PLAYLISTS RECOMMENDATION SYSTEM & CLUSTER OF SONGS

G-13 : Luca Avitabile, Marco Colombi, Simone Marretta, Luca Scofano, Daniele Trappolini.

The main goal of our project is to create a playlist Recommendation System using the Spotify's API.

Our Recommendation System is able to return the playlist with the highest number of followers given a certain genre and number of songs.

BONUS PART!

We also implemented a Clustering that is able to cluster a song given its features taken from the Spotify's API.

How the work has been divided:

- *Data Collection*
- *Data Preprocessing*
- *EDA*
- *Regression Models*
- *Recommendation System*
- *Clustering*

Let’s move to the first part...

DATA COLLECTION WITH Spotify API

… IN FEW STEPS

1. Get Authentication token for API
2. Sample Spotify playlists
3. Store playlists with attributes in a file

4. Get track info from the sample of playlists

5. Store track infos in a json file

DATA PREPROCESSING

... IN FEW STEPS

Spotify®

1. Load file coming from data collection

2. Extract track features

3. Build playlists dataframe

4. Build tracks dataframe

5. Save dataframes in 2 csv files

Regression Models

Goal: Find the optimal Regression Model to predict the log number of followers of a playlist

According to:

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS} \quad \& \quad RMSE = \sqrt{\frac{1}{N}(Y_{obs_i} - Y_{pred_i})^2}$$

Data Preprocessing

We transformed our target variable by computing its log value: so our actual predict variable is the logarithm of the number of followers.

We also converted our categorical variables (genre, track time signature mode, track key mode) using the *One-Hot Encoding* approach.

After that, we scaled the variables left with a "*Standard*" approach: we subtracted to each variable its mean and then we divided the difference by its standard deviation.

MODELS:

1. ***Linear Regression***
2. ***Ridge***
3. ***Lasso***
4. ***Random Forest***
5. ***Support Vector Machine (SVM)***
6. ***AdaBoost***
7. **Particular Models:**
   - ***Ensemble Model*** with the average predictions of all the models above
   - ***Ensemble Model*** with the average predictions of the best three models above
   - ***Meta Model*** using a Linear Regression that takes in input the predictions of all the models above
   - ***Meta Model*** using a Linear Regression that takes in input the predictions of the best three models above

Model	*R-Squared*	*RMSE*
Linear Regression	0.2022	3.5819
Ridge Regression	0.2042	3.5774
Lasso Regression	0.2035	3.5790
Random Forest	**0.4512**	**2.9706**
Support Vector Machine	0.3891	3.1343
AdaBoost	0.3929	3.1247
Ensemble - All Models	0.3867	3.1405
Ensemble - RF+SVM+AB	0.4719	2.9139
Meta Model - All Models	0.4939	2.7955
Meta Model - RF+SVM+AB	**0.4943**	**2.7942**

Adding Interaction Terms

Now we consider a new dataset.

We add the Interaction Terms between:

- the ***Genre***
- and the ***Audio Features***

as suggested by our EDA.

Interaction Terms

=

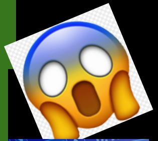
One Hot Encoded Genre Columns

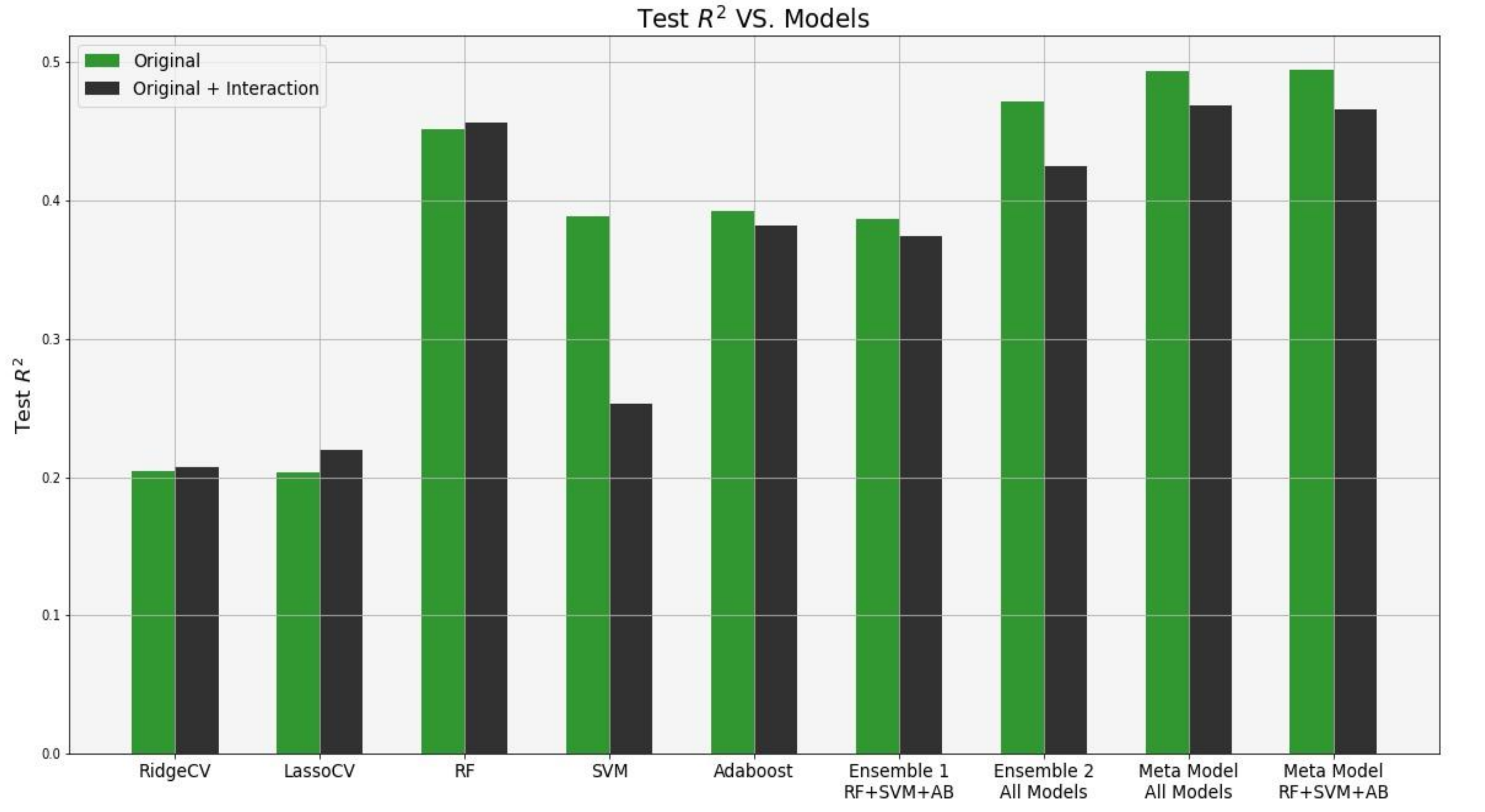
x

Audio Features Columns

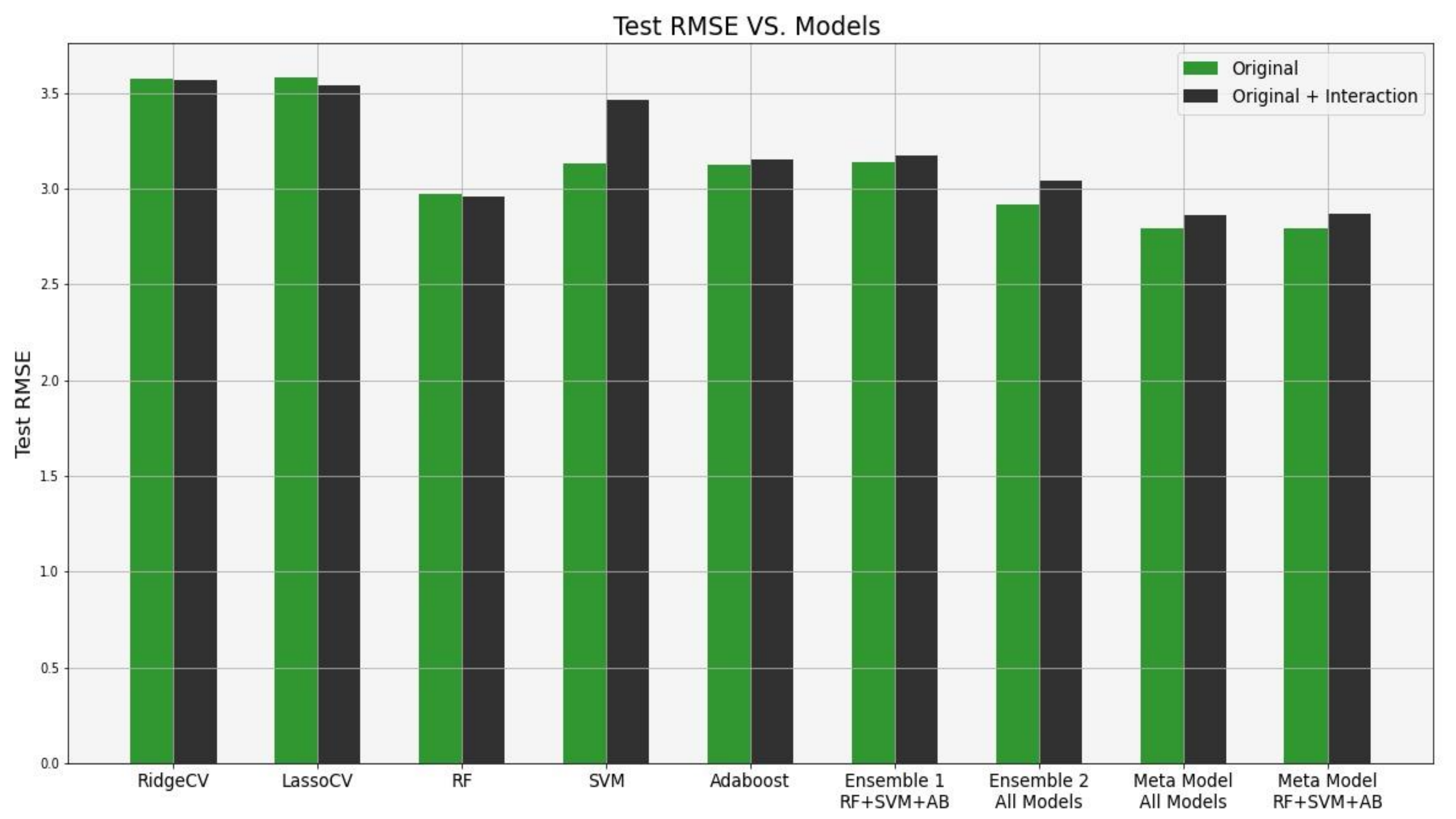
Another time, we train our models with this new dataset and we will see what is going to happen.

Model	***R-Squared***	***RMSE***
Linear Regression	-6.7799	104416949781.1035
Ridge Regression	0.2072	3.5704
Lasso Regression	0.2195	3.5428
Random Forest	**0.4566**	**2.9559**
Support Vector Machine	0.2533	3.4651
AdaBoost	0.3819	3.1527
Ensemble - All Models	0.3745	3.1715
Ensemble - RF+SVM+AB	0.4246	3.0418
Meta Model - All Models	***0.4686***	***2.8645***
Meta Model - RF+SVM+AB	0.4661	2.8712



Test R^2 VS. Models
Original
Original + Interaction
Test R^2
0.0
0.1
0.2
0.3
0.4
0.5
RidgeCV
LassoCV
RF
SVM
Adaboost
Ensemble 1
RF+SVM+AB
Ensemble 2
All Models
Meta Model
All Models
Meta Model
RF+SVM+AB



Test RMSE VS. Models
Test RMSE
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Original
Original + Interaction
RidgeCV
LassoCV
RF
SVM
Adaboost
Ensemble 1
RF+SVM+AB
Ensemble 2
All Models
Meta Model
All Models
Meta Model
RF+SVM+AB

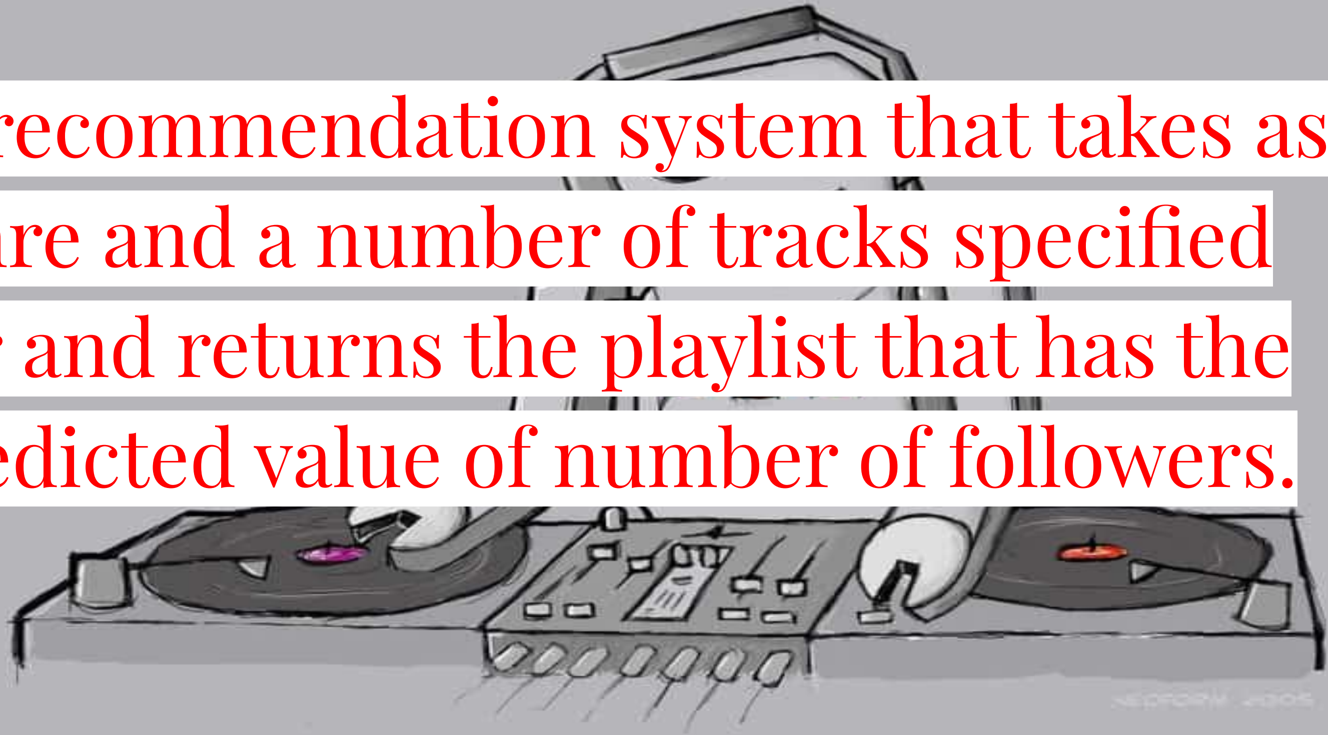
Conclusions (about Regression Models)

In the end the Interaction Terms didn't perform well in order to improve the score of our models

So for our recommendation system, we will consider the second Meta Model with our original dataset

MUSIC RECOMMENDATION SYSTEM

OUR GOAL



We built a recommendation system that takes as input a genre and a number of tracks specified by the user and returns the playlist that has the highest predicted value of number of followers.

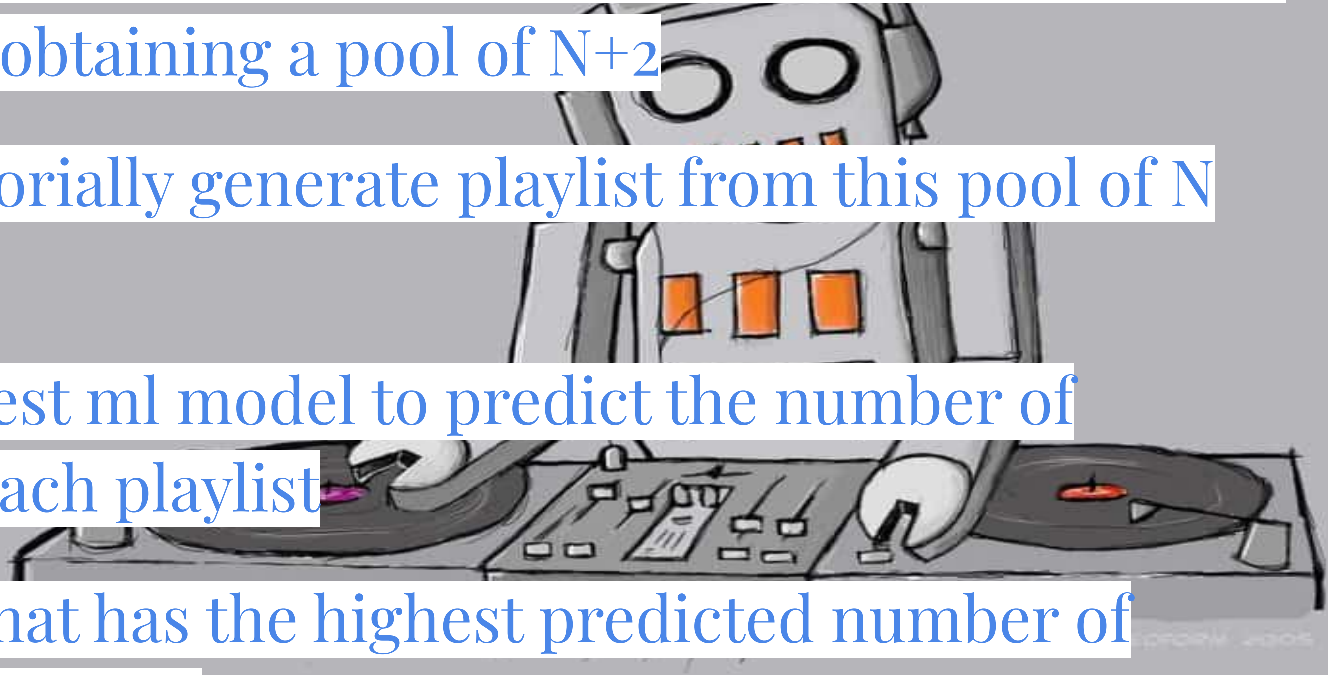
What the system does

-We retrieve the tracks with the same genre and filter them by popularity obtaining a pool of N+2

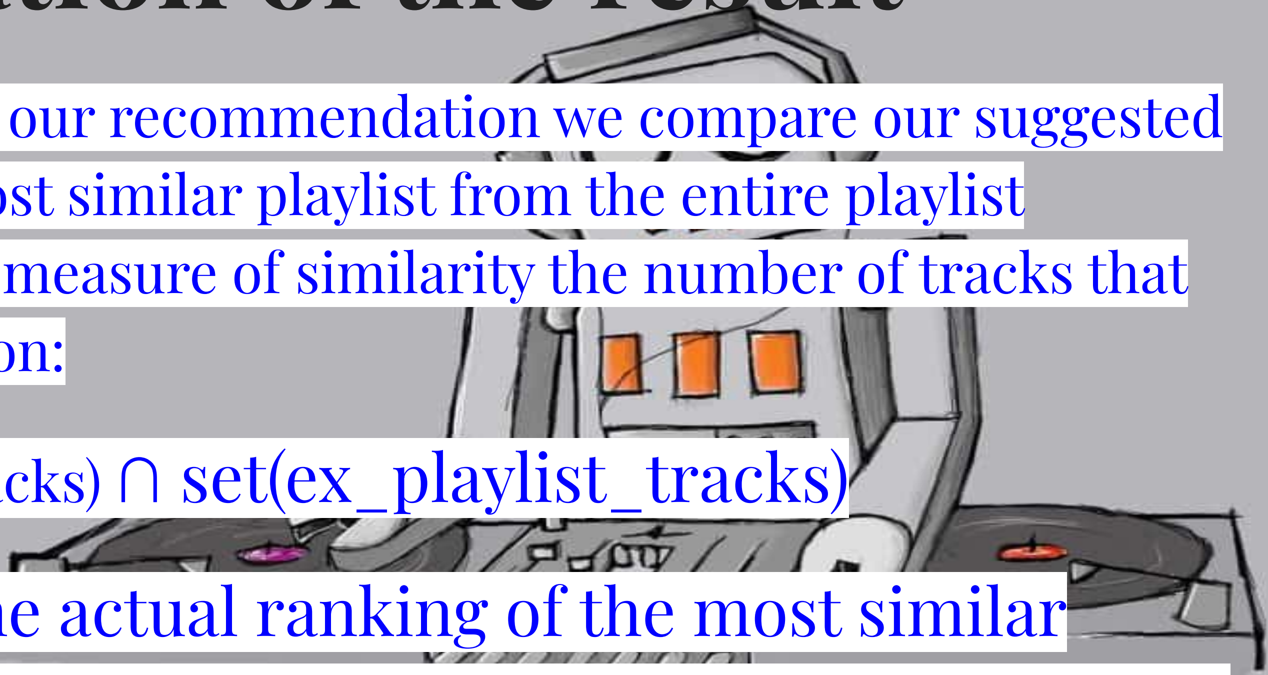
-We combinatorially generate playlist from this pool of N tracks

-We use our best ml model to predict the number of followers for each playlist

-The playlist that has the highest predicted number of followers is returned.



Validation of the result



In order to validate our recommendation we compare our suggested playlist with the most similar playlist from the entire playlist database.We use as measure of similarity the number of tracks that they have in common:

set(rec_playlist_tracks) ∩ set(ex_playlist_tracks)

Finally we see the actual ranking of the most similar playlist in our playlist database.An high rank within genre means that the recommendation was good.

Some results:

```
tracks that are missing : 0
The recommended playlist is:
```

	track ID	track name	artists names
0	4Oun2ylbjFKMPTiaSbbCih	WAP (feat. Megan Thee Stallion)	[Cardi B, Megan Thee Stallion]
1	6UelLqGlWMcVH1E5c4H7lY	Watermelon Sugar	[Harry Styles]
2	1xQ6trAsedVPCdbtDAmk0c	Savage Love (Laxed - Siren Beat)	[Jawsh 685, Jason Derulo]
3	2XU0oxnq2qxCpomAAuJY8K	Dance Monkey	[Tones And I]
4	3ZG8N7aWw2meb6Url5ZmnZ	Relación	[Sech]
5	4HBZA5flZLE435QTztThqH	Stuck with U (with Justin Bieber)	[Ariana Grande, Justin Bieber]
6	4xqrdfXkTW4T0RauPLv3WA	Heather	[Conan Gray]
7	45bE4HXI0AwGZXfZtMp8JR	you broke me first	[Tate McRae]
8	2ygvZOXrIeVL4xZmAWJT2C	my future	[Billie Eilish]
9	7qEHsqek33rTcFNT9PFqLf	Someone You Loved	[Lewis Capaldi]

```
Predicted num_followers: 372.0338844803864
[5.34970171 5.34970171 5.34970171 5.34970171]
The most similar playlist's predicted num_followers: 209.54548419235527
There are 2626 playlists in genre = pop
The most similar playlist's rank within genre is: 5
```

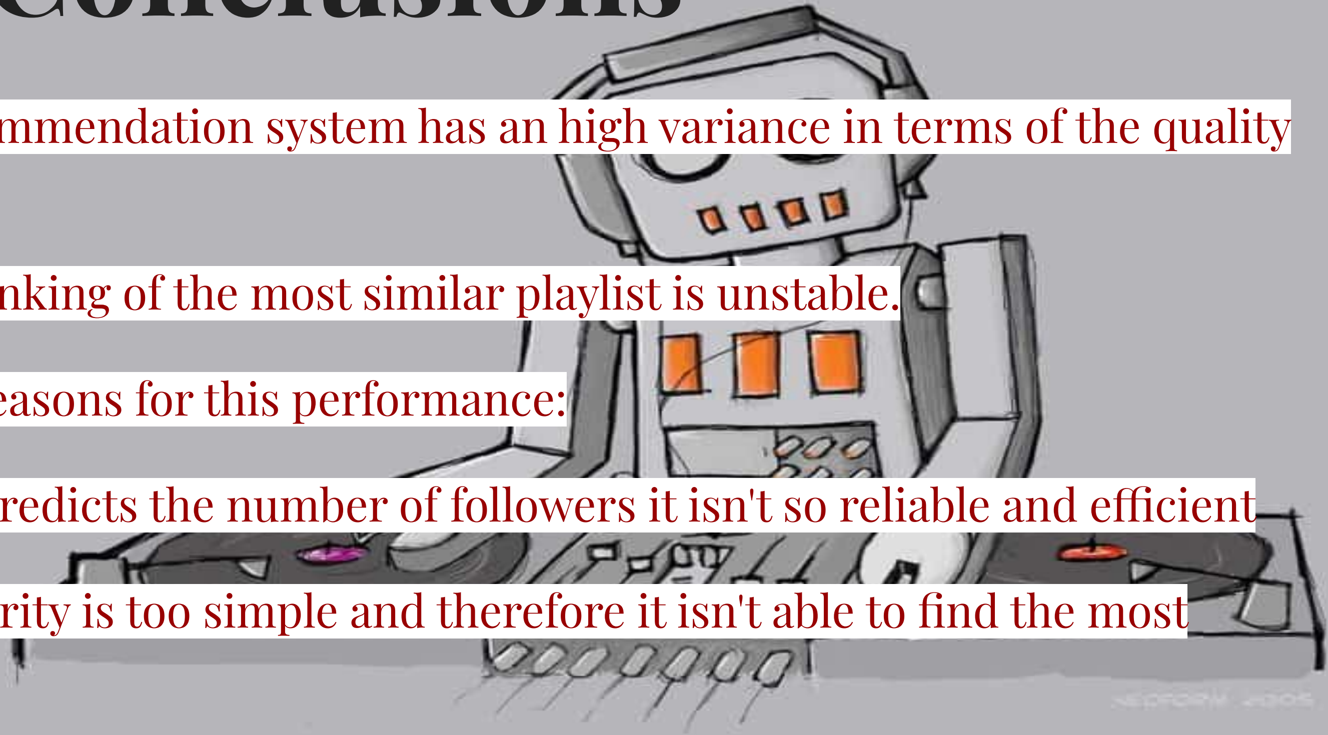
Some results

```
tracks that are missing : 0
The recommended playlist is:
```

	track ID	track name	artists names
0	0pqnGHJpmpxLKifKRmU6WP	Believer	[Imagine Dragons]
1	08mG3Y1vljYA6bvDt4Wqkj	Back In Black	[AC/DC]
2	2374M0fQpWi3dLnB54qaLX	Africa	[TOTO]
3	1zB4vmk8tFRmM9UULNzbLB	Thunder	[Imagine Dragons]
4	1JSTJqkT5qHq8MDJnJbRE1	Every Breath You Take	[The Police]

```
Predicted num_followers: 307.67530067397064
[5.52169429 5.52169429]
The most similar playlist's predicted num_followers: 249.05835063967493
There are 788 playlists in genre = rock
The most similar playlist's rank within genre is: 55
```

Conclusions



We find that our recommendation system has an high variance in terms of the quality of the prediction.

In other words the ranking of the most similar playlist is unstable.

There could be two reasons for this performance:

-The ml model that predicts the number of followers it isn't so reliable and efficient

-Our metric of similarity is too simple and therefore it isn't able to find the most similar playlist

We think that both reasons could contribute to the performance of our recommendation system.

BONUS PART

CLUSTERIZATION OF THE TRACKS

AND

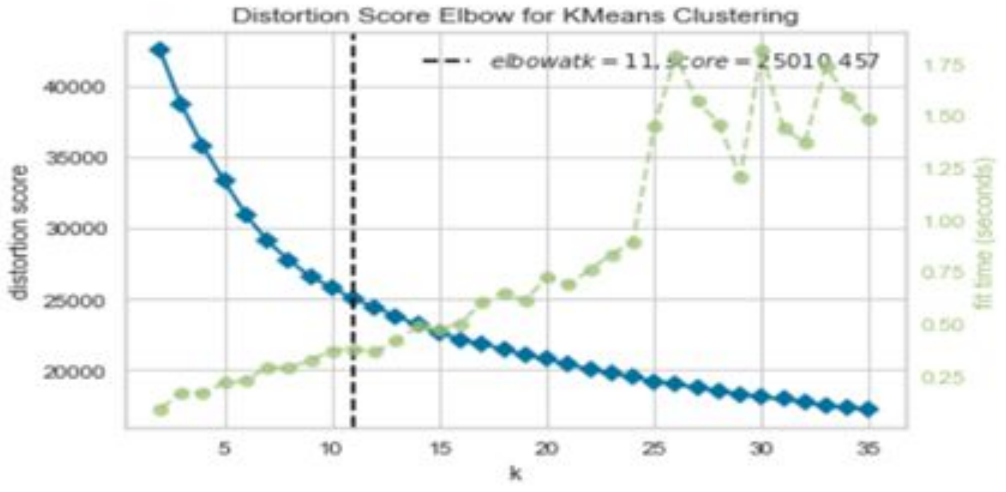
A GRAPHIC VISUALIZATION

MERGING THE DATA FROM 2 DIFFERENT JSON TABLE TO OBTAIN A DATAFRAME WITH PLAYLIST ID, TRACKS-ID AND THEIR METADATA

	id	playlist	acousticness	danceability	energy	explicit	instrumentalness	key	liveness	loudness
0	1bWZPuueQsBbU5BEgYuynn	37i9dQZF1DX9slqqvKsjG8	0.995	0.443	0.0316	False	0.930	0.0	0.1190	-25.607
1	3qp1ushu4Ve2Vl5keFaUDM	37i9dQZF1DX9slqqvKsjG8	0.989	0.242	0.0544	False	0.915	8.0	0.1070	-26.277
2	0IBQM8tB107DZs0YCcXSt0	37i9dQZF1DX9slqqvKsjG8	0.986	0.276	0.0507	False	0.867	2.0	0.1070	-25.719
3	2peMLXHmImghobIbAkNjla	37i9dQZF1DX9slqqvKsjG8	0.993	0.341	0.1060	False	0.914	7.0	0.1040	-19.423
4	6VKid1zSeLVUEV3oQR2i0k	37i9dQZF1DX9slqqvKsjG8	0.987	0.378	0.0603	False	0.894	0.0	0.1110	-24.635
5	3uz5pvPIZEkgRgOpaP20Ge	37i9dQZF1DX9slqqvKsjG8	0.994	0.484	0.0292	False	0.950	5.0	0.1100	-23.059
6	1BZo2CHWy9gSgidvexgNYM	37i9dQZF1DX9slqqvKsjG8	0.995	0.291	0.0121	False	0.827	10.0	0.1120	-28.978
7	1PYVWMwBocquvlCpYzwwxA	37i9dQZF1DX9slqqvKsjG8	0.995	0.361	0.1370	False	0.949	3.0	0.0988	-25.830
8	7nC2EOpMnpDT2DkvniimSm	37i9dQZF1DX9slqqvKsjG8	0.972	0.389	0.1000	False	0.934	10.0	0.1130	-22.464
9	7BbUNLqsUWQAM0QUNoFZWs	37i9dQZF1DX9slqqvKsjG8	0.993	0.390	0.0434	False	0.938	2.0	0.1080	-22.244
10	7aw2vM9GkttgVrbao29wju	37i9dQZF1DX9slqqvKsjG8	0.996	0.258	0.1320	False	0.957	1.0	0.1020	-28.306

INITIALIZE THE CLUSTERIZATION DISCOVERING THE K* (OPTIMAL NUMBER OF CLUSTER)

```python
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
model = KElbowVisualizer(KMeans(), k=35)
model.fit(train_new.loc[:, clusterCols])
model.show()
```



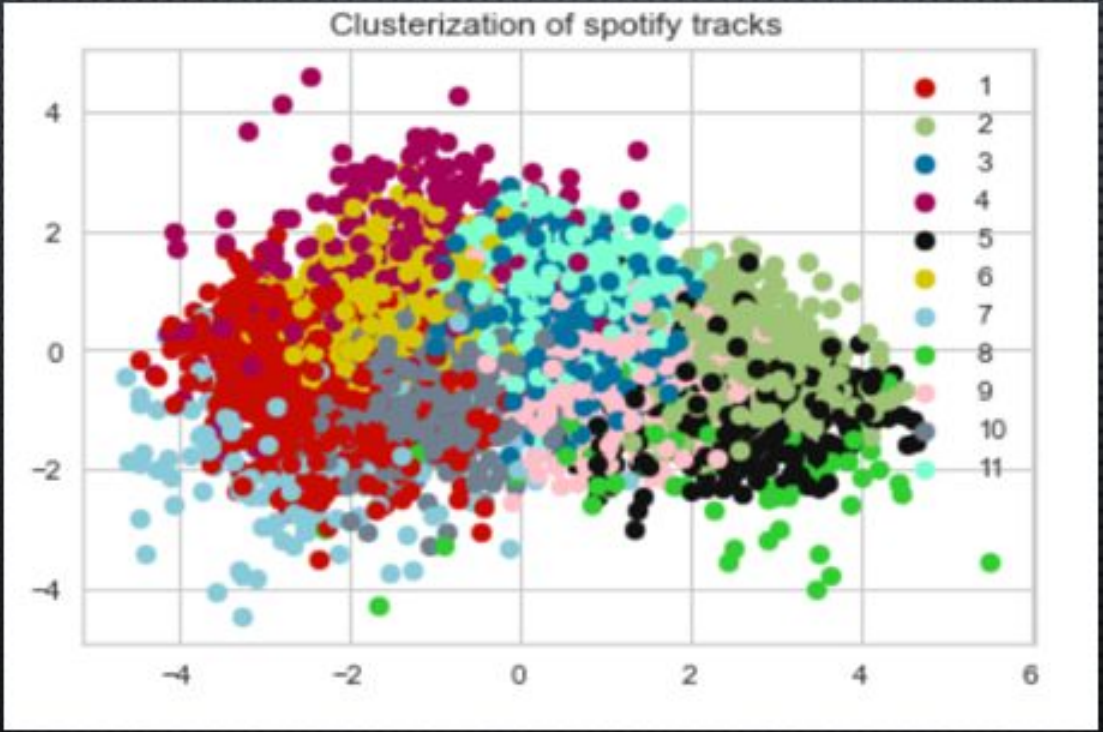
ASSIGN CLUSTER

	playlist	id	cluster_label	acousticness_scaled	danceability_scaled	energy_scaled	instrumentalness_scaled
0	54AOa7ClyM1pxgi3Wypu4G	2lOFt38PO9BqPWVJLXTBbC	2	0.530466	0.755525	0.033153	0.571369
1	3po6Vxpx1CY1A4NkO7NTRz	3qS1P8A7CDtkbJdLTx0GZf	0	0.997682	0.076022	-0.070511	0.577184
2	74sUjcvpGfdOvCHvgzNEDO	6utQSu0CAvFb8wnEckHVbt	10	-1.232754	0.119861	0.007237	-0.286317
3	1YJe9dmtfOWC2lKbIdJUWm	1tBOzT3RQZWDS5zkBrMq4r	8	1.015852	-1.200785	-0.877610	-0.318298
4	37i9dQZF1DX3qCx5yEZkcJ	03qYV8hat9j4ttaPN2HKfh	3	-1.058067	1.111715	0.603306	-1.620818
5	4DvDLiZxLeUlHWMErYhncB5	3NciuZi7hSVWotPjMxWu55	9	-0.009427	0.443173	0.658840	0.705110
6	148My4xA7WoEaNDmnl2A8Z	16fPrLFNDUXHBnarWpmVXt	6	-1.332167	0.837722	1.043878	0.728370
7	4s6BkxYZKg0OeIJClhpsO5	2bJdcsMjBDamfjx7Wrr9R	2	0.797818	0.371934	-0.385206	0.478332
8	37i9dQZF1DXdlhitnpe6FT	3bkNIeAr9FJXzCHUaF0lSU	1	-0.388391	-2.258398	-0.455549	0.850481
9	1dQQkNsgAcfAzO1KTvHOTR	0Y97dC6xd7woak0lJMQyVs	0	-1.406455	0.750045	0.840252	0.620795
10	4DvDLiZxLeUlHWMErYhncB5	47LQEI2udisu7sRC9dYU4l	6	-1.337099	0.563729	1.121626	0.263184
11	37i9dQZF1DX692WcMwL2yW	2Xih3Id83Y3AoODWCre31i	2	0.250137	0.815803	-0.007572	0.199221
12	37i9dQZF1DX7cmFV9rWM0u	0UefCGxonFYJRoVKzVKFAq	3	-1.382263	1.270631	1.754717	-1.956039
13	37i9dQZF1DWYmSg58uBxin	2apnPHVAbn0ny2K8RowbpN	7	1.127464	-1.584375	-1.163057	0.690573
14	54AOa7ClyM1pxgi3Wypu4G	5c1diLn5lZpBeXooe5LytH	9	0.878283	0.328096	-0.777648	0.742907
15	1la2NHrMKqHtzrewaAJ46Q	2gtiic8nRNAegDFAs9xeYk	0	-0.710251	1.681621	-0.103832	0.478332
16	3VPTv1SsLhMagZzQaCJWpL	2EjMo9MXrTyecbm0wTB004	3	-1.351375	0.103421	0.918000	-1.957220
17	37i9dQZF1DX8Uebhn9wzrS	2qnxhWmaZhgckkAwRDKxUm	6	-0.388391	0.585649	0.136817	0.713833
18	3dwfxNbpKYzbnazjjUEUse	7ubkJak9LSzPMolmespApd	9	0.133333	1.287071	0.221970	0.652777
19	0UmZ7JiX8h3RIYgjIKEWpi	73lw6SsjJJASNSjkcAYgZY	2	0.683609	-1.995365	1.332656	0.187591

ORDER RANK OBTAINED THANKS TO THE EUCLIDEAN DISTANCE FROM RESPECTIVE CENTROIDS.

	0	1	2	3	4	5	6	7	8	9	10
0	0	3	4	8	10	7	1	6	2	5	9
1	1	6	10	2	4	8	7	3	5	0	9
2	2	6	1	10	8	4	5	7	3	0	9
3	3	0	4	8	10	1	6	2	7	5	9
4	4	8	10	1	6	3	2	0	7	5	9
5	5	2	6	10	8	4	1	7	3	0	9
6	6	2	1	10	8	4	7	5	3	0	9
7	7	10	1	4	8	6	2	3	0	5	9
8	8	4	6	2	10	1	3	7	0	5	9
9	9	1	2	10	6	8	4	7	3	5	0
10	10	1	6	2	4	8	7	3	5	0	9

CLUSTER VISUALIZATION: THANKS TO THE PCA DIMENSION REDUCTION.



Thanks for your attention!