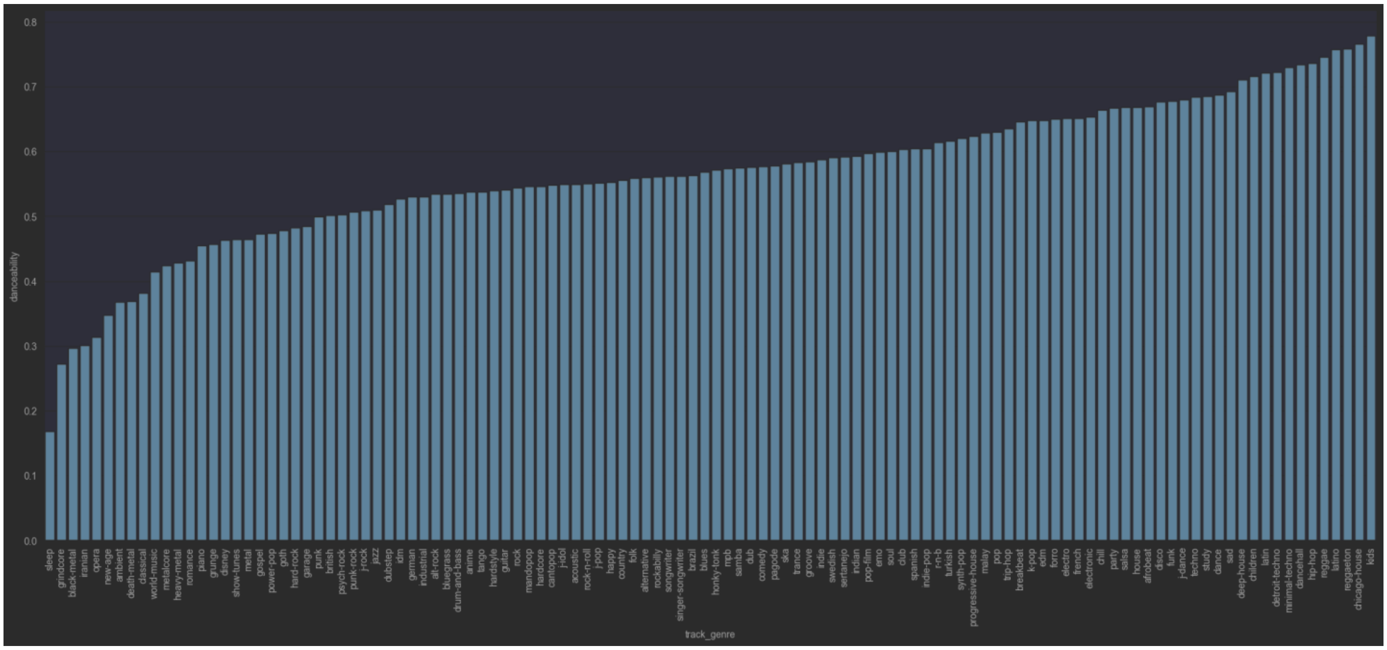


Figure 11.: Average value of ‘danceability’ across genres.

‘speechiness’ feature does not show really big of a distribution. Most of the genres moves between 0.03-0.15. Of course, we have some outliers here too but the most significant is that ‘comedy’ genre has a ~7.56 score here which is really jumps out from the other genres (*Figure 12.*).

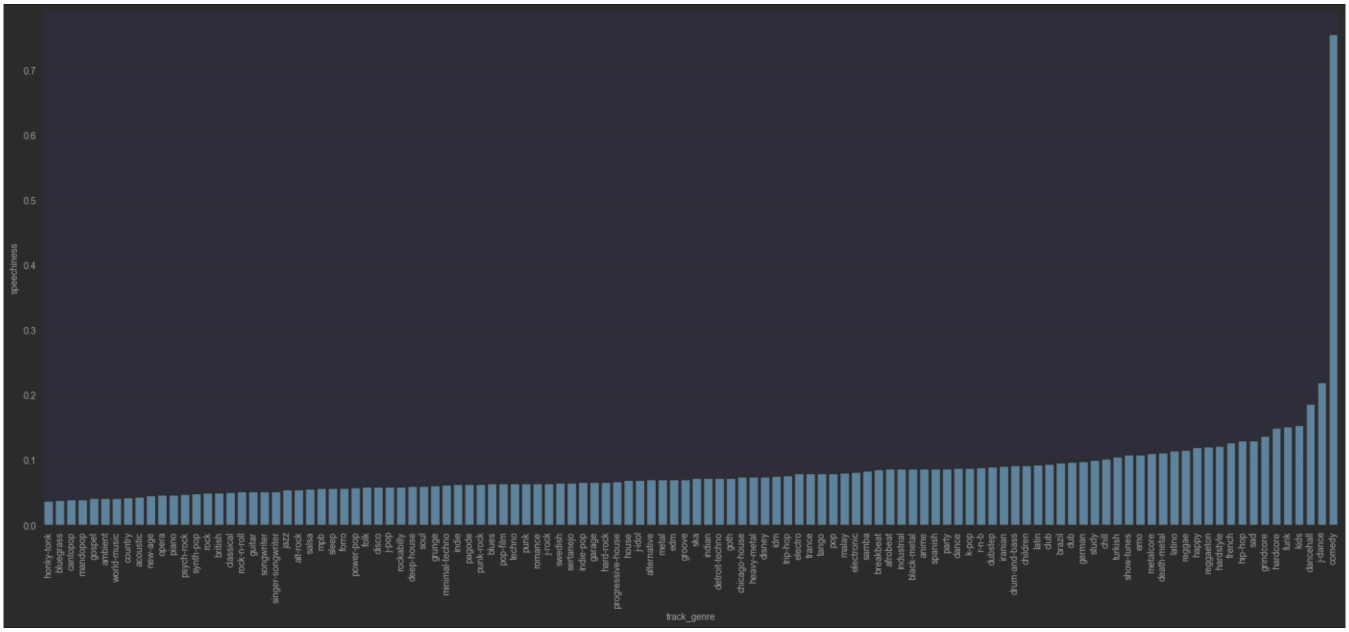


Figure 12.: Average value of ‘speechiness’ across genres.

Another feature ‘energy’ just like the danceability has quite a high range of distribution. Most of the genres moves between 0.4-0.7. On the top of this score, we can find the ‘death-metal’ genre with ~0.93 but surprisingly in the first 5 highest score we can find a few quite similar genre like ‘metalcore’, grindcore’ and ‘hardstyle’. With lowest scores we can find ‘classical’, ‘new-age’ and ‘ambient’. All of these genres about to create artistic inspiration, relaxation, and optimism so it is not so surprising that they have a low energy score (*Figure 13*).

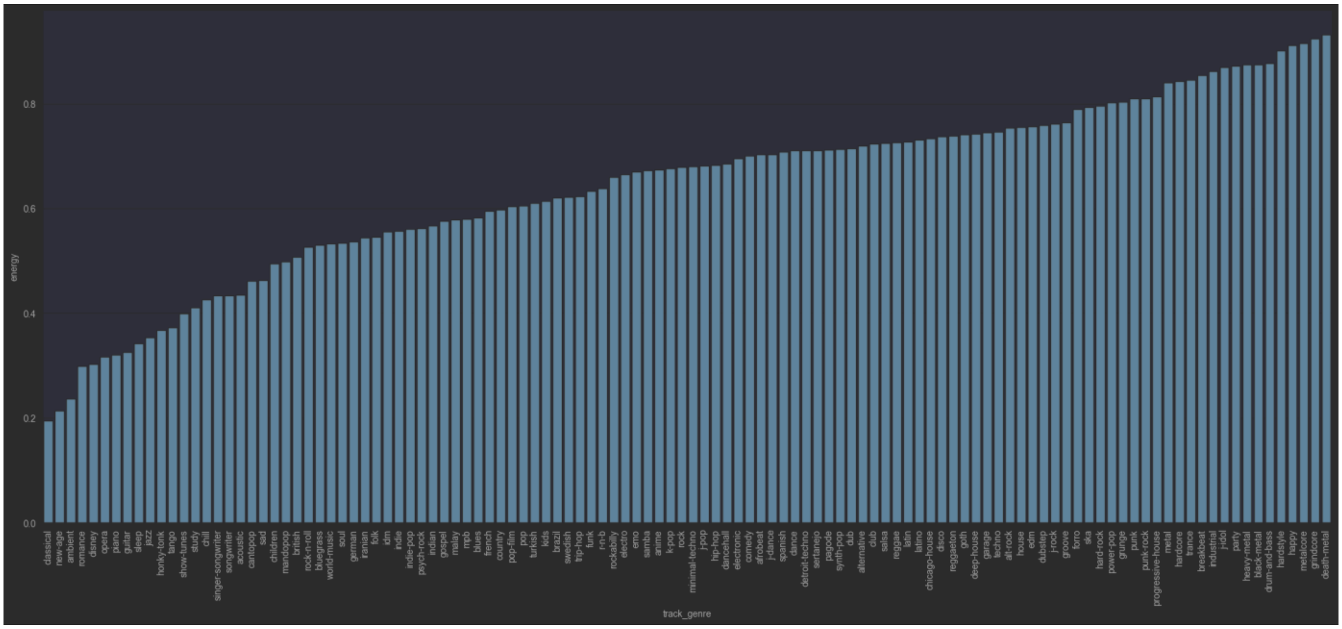


Figure 13.: Average value of 'energy' across genres.

I feel like 'acousticness' scores moves just the opposite as 'energy'. It is also highly distributed. Moves between 0.1-0.8 mainly. But here the highest score genres are 'classical', and 'romance'. But we can find 'new-age' and 'ambient' genres high on this list while 'death-metal', 'grindcore' and 'metalcore' are at the bottom of this list (*Figure 14.*).

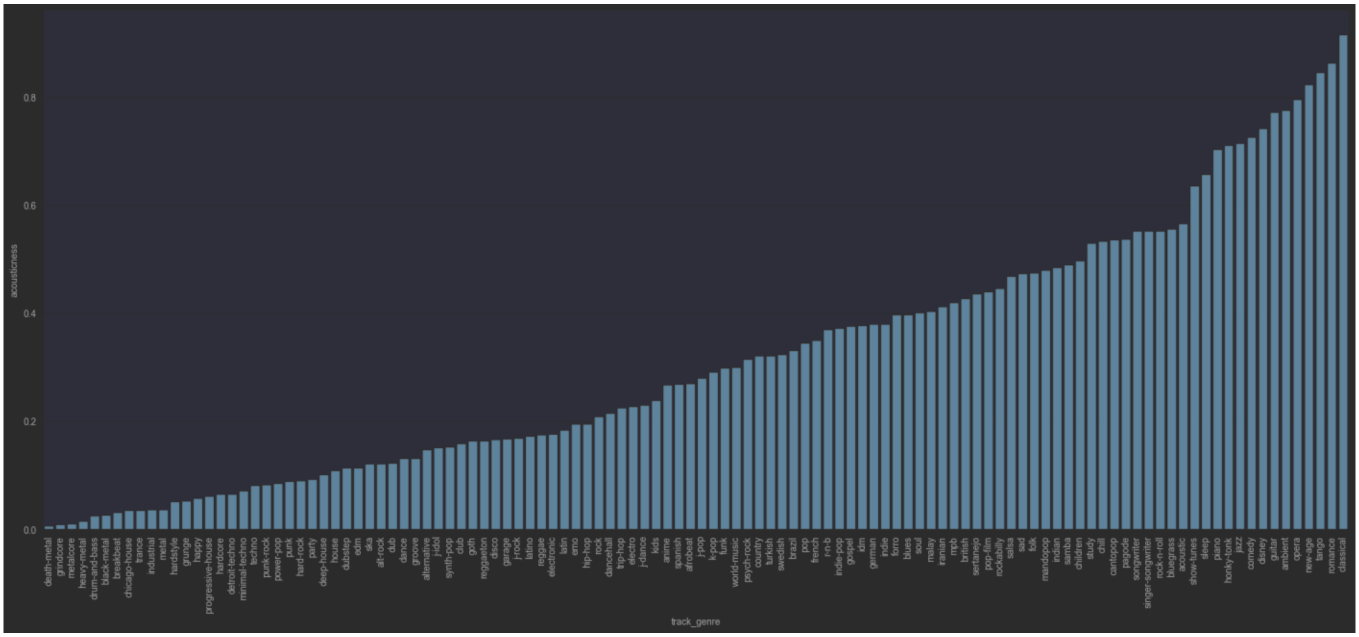


Figure 14.: Average value of 'acousticness' across genres.

Name: Borbála Jakab
Neptun ID: YUBYNR

Conclusion:

To start with for me personally I feel like this was a pretty big dataset to work with as I have never worked with such a large and complex dataset before. It contains a lot of different variables, rows and columns. However, I know that for other people it could be just a small everyday analysis.

During the data exploration, my dataset needed some manipulation as it contained missing values and duplicates in it but after every change I did it was still a large dataset.

The dataset I worked with overall showed pretty high distribution on a lot of different audio features like 'energy', 'danceability' or 'valence'. Not just simply their distribution but across the genres and also the average values of these audio features in every genre showed similarly high distribution. On the other hand, my data contained some audio features which have binary values like 'explicit' or 'mode'. The distribution of these were not so helpful in my work.

The correlation between the audio features only showed significant correlation in just a few pairs like 'loudness' and 'energy' where the correlation was positive and 'accusticness' and 'energy', and 'accusticness' and 'loudness' where the correlation was negative.

Looking at the average values of features like danceability, energy, and valence for each genre I got some surprising results. As for example the most danceable music based on the average values is 'kids' music. Or that in 'speechiness' there is not a really big distribution across the genres but there is one with a pretty high outlier compared to the other genres.

So overall the distribution of the variables is high and there are some pretty surprising results here.

As for the PCA it was important to normalize my dataset because without it my whole dataset was one main cluster on one of the axes but after normalization, I get 2 pretty separated cluster.

The song recommendation is a pretty useful function for this dataset. I feel like I made a function which is in my opinion works acceptable. I think it would be also fun to just make a function to look up songs with the same artist we are looking for. It could give us information if there is any other song from the artist we searched. If yes what kind of audio features, it has and also after this we could use the recommendation function if we want to find something similar to that song. Or by using the recommendation function we would look up similar song but for this function the input would not be a song title but an artist. So we would look up that artist, what kind of song he or she makes and based on that we would recommend an another artist who makes similar song.

The hardest part of this task for me definitely was the classification of the songs. I feel like that the models I made does not work well despite the fact that I feel like a tried a lot of things to make it work acceptable it is still not good.

One of the reasons why I think this classification was hard is the nature of my dataset. As I started working with it, I saw that there are songs which are identical only their genre is different. But their audio features are totally the same. So, from this it comes that the classification could be really hard as my model has 2 identical songs with identical audio features but then what should be the genre as it can be different. And sometimes these 2 genres are not even similar to each other so we would think based on this that probably the audio features also should not be similar, but they are the same.

Also, a lot of times song in the same music genre have really different audio features so this also makes the classifications hard.

Maybe the models could be better if I would use less audio features. If I would use only those which are significantly different in the genres. I think that could make the models better. Or maybe working with even bigger data.

As I mentioned earlier, the distribution in my data is quite high in a lot of audio features and in the popularity scores. So, predicting the popularity score based on the song's audio features were also a difficult task. Popularity scores overall did not really have correlation with any other audio features so predicting it based on them were hard. For the easier dataset I made it worked pretty well but for the original dataset my model is pretty bad. But at the end the Polynomial Regression got better R2 scores for both datasets.

I personally think these tasks were really interesting and challenging for me who did not work with these kinds of classifiers before. As I mentioned before I think it would be interesting to make a new recommendation system where the input is just an artist. We would look up that artist, what kind of song he or she makes and based on that we would recommend another artist who makes similar song.

Name: Borbála Jakab
Neptun ID: YUBYNR

Or I personally listen to music when I learn. It is just some background sounds which helps me concentrate. I think it would be awesome to make a spotify playlist based on specific audio features which are calm and relaxing which could be perfect for some background noise for studying. Or with other features a playlist which is perfect for running. So build a recommendation system based on some features we give and as an output we would get a playlist.