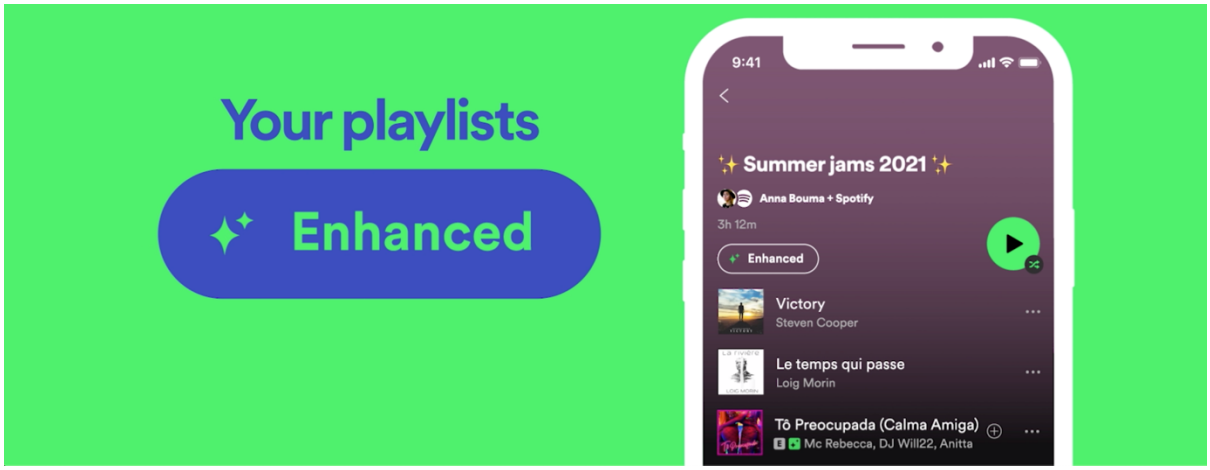
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ
SCHOOL OF BUSINESS
MSc IN BUSINESS ANALYTICS

Machine Learning & Content Analysis

Spotify Playlist Recommendation



Kostas Charamaras – P2822136
Argyris Oikonomopoulos – P2822126

Athens, 3/09/2022

Table of Contents

1. Introduction

Music is one of the most enjoyable activities and it can influence people of all ages and cultures around the world. Nowadays on of the most popular ways to listen to music is thru the online audio streaming services. Spotify is one of the largest audio streaming providers founded on 23 April 2006 by Daniel Ek and Martin Lorentzon in Sweden. Currently Spotify has with over 433 million monthly active users, including 188 million paying subscribers and over 80 million tracks.

To create a cool and creative playlist can take a lot of time to search and find the right songs matching your mood and your liking. The scope of this assignment and our goal in general is for spotify users to be able to express their situation or mood within a sentence and in a few seconds with the use of Machine learning concepts create a playlist to enjoy.

2. Chapter 2

For the implementation of this project we are going to use two datasets you can find bellow:

- https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db
- https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?resource=download&select=tracks.csv

Both datasets include features such as track name, artist, popularity, release date, if the track is explicit or not, gerne, danceability, energy, key, loudness , mode, speechiness, acousticness, liveness, valence, tempo, duration, time signature.

More info about Spotify Metrics :

- **Danceability**: Describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
- **Energy**: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale
- **Key**: The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on.
- **Loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks.
- **Mode**: Indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

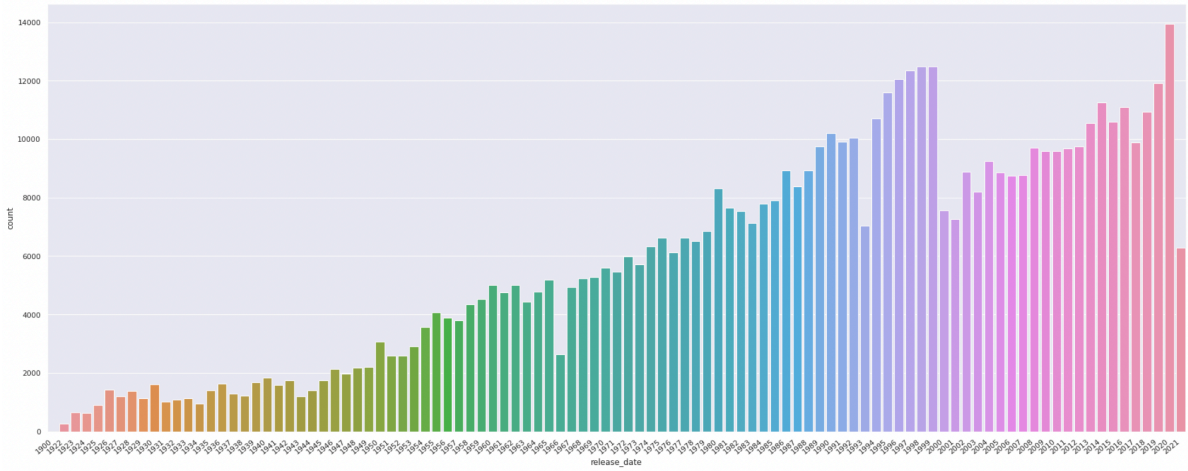
- **Speechiness**: This detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
- **Acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- **Instrumentalness**: Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”.
- **Liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
- **Valence**: Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
- **Tempo**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece, and derives directly from the average beat duration.
- **Duration**: The duration of the track in milliseconds.
- **Time Signature**: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

*Source: https://towardsdatascience.com/what-makes-a-song-likeable-dbfdb7abe404

2.1 Exploratory data analysis in Big dataset

In the first dataset we can see that we have 586672 records and 20 columns . We are going thru a cleaning process of the dataset removing null values (~ 0.012% of the dataset), fixing data types of the features removing unnecessary columns , removing brackets and commas from columns.

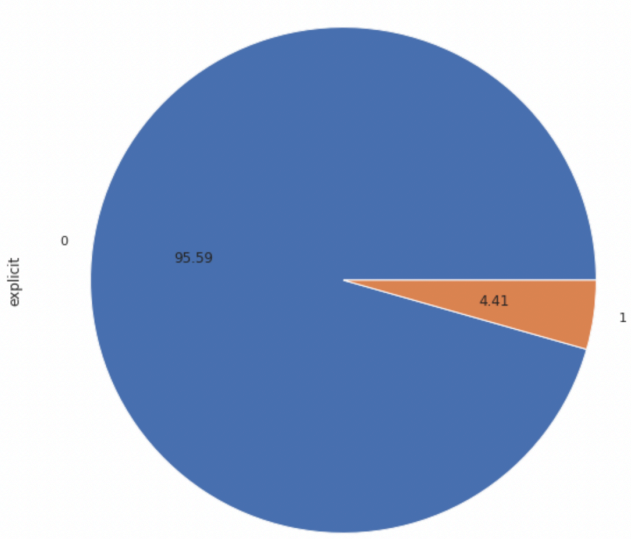
In the plot below we can see that the majority of the songs are after 1995, and 2021 having almost 40% of our dataset



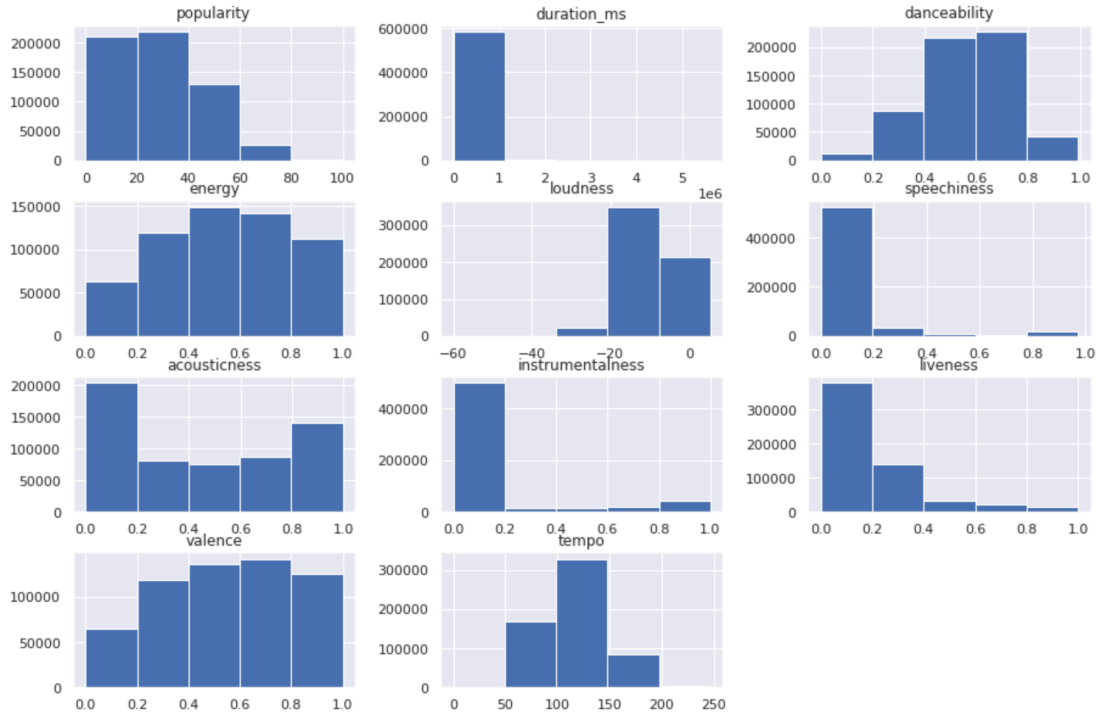
Most popular song in the songs Peaches by justin Bieber followed by drivers licence by olivia Rodrigo and others.

name	artists	popularity
Peaches (feat. Daniel Caesar & Giveon)	Justin Bieber, Daniel Caesar, Giveon	100
drivers license	Olivia Rodrigo	99
Astronaut In The Ocean	Masked Wolf	98
telepatía	Kali Uchis	97
Save Your Tears	The Weeknd	97
Leave The Door Open	Bruno Mars, Anderson .Paak, Silk Sonic	96
Blinding Lights	The Weeknd	96
The Business	Tiësto	95
Fiel	Los Legendarios, Wisin, Jhay Cortez	94
Bandido	Myke Towers, Juhn	94
Friday (feat. Mufasa & Hypeman) - Dopamine Re-...	Riton, Nightcrawlers, Mufasa & Hypeman, Dopamine	94
WITHOUT YOU	The Kid LAROI	94
Streets	Doja Cat	94
Heartbreak Anniversary	Giveon	94
LA NOCHE DE ANOCHE	Bad Bunny, ROSALÍA	93
Good Days	SZA	93
Paradise (feat. Dermot Kennedy)	MEDUZA, Dermot Kennedy	92
DÁKITI	Bad Bunny, Jhay Cortez	92
Watermelon Sugar	Harry Styles	92
positions	Ariana Grande	92

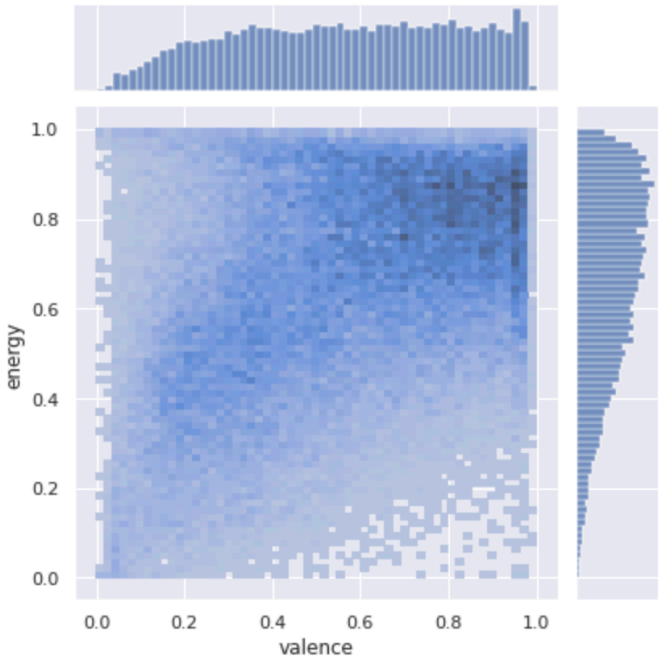
Also almost all of our songs are clear without insulting and swear words.



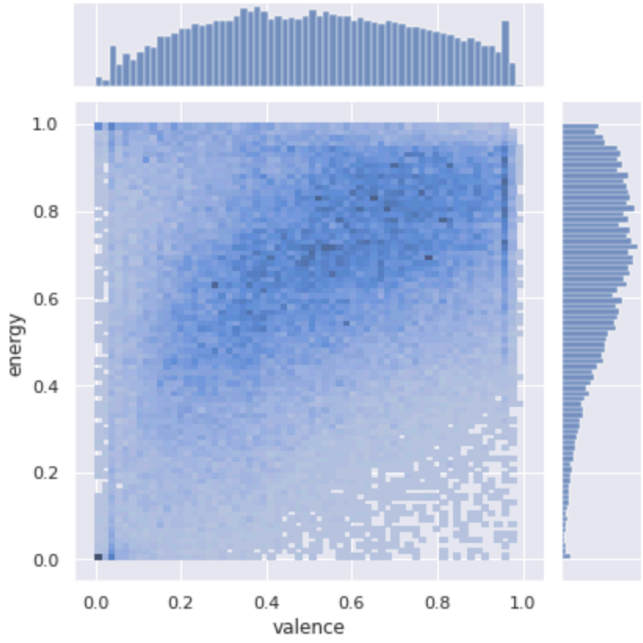
Below we can see the histograms from the spotify features. Speechiness and instrumentalness are heavily skewed left so most likely we will not use the for prediction.



Joint plot energy and valence from 2000 until 2010



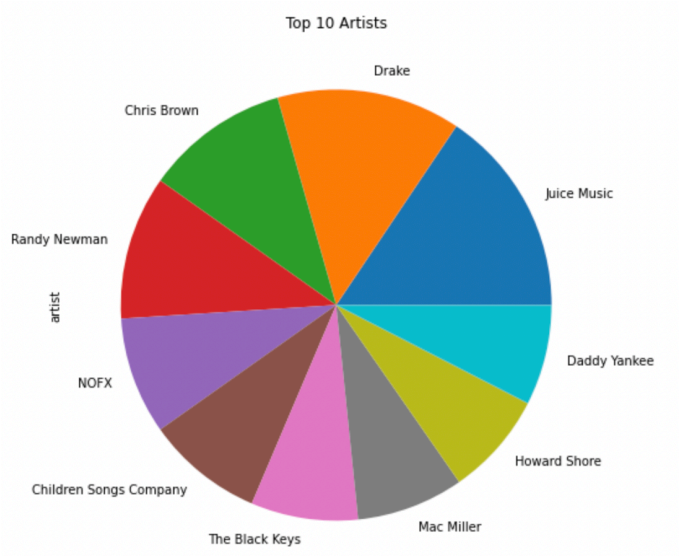
Joint plot energy and valence from 2011 until 2021



From the two jointplots above we can see that the energy level from both decades remains the same , valence from 2011 and after seem to have lower values so we have songs that are less happy and are more sad.

2.2 Exploratory data analysis Lyrics Dataset

In the lyrics dataset we condact about the same analysis. Below we can see the artist we the most tracks in the dataset



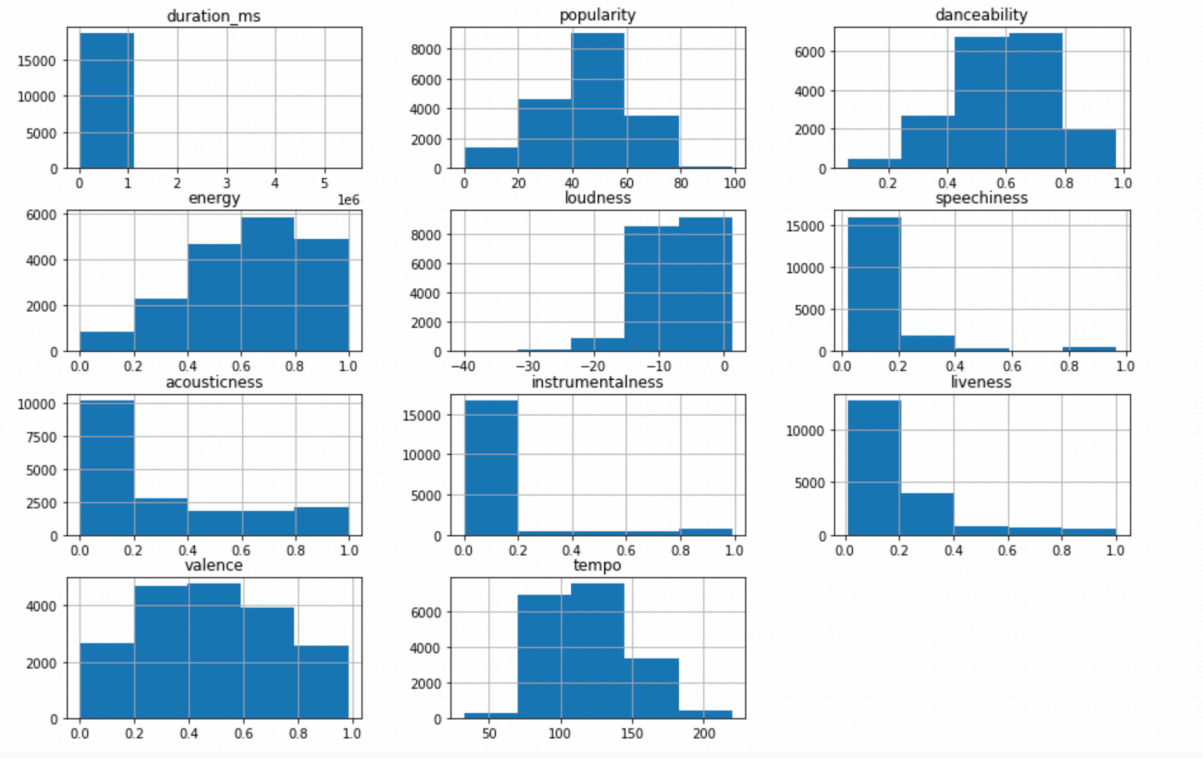
Most popular songs

track_name	artist	popularity
Wow.	Post Malone	99
Sweet but Psycho	Ava Max	97
ZEZE (feat. Travis Scott & Offset)	Kodak Black	93
Drip Too Hard (Lil Baby & Gunna)	Lil Baby	92
NASA	Ariana Grande	91
Don't Call Me Up	Mabel	91
Sucker	Jonas Brothers	91
Adictiva	Daddy Yankee	91
Arms Around You (feat. Maluma & Swae Lee)	XXXTENTACION	90
no tears left to cry	Ariana Grande	90
Beautiful (feat. Camila Cabello)	Bazzi	89
idontwannabeyouanymore	Billie Eilish	89
Splashin	Rich The Kid	88
Havana	Camila Cabello	88
Shotgun	George Ezra	87
Saturday Nights	Khalid	87
Electricity (with Dua Lipa)	Silk City	87
Ya No Tiene Novio	Sebastian Yatra	87
Te Boté - Remix	Nio Garcia	87
I Love It (& Lil Pump)	Kanye West	87

Popular gerne

```
Indie                996
Hip-Hop              972
Alternative          966
Rap                  950
Folk                 930
Children’s Music     929
Pop                  929
Rock                 926
R&B                  905
Soul                 887
Dance                859
Country              854
Blues                853
Reggaeton            812
Ska                  794
Reggae               761
Electronic           709
World                692
Jazz                 614
Comedy               595
Anime                448
Movie                435
Children's Music     389
Soundtrack           205
Classical            147
Opera                 66
A Capella              5
Name: genre, dtype: int64
```

Same as the big dataset we get histograms of spotify features and again we see speechiness and instrumentalness are heavily skewed right.



2.3 Fetch lyrics from genius

To get the lyrics from tracks of the lyrics dataset we connected to genius api and fetch lyrics that we available. We requested from genius 25000 track lyrics and we end up getting 18628. This process takes very long time to complete , it took as around 17 hours.

```
Done.
Searching for "Red Rooster (with Henry Gray and Howlin' Wolf Band) - Live" by Kenny Wayne Shepherd...
100%|██████████| 24998/25000 [16:48:17<00:03,  1.90s/it]
No results found for: 'Red Rooster (with Henry Gray and Howlin' Wolf Band) - Live Kenny Wayne Shepherd'
Searching for "Isabelle" by Kamasi Washington...
100%|██████████| 24999/25000 [16:48:18<00:01,  1.59s/it]
Specified song does not contain lyrics. Rejecting.
Searching for "Around the World" by Yo Gotti...
100%|██████████| 25000/25000 [16:48:20<00:00,  2.42s/it]
Done.
CPU times: user 27min 18s, sys: 2min 11s, total: 29min 29s
Wall time: 16h 48min 20s
```

2.4 Data Processing

The original data from the lyrics dataframe which was used to train the RNN models needed some processing in order to be trainable-ready for our models. Firstly, only 6 audio features were kept from the dataset (energy,popularity,danceability,valence, acousticness, liveness), because these could be produce some actual insights by using the lyrics as inputs, and because they seemed to be more smoothly distributed across. For example, it would not make any sense to predict the tempo of a song based on its lyrics. Also, for instance Spotify's audio feature speechiness was heavily negatively skewed (~95% 0-0.2) and thus could not produce any interesting prediction - regardless of the training of the models.

At this point it is worth noting that the numeric data of the datasets needed to be scaled, because later on we will use the Euclidean distance.

Then we can start to process the lyrics' text. As can be seen in the clean_text function in the ML part notebook, at first, we lowered-case all the lyrics' text, then removed all the punctuations and special characters. Finally, we removed the basic English stopwords. Then, we splitted the data into train validation and test proportions, accordingly.
Now, we can use the TF-IDF method and fit the according vectorizer to the lyrics data.

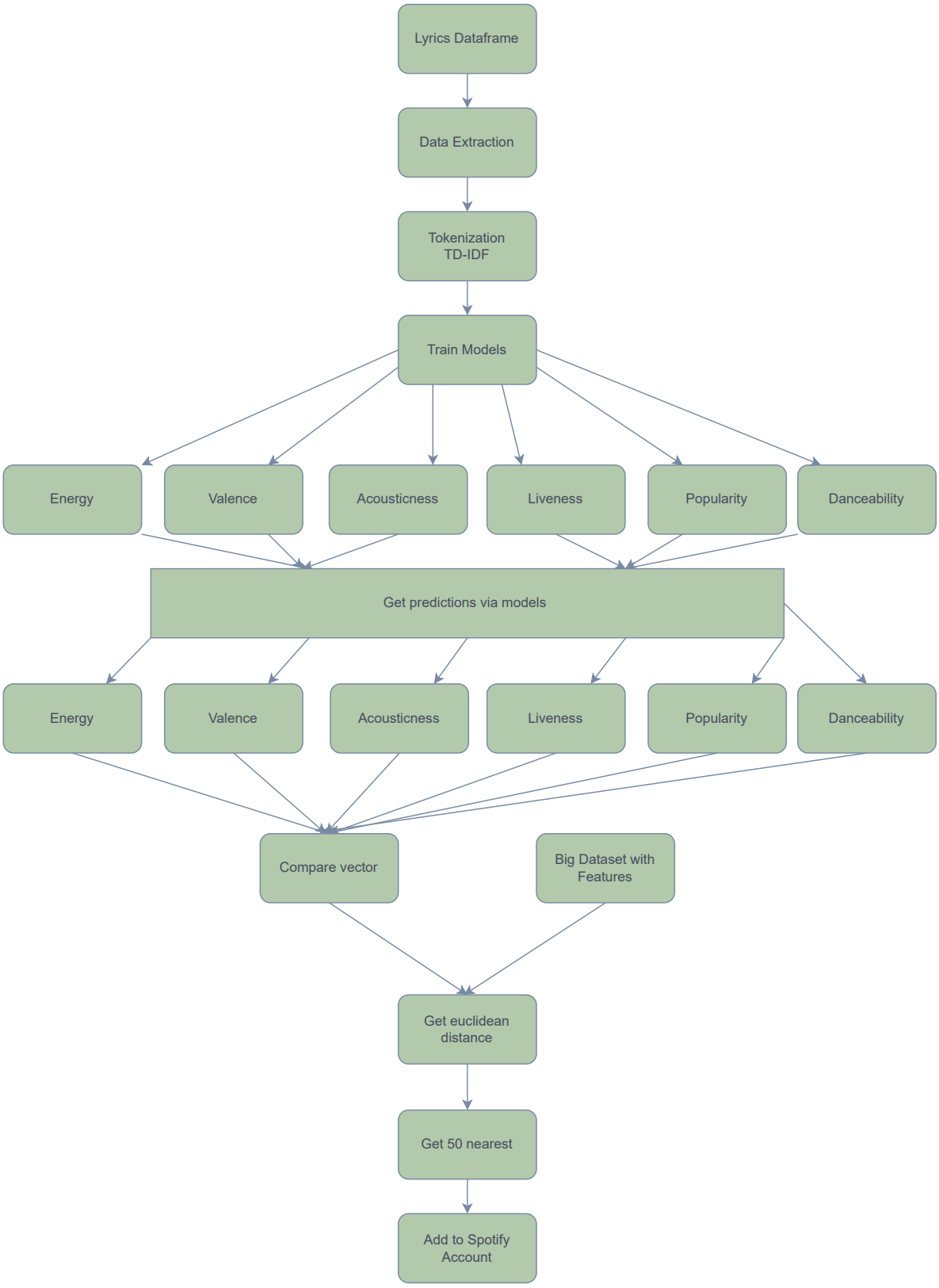
2.5 Methodology

The project is based on the idea that we can train RNN models to predict a Spotify audio feature based on an input text. In order to train these models, we fetched the lyrics of 18628 tracks (the process can be found on Get-Lyrics-ML-Assignment.ipynb), to act as the inputs. After some text processing (i.e. removing stop words/lowering characters etc.) we then can extract these text data, and pass it through a TF-IDF tokenizer, in order to transform them into a matrix of TF-IDF features.

Now, we can train our models (2 models with different optimizers (Adam, SGD) for each audio feature energy,popularity,danceability,valence, acousticness, liveness). After the training and the assessment of the performance of each model, we can use the best ones to predict each audio feature, based on an input text (the one given by the user).

At this point, we can store the 6 predicted audio feature values, into one 6 position vector. This vector can run through the big dataset, (the pool of already fetched ~600k tracks) and compute the euclidean distance from the in-place audio-feature values. Consequently we can find the 50 nearest songs from the text-from-user derived vector, and we can finish the whole process by adding these 50 tracks to a newly created playlist in Spotify. This can be done by utilizing Spotify's API.

Project WorkFlow



3. Chapter 3

3.1 Algorithms, NLP architectures/systems

We are going to use Keras Sequential models for predicting each audio feature. Since the audio features are continuous numbers, ultimately we want to implement a linear regression for each one of them, and we can measure how well the models predict, with the use of Mean Squared Error.

We are going to train 2 models for each audio feature, the one will run with Adam optimizer, and the other one with Stochastic Gradient Descent (SGD) optimizer. Then we can choose which one works better for each feature and use it to continue. For our models, we use 2 Dense layers of size 256 respectively, with relu activation, and the final layer is a Dense layer of size one and linear activation, since we are trying to solve a linear regression problem. As mentioned before, we calculate the loss between each epoch by the MSE produced between the train and the validation sets.

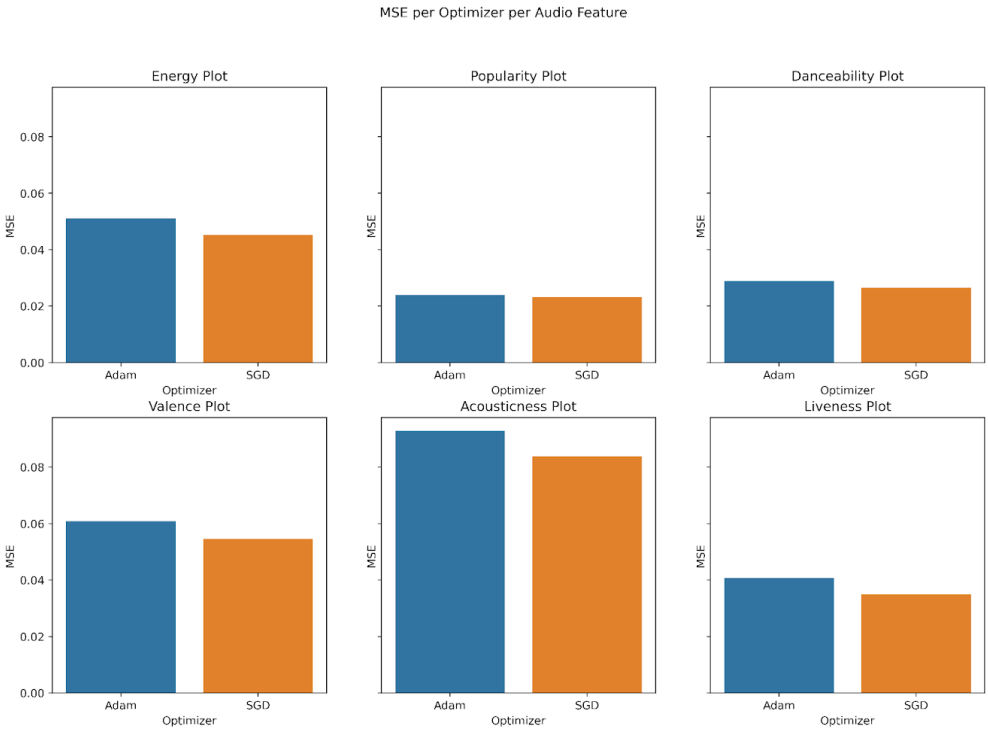
3.2 Experiments – Setup, Configuration

In this step, we want to setup our prediction pipeline. As mentioned before, we want to get the predicted values of the 6 audio features from the respective best-performing models, and put them in a vector which will be compared to the pool of tracks to find the 50 nearest.

In order to do so, we can give an input text to the algorithm, for example, in the attached .ipynb we gave 'Today I feel great, went for a walk in the park and then came back home to listen to my favorite songs and to drink a glass of wine' as the input text. Then this text was tokenized with the same TF-IDF tokenizer we used before, and was fed to the 6 SGD models that we found that were more accurate with their predictions (more on this later). Then the 6-position vector ran through the pool dataframe to find the 50 nearest tracks, in terms of Euclidean Distance. It is worth noting here, that because of the large size of the pool dataset, we saw that many songs that were finally selected, were rather unpopular and would not give a value to the user. To combat this, we only selected these with popularity greater than 0.5 (greater than the median). Also, we gave a little bit more weight towards the popularity of the song than the similarity of the songs selected (0.9*popularity+0.8*similarity). With this way we could produce more pleasing songs for the user. Finally, we give the user the ability to name the auto-generated playlist and enjoy it on her/his Spotify account.

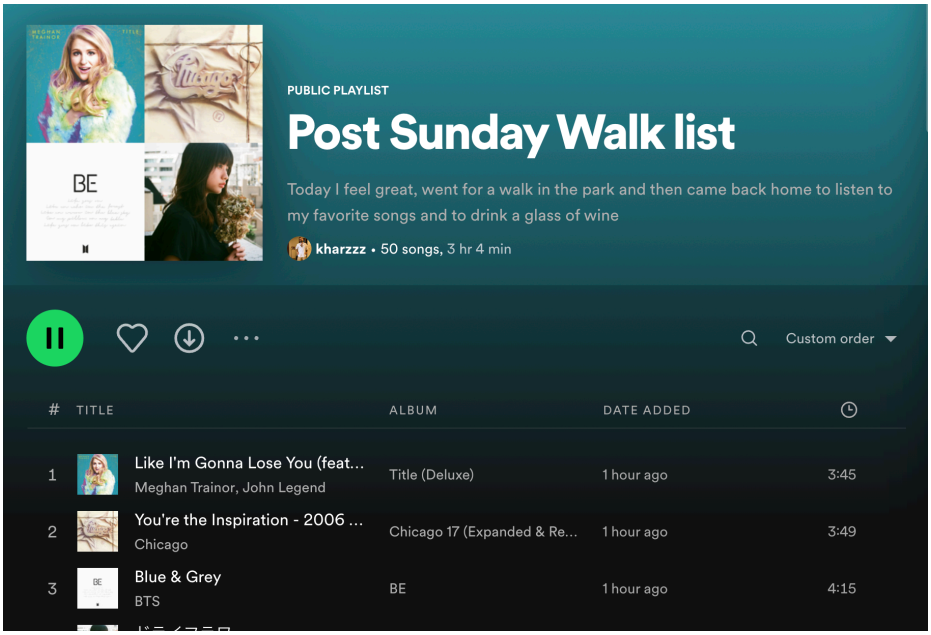
3.3 Model Performance

Regarding the predicting ability of the models, we can see that for 5 out of 6 models, SGD was better than Adam optimizer, in terms of MSE, and thus was used for the project. For the popularity model, Adam and SGD were equal at 2%.



4. Results

After giving the input text we mentioned before ('Today I feel great, went for a walk in the park and then came back home to listen to my favorite songs and to drink a glass of wine'), we got the results. As can be seen, a new playlist was created in the Spotify account, which consists of 50 songs that 'describe' the feeling of the user's text.



5. Discussion, Comments/Notes and Future Work

In this project, we created an automatic playlist creation tool that can be very helpful for the user. The use of RNNs for a regression problem produced very pleasing results.

After the completion of this project, a lot of insightful comments can be made.

- The use of TF-IDF is extremely handy, it can be more efficient than a simple Bag of Words approach, as it measures frequency and assigns weights to words/tokens, but it leaves something to be desired in terms of context and semantics capturing. Maybe word embeddings with Word2Vec or even better the use of some pretrained models (BERT, distilBERT etc.) can be more fruitful. For us, we experienced some technical issues with using Transformers and thus, opted to get the job done with the TF-IDF method which still produces very good results.
- For the biggest part of this project, the data we used is static. In the future it would be interesting to use dynamic data (maybe provided by the user's music taste - with the help of Spotify's API or something similar for any other streaming platform). This could produce a more tailor-made product for the user, and thus provide a more pleasant result for her/him.
- Country/Market specialization would've made sense in a project like this, but there was not such available data to get our hands on.
- Another idea which could've made sense is the combination of lyrics with other song details (e.g. song waveform etc.) to train models. Unfortunately, there are not lots of such data free for us to get at the time, and thus it's really difficult to train models with such a small sample.