Guess the Song

A Recommender System with Clustering and Classification

Team 6

Beca Wang, Chuyu Chen
Linna Li, Macy Bao
Yue Wu, Yizhou Zhang

Table of contents

01

Introduction

Business & Data Background

02

EDA

Interesting Findings in Datasets

03

Modeling Methodology

Modeling & Application

04

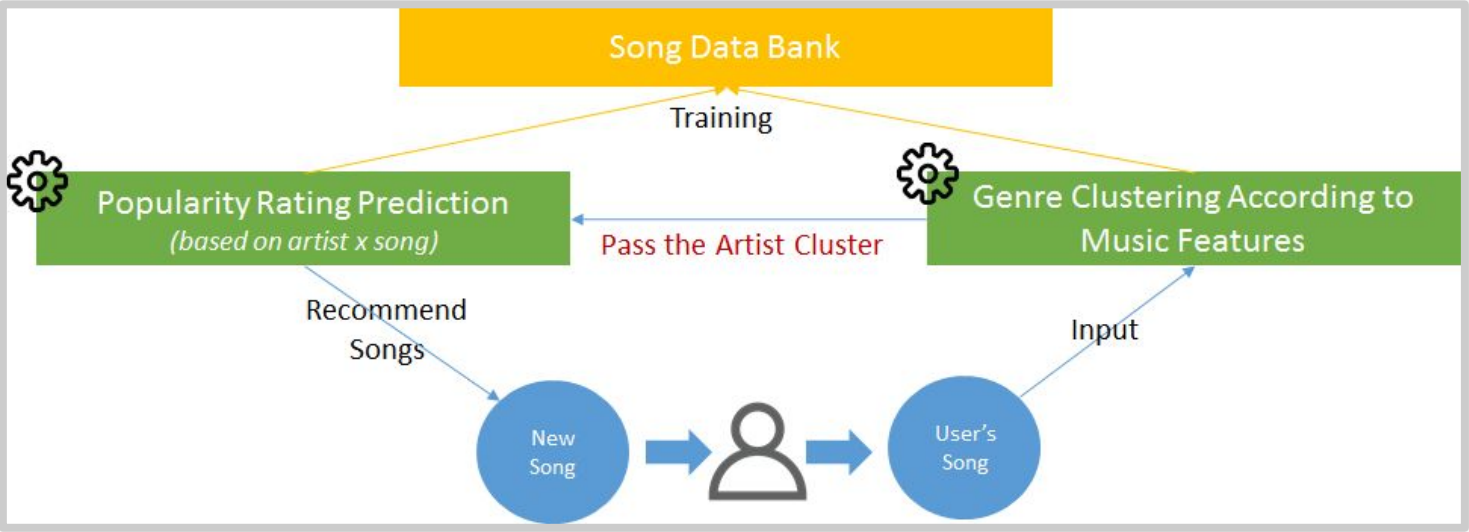
Conclusion

Current Solution & Next Step

Introduction

Business Objective: adds in suggested tracks that our users might like, so that they keep exploring new songs and listening to more albums. It could improve user experience and add to user stickiness.

Goal: build a recommender system based on musical features of songs user liked / already in the playlist and the predicted popularity of the track. (i.e. User preference & popularity)



Exploratory Data Analysis

Metadata

Source: Kaggel Spotify dataset

About: popularity, artists, and other musical features of Spotify song tracks from 1921-2020. 174,389 entries (songs) and 19 variables.

Identifier	id	Auto-generated by Spotify
Numerical	popularity	Target of popularity prediction
	acousticness, danceability, energy, duration_ms, loudness, year...	Musical features
Dummy	mo	
Categorical	key	

```
genres
[]                                                                                                  13563
['movie tunes']                                                                                        68
['show tunes']                                                                                         53
['orchestral performance']                                                                             47
['vocal trance']                                                                                       46
                                                                                                      ...
['dark progressive house', 'deep progressive house', 'focus trance', 'progressive breaks']             1
['dark progressive house', 'focus trance', 'progressive breaks']                                       1
['dark progressive house', 'georgian electronic']                                                      1
['dark progressive house']                                                                             1
["australian children's music", "children's folk", "children's music", "preschool children's music"]   1
Name: genres, Length: 10841, dtype: int64
```

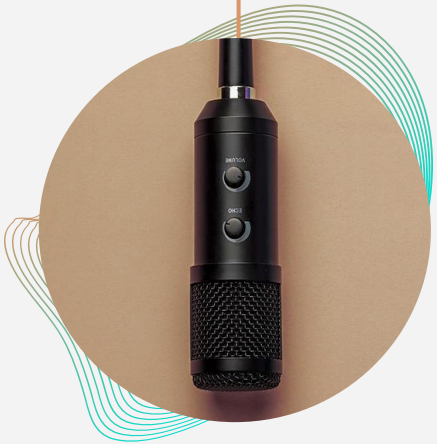
Cleaned Dataset

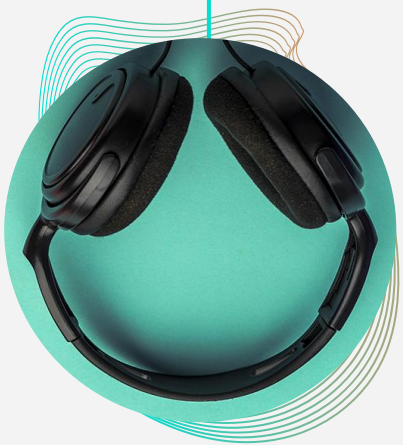
01

Genre

Genres

- Converted duration ms to min
- Dropped artist column
- Remove outliers





02

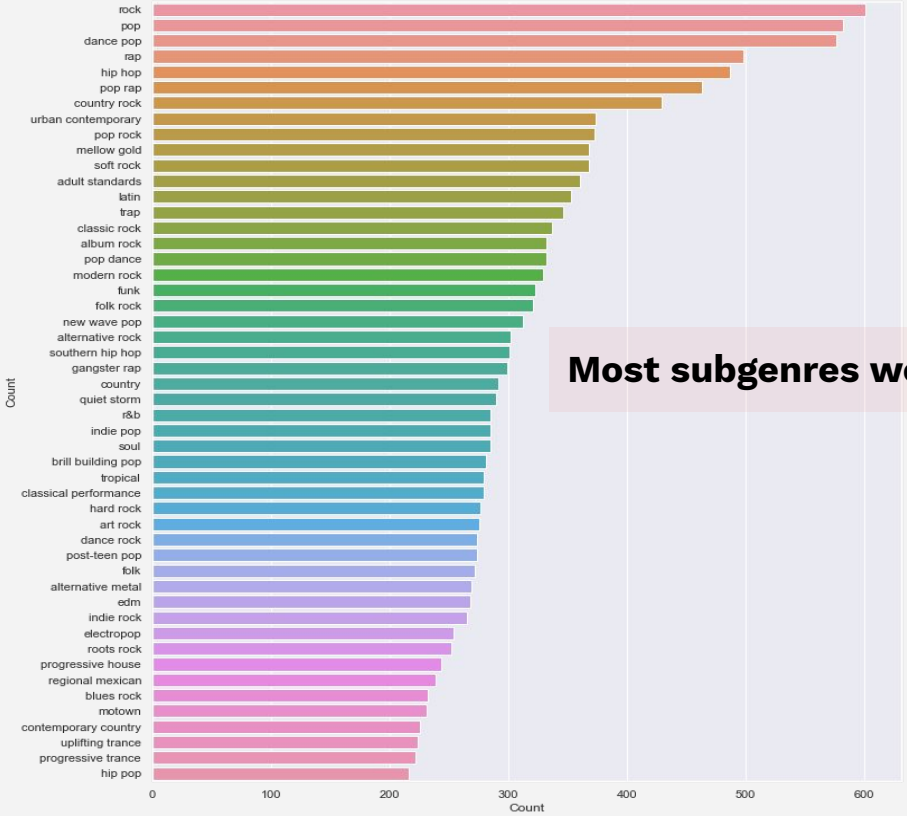
Popularity

Songs + Artists

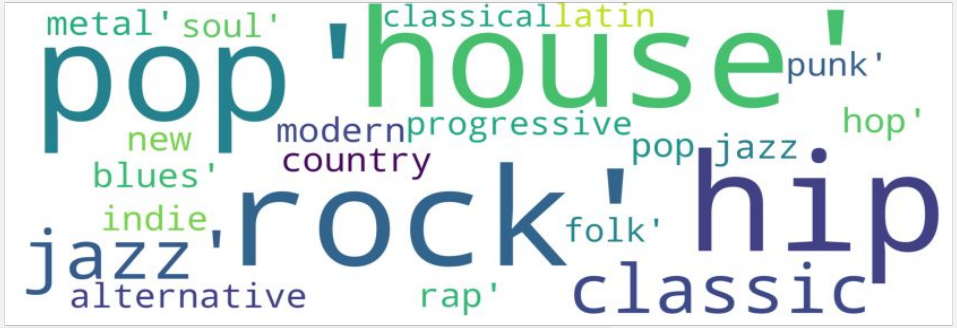
- Number of Artists
- Num Words in Song Name
- Group years to periods
- Expand based on the artist
- Create unique key of dataset
- Attach year info
- Attach artist info
- Fill missing with median

01. Genre

Most popular genres

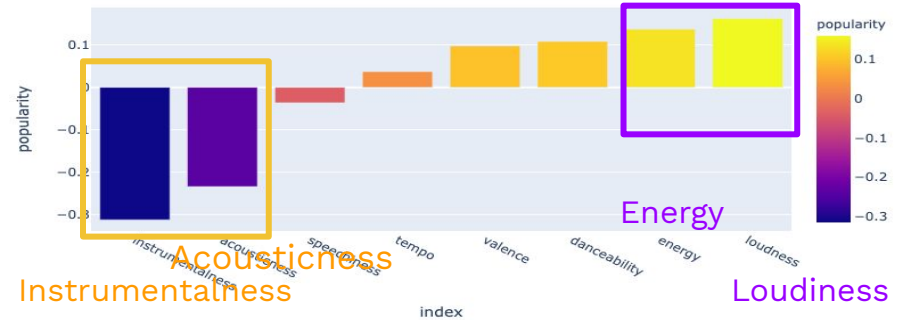


Most frequent words from genres

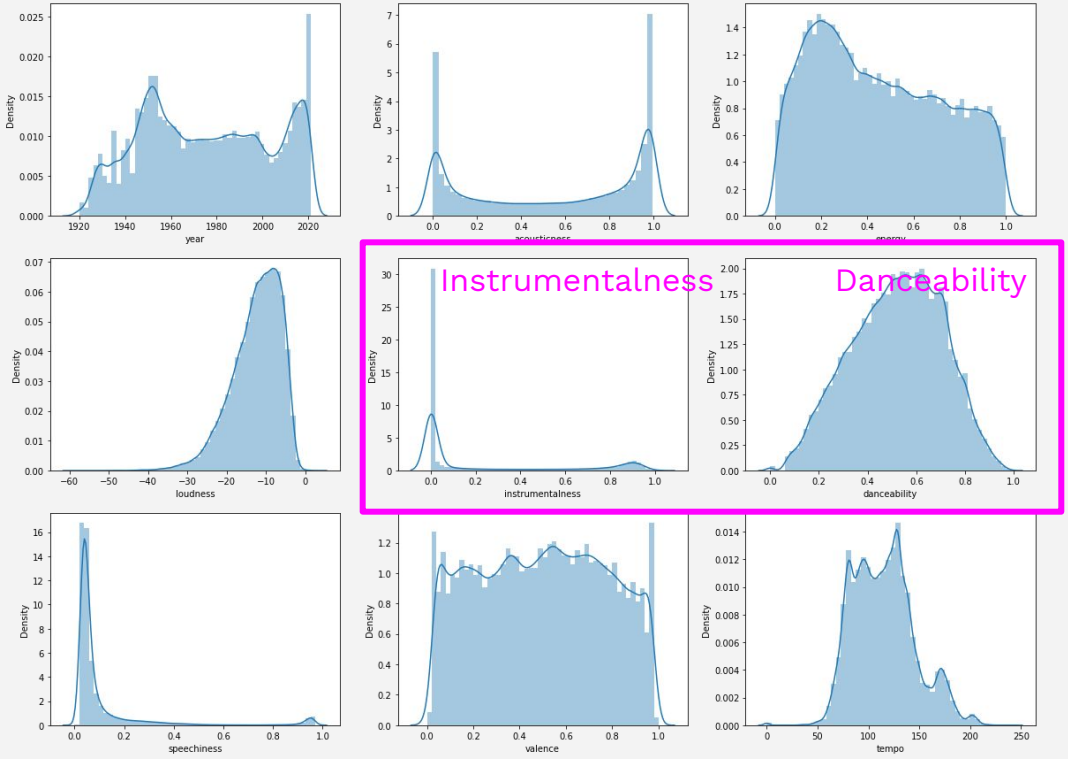


Most subgenres would come from these genres

Audio Features vs. Popularity

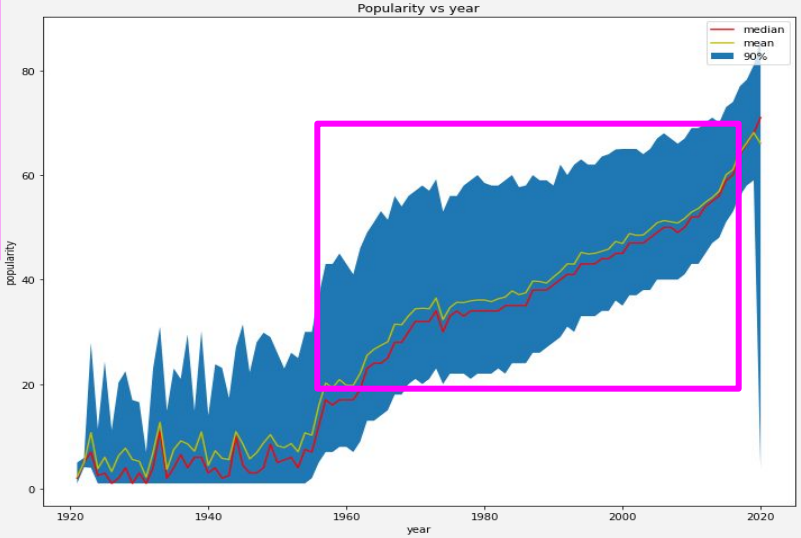


02 Popularity



Instrumental:
Songs in the dataset are mostly non-instrumental.

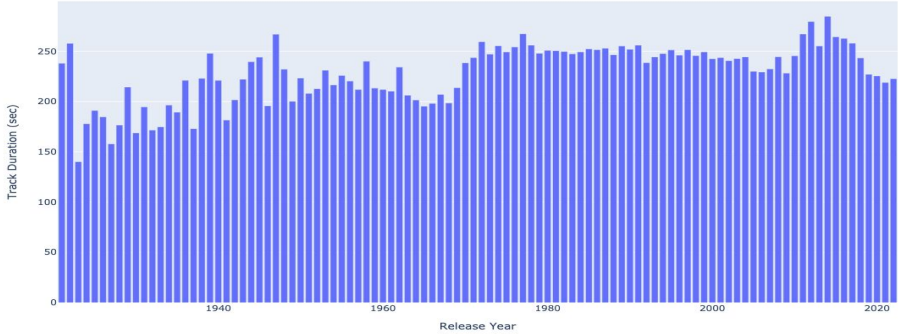
Danceability :
Most songs around 0.5 - 0.7 danceability.



02 Popularity

Number of Tracks over Period

Track Duration over years



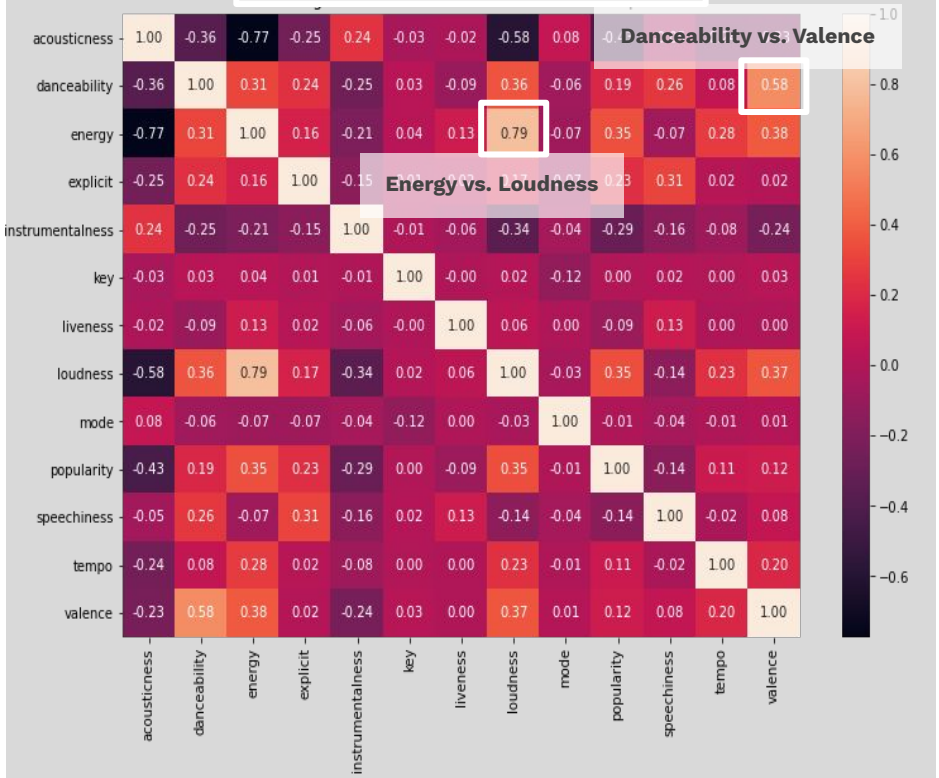
Most common words

In Song Names

In Artists

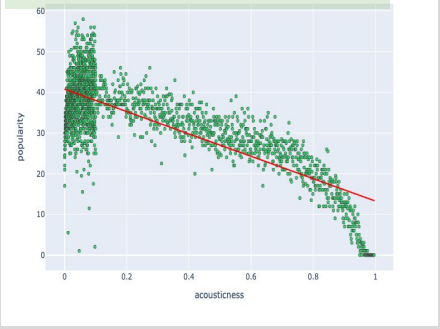
02 Popularity - Correlation

Song Features

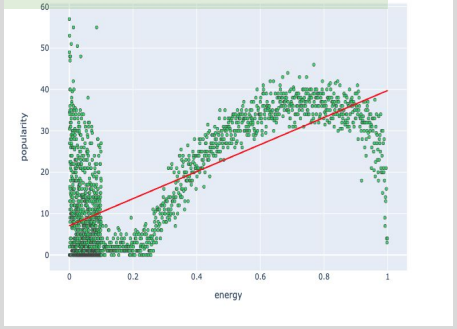


Song Features vs. Popularity

Popularity vs. Acousticness

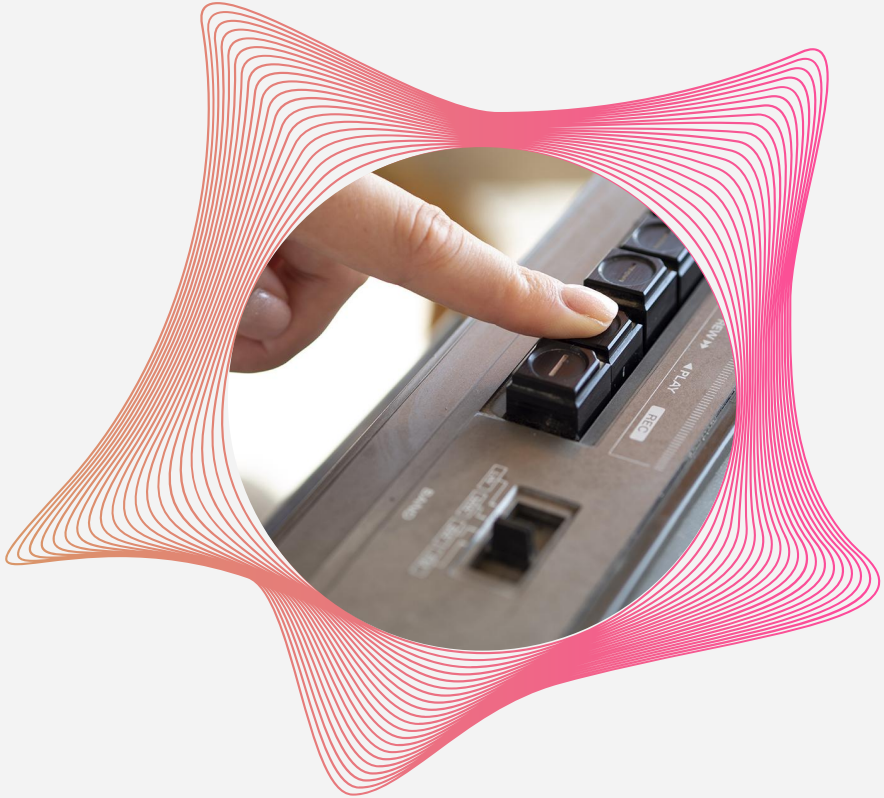


Popularity vs. Energy

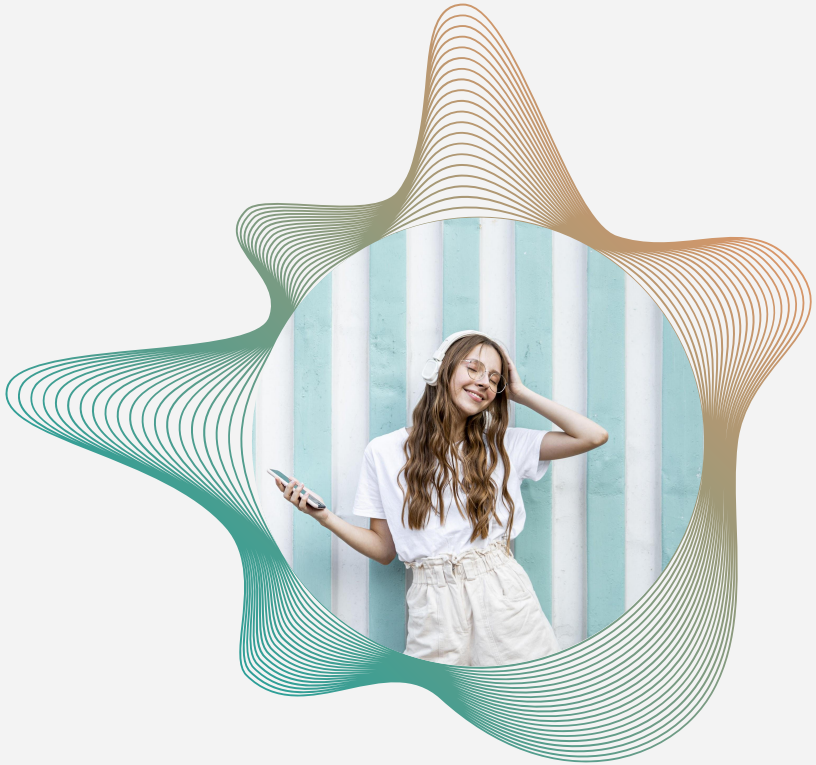


Modeling Methodology

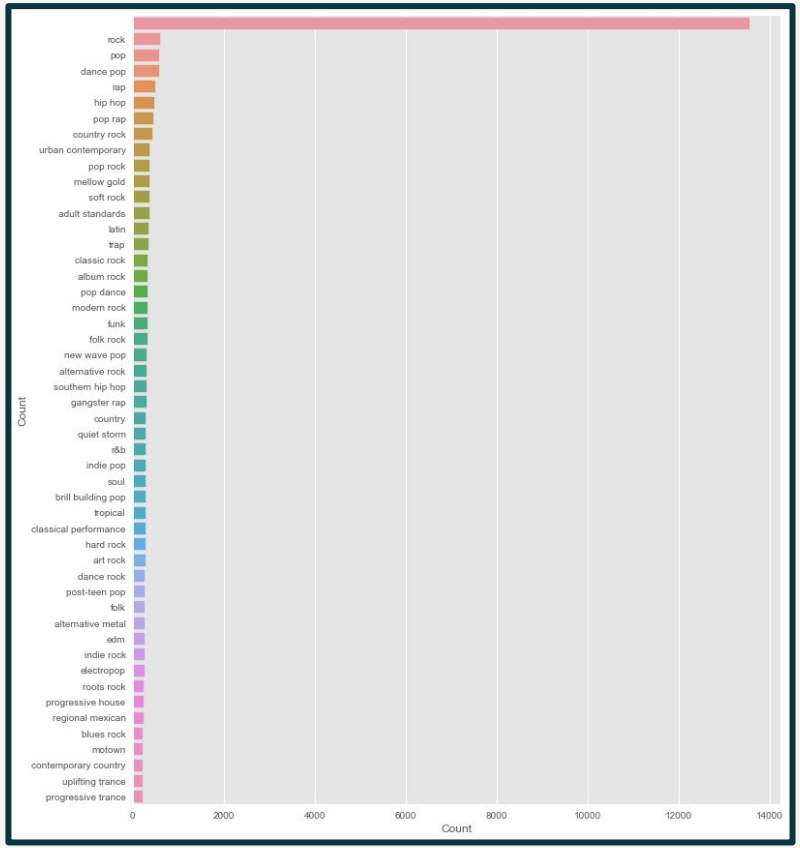
Building up the two components of the recommendation system



Clustering



Clustering Component - Data Preparation

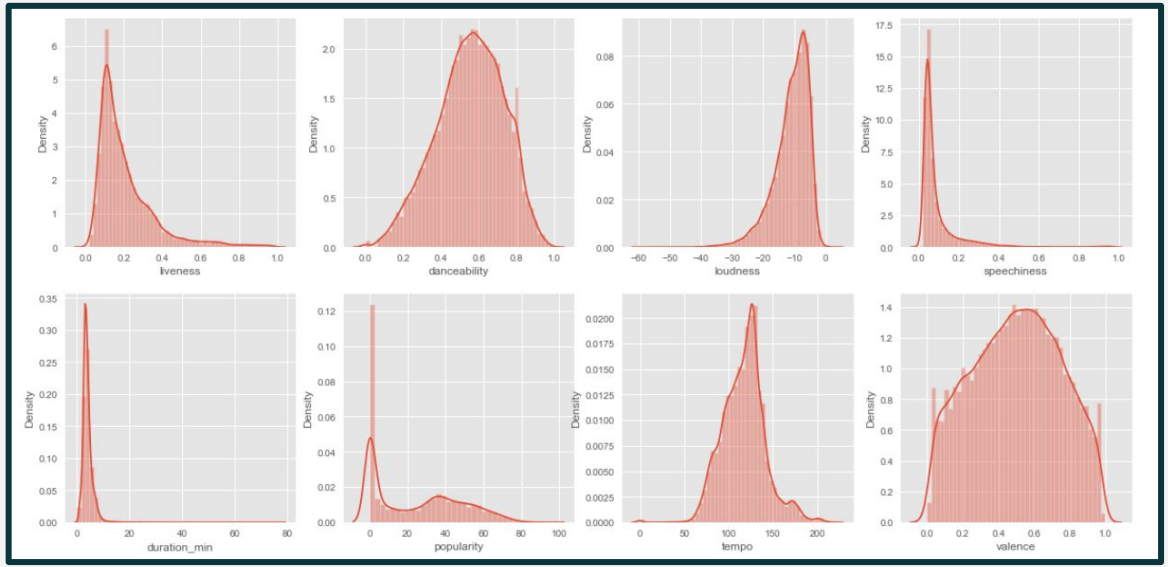


Explore genre column

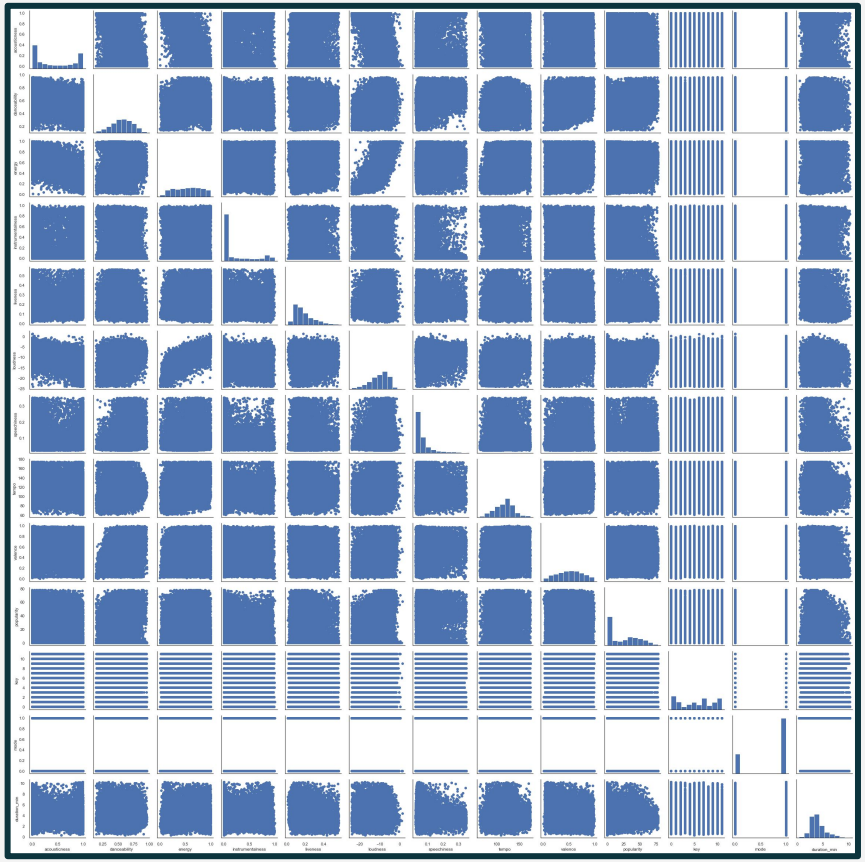
- After the cleaning, we have 3232 unique music genres in the dataset (including null value)
- Since there are too many null values and unique genres which is confusing, we choose to drop the column and make the clustering ourselves according related music features and differentiate super-genres

Clustering Component - Data Preparation

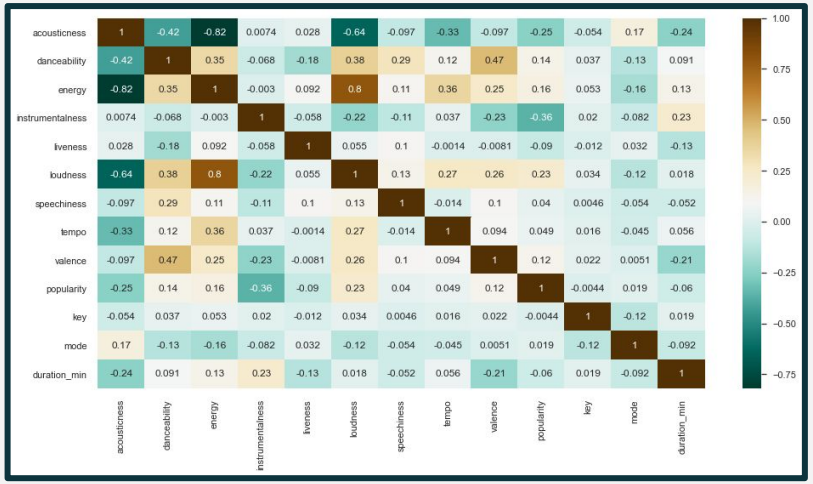
- **Removed outliers for better model performance using a threshold of 0.99**



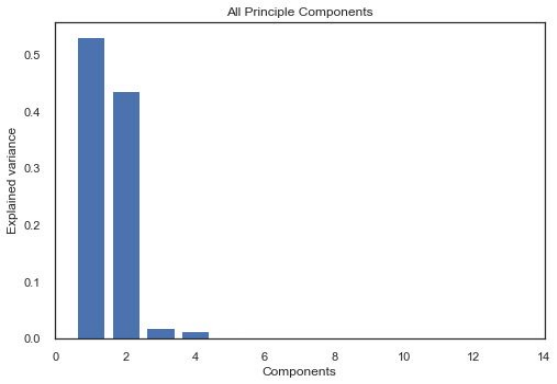
Clustering Component - Data Preparation

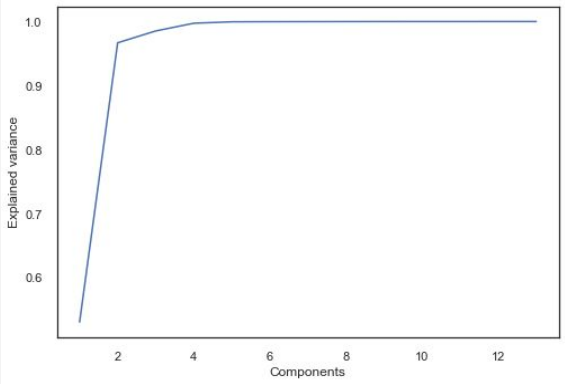


Check collinearity and correlation

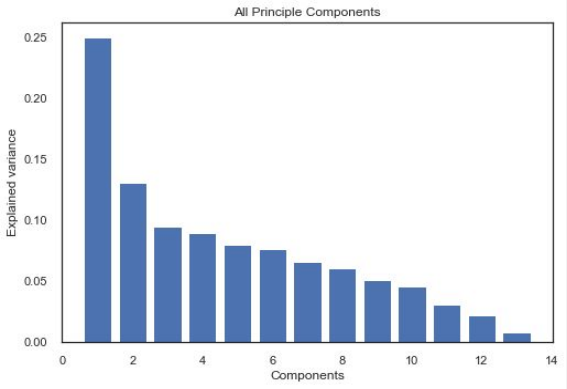


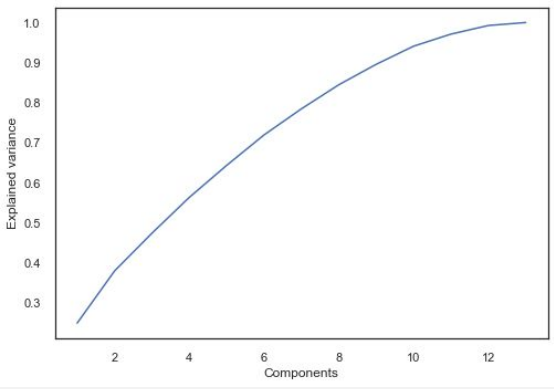
Clustering Component - Dimension Reduction





PCA on original dataset



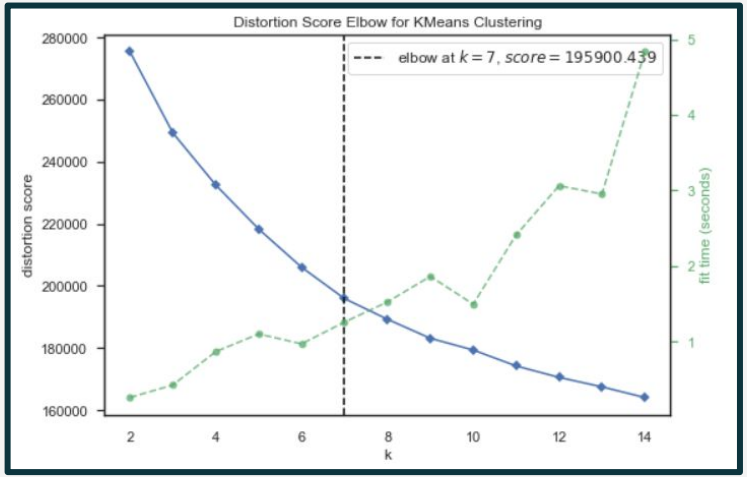


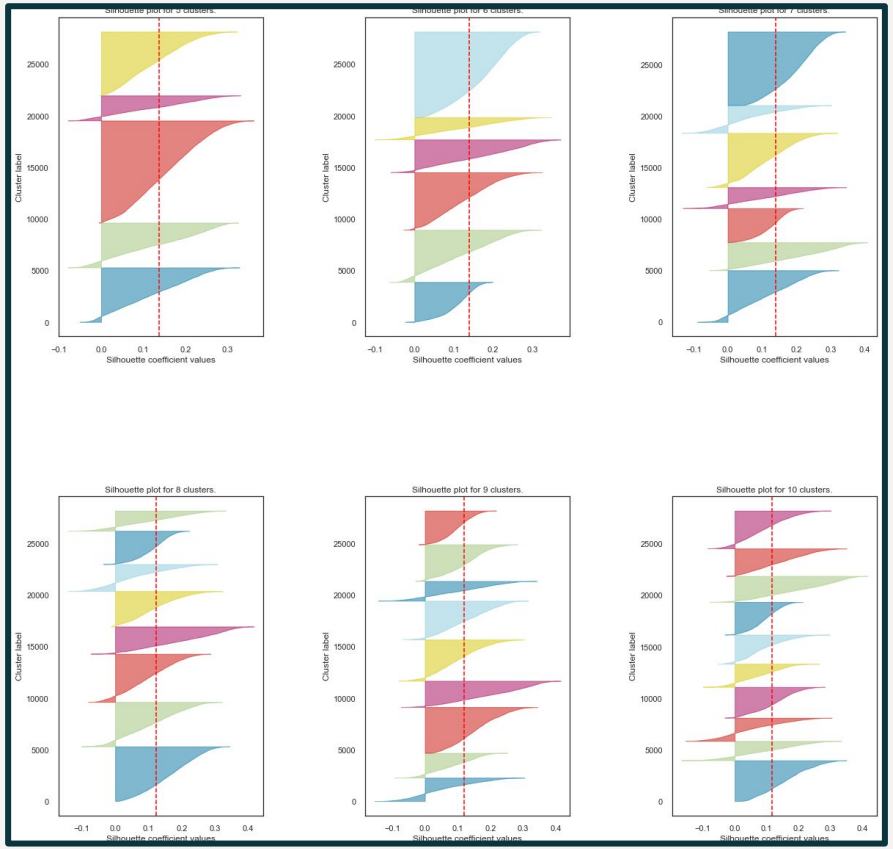
PCA on scaled dataset

Clustering Component - Modeling (KMeans)

Silhouette Score
K = 6

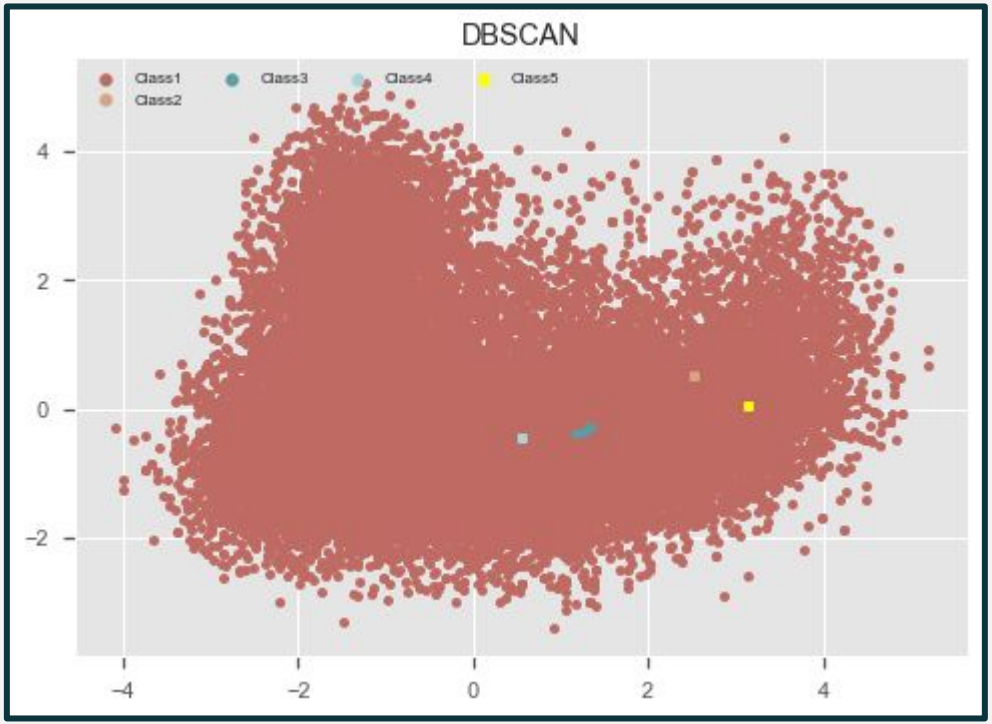
Elbow Method



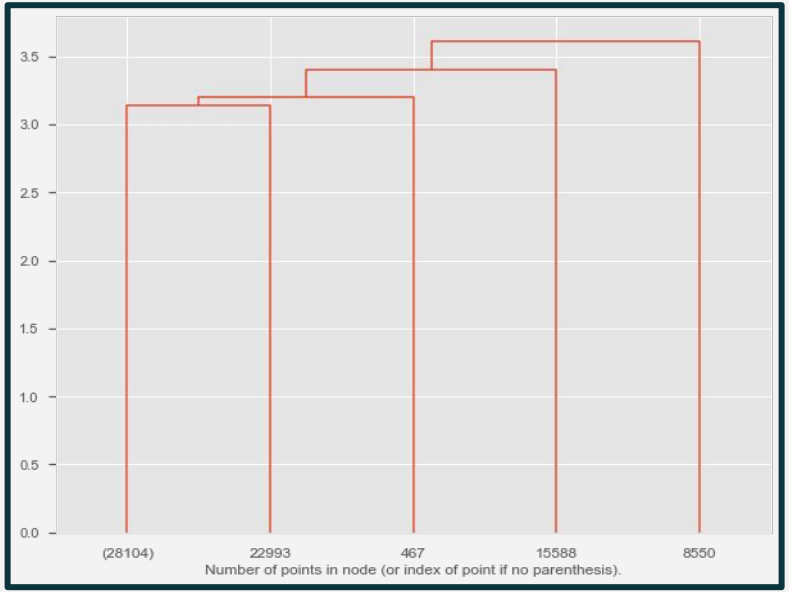


Clustering Component - Modeling (DBSCAN)

- **DBSCAN performs poorly on our dataset**
- **Since the data is bunched together, DBSCAN keeps combining the points into 1 giant cluster, and makes a few new clusters for outliers which are further away**

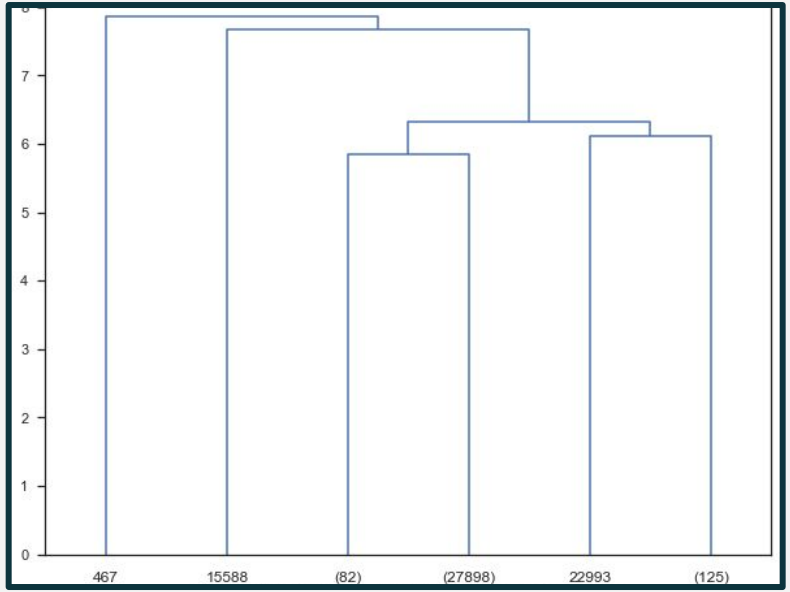


Clustering Component - Modeling(Hierarchical)

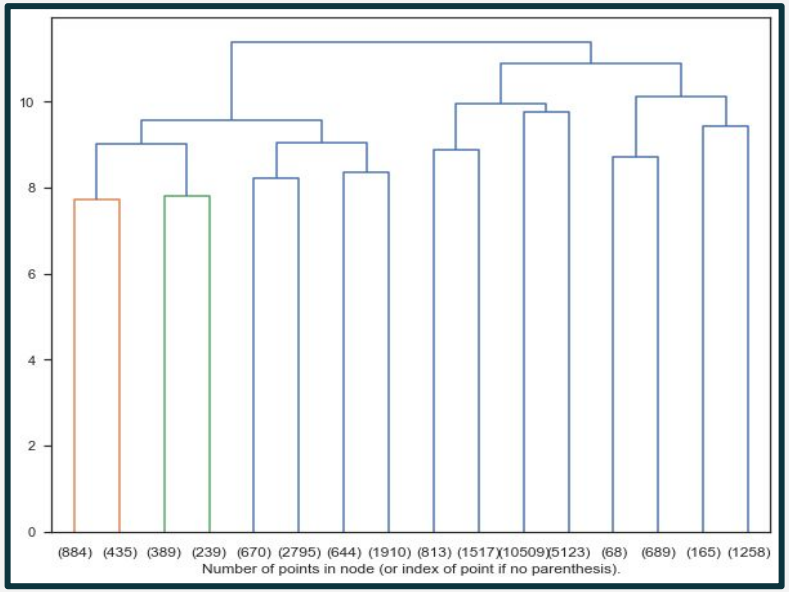


Single linkage

Average linkage

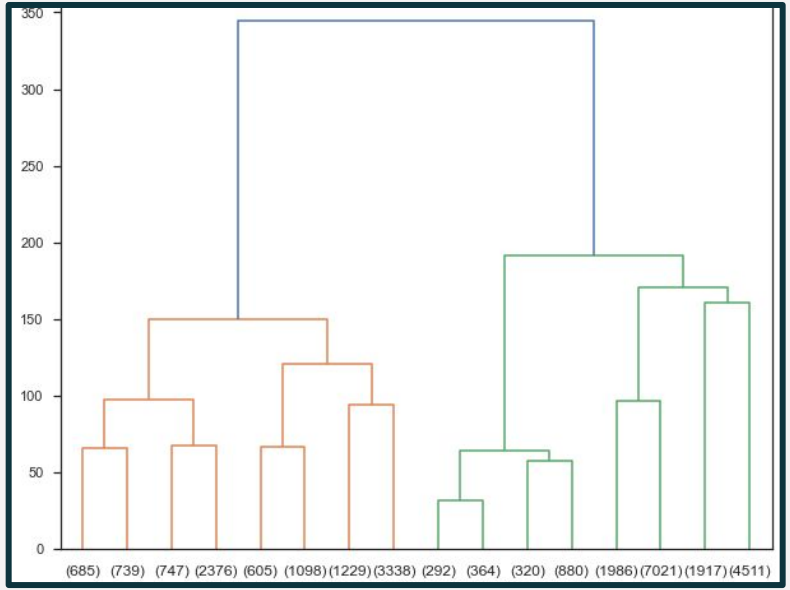


Clustering Component - Modeling (Hierarchical)



Complete linkage

Wald linkage



Clustering Component - Modeling (Affinity Propagation)

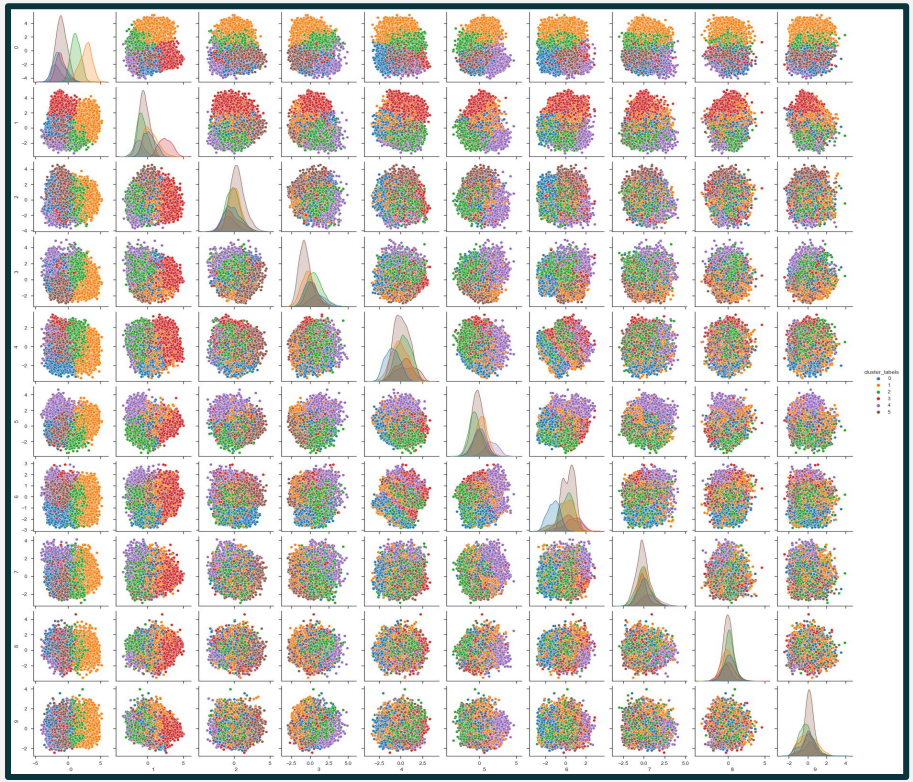
- **Affinity propagation is a clustering algorithm based on the concept of "message passing" between data points**
- **This runtime of algorithm on our dataset keeps crashing**

Affinity Propagation

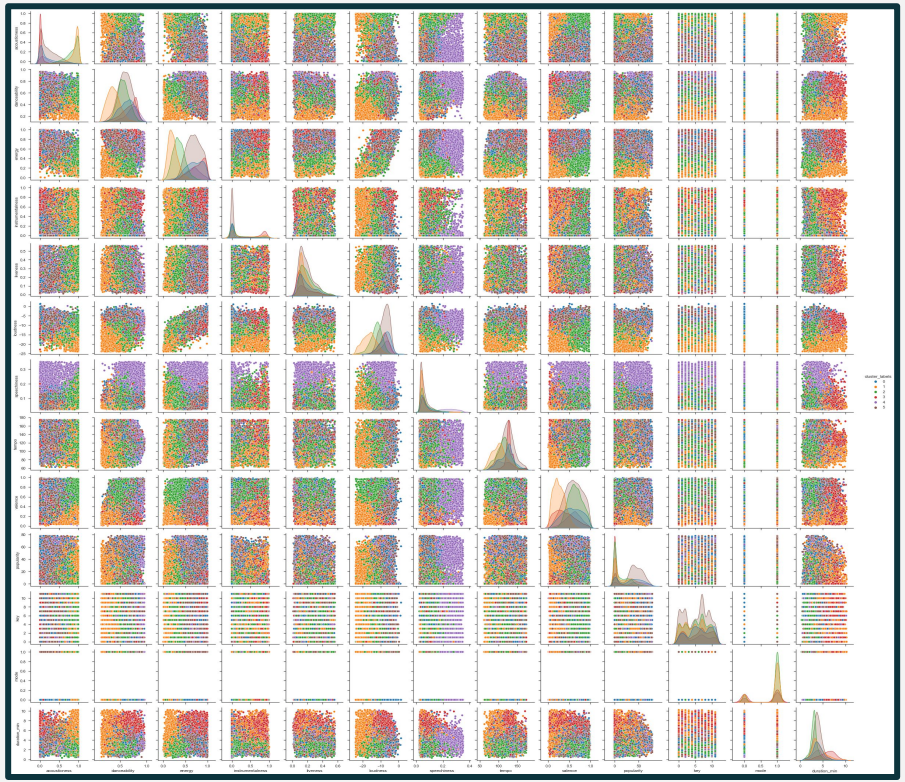
```
# from sklearn.cluster import AffinityPropagation

# ap = AffinityPropagation()
# ap.fit(data4)
```

Clustering Component - KMeans Performance

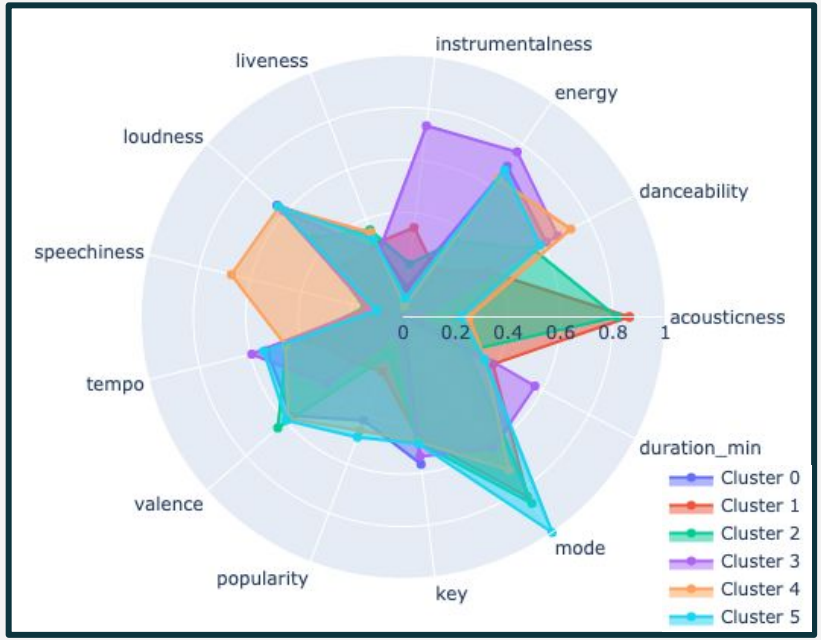
Pair plot of scaled dataset

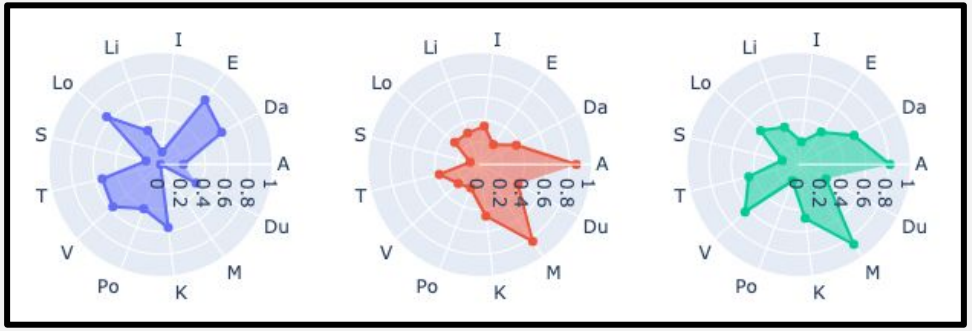


Pair plot of original dataset



Clustering Component - Validation





- **Cluster 0: R&B**

  These songs have moderate tempo and danceability, with a positive outlook. They are popular among people.

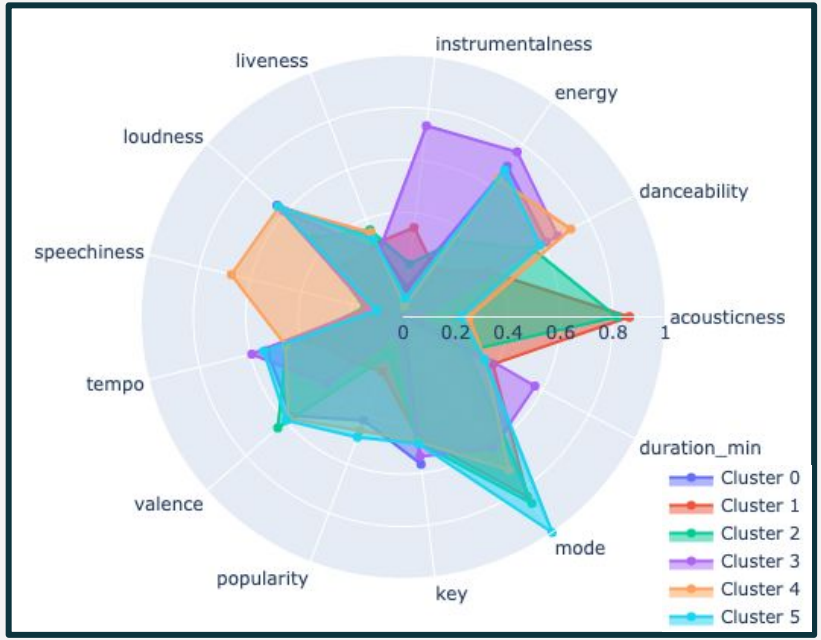
- **Cluster 1: Rock**

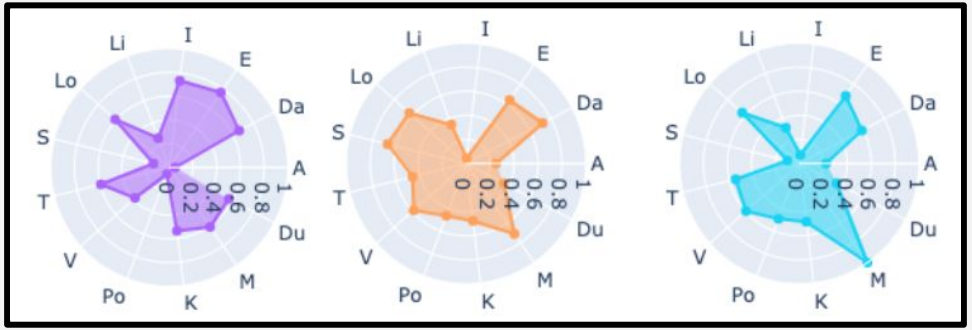
  High usage of acoustic instruments. these have moderate tempo but aren't very popular.

- **Cluster 2: Classical**

  Songs with low speechiness which use acoustic instruments. These have moderate dancibility, are cheerful, but aren't very popular.

Clustering Component - Validation





- **Cluster 3: Electronic**

  High energy, loud songs with high use of electronic instruments, high tempo and low speechiness. These songs also rank fairly high on danceability, but have a niche audience.

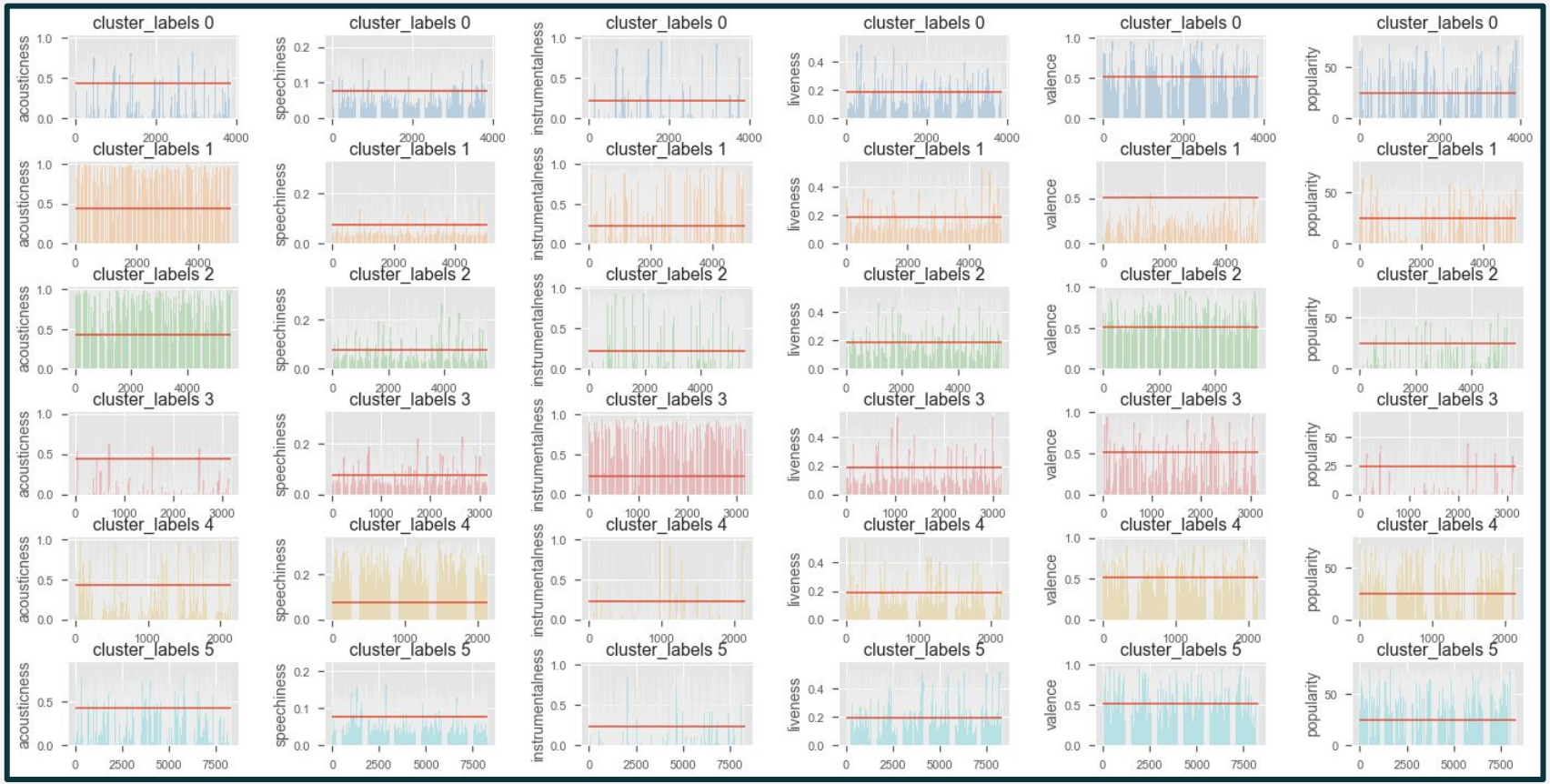
- **Cluster 4: HipHop/Rap**

  These songs are fairly popular, and they contain the most verbose tracks and danceability, low in acousticness.

- **Cluster 5: Pop**

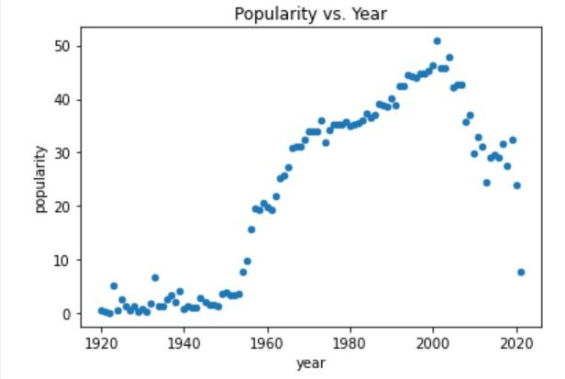
  Loud songs with high energy and high danceability. These have a positive outlook, a good mix of speechiness and instrumentalness and are widely popular

Clustering Component - Validation



Popularity Prediction

Popularity Component - Data Preparation

Predictive Variables	Target Variable
• **49 predictors** in total • Non-eng word detection • Artist expansion • Group period by year 	• **Popularity** • Redefine popularity 1 64041 4 55916 2 40608 3 40160 5 33647

Popularity Component - Modeling (Regression)

	Training Accuracy	Testing Accuracy
Linear Regression	0.59	0.59
XGB Regression	0.76	0.64
Gradient Boosting	0.72	0.69

Popularity Component - Modeling (Ordinal Classification)

By applying Ordinal Multi-classification, we will take advantage of the ordered class value by transforming a k-class ordinal regression problem to a k-1 binary classification problem. In our case:

- binary target is 1 if ratings > 1 → Pr(Popularity > 1)
- binary target is 2 if ratings > 2 → Pr(Popularity > 2)
- binary target is 3 if ratings > 3 → Pr(Popularity > 3)
- binary target is 4 if ratings > 4 → Pr(Popularity > 4)

Modeling:

- Base Model: Decision Tree
- Tune the tree by max_depth and min_samples_split

Performance:

- Overall accuracy < 50%
- Potential Reason:
  - Simple DT model
  - No careful tuning
  - Imbalanced data

$$Pr(y=1) = 1-Pr(Pop > 1)$$
$$Pr(y=2) = Pr(Pop > 1) - Pr(Pop > 2)$$
$$Pr(y=3) = Pr(Pop > 2) - Pr(Pop > 3)$$
$$Pr(y=4) = Pr(Pop > 3) - Pr(Pop > 4)$$
$$Pr(y=5) = Pr(Pop > 4)$$

ID	Song	Popularity
A	Keep A Song	1
B	I Put A Spell On You	2
C	Golfing Papa	3
D	Oli mera paizei zaria	4
E	El Africano	5

Pop>1

ID	Song	Popularity
A	Keep A Song	0
B	I Put A Spell On You	1
C	Golfing Papa	1
D	Oli mera paizei zaria	1
E	El Africano	1

Pop>2

ID	Song	Popularity
A	Keep A Song	0
B	I Put A Spell On You	0
C	Golfing Papa	1
D	Oli mera paizei zaria	1
E	El Africano	1

Pop>3

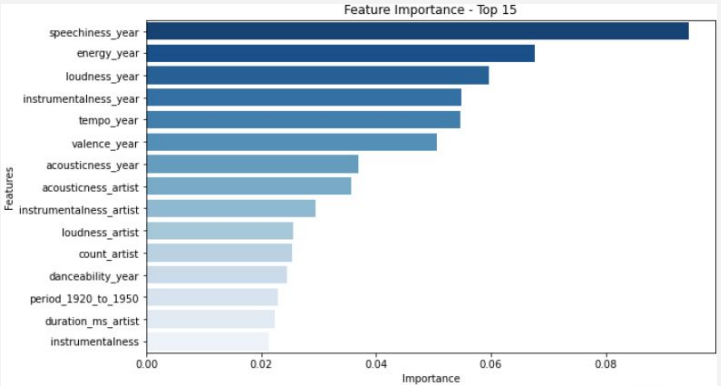
ID	Song	Popularity
A	Keep A Song	0
B	I Put A Spell On You	0
C	Golfing Papa	0
D	Oli mera paizei zaria	1
E	El Africano	1

Pop>4

ID	Song	Popularity
A	Keep A Song	0
B	I Put A Spell On You	0
C	Golfing Papa	0
D	Oli mera paizei zaria	0
E	El Africano	1

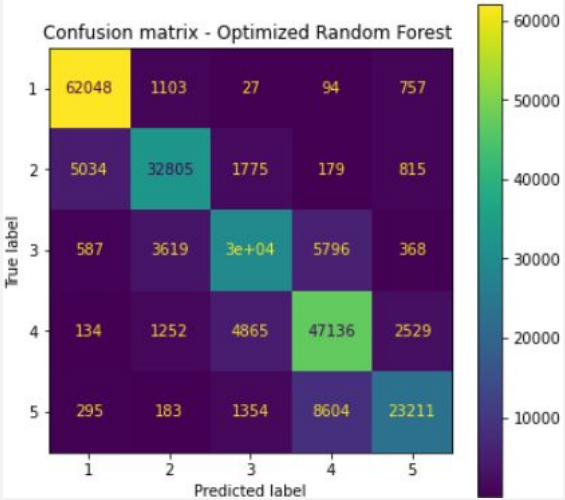
Popularity Component - Modeling (Multi-Classification)

- **Random Forest Classification**
  - Bagging = Bootstrap + Aggregating
  - Reduces overfitting by random sampling on both observations and features when building trees in a parallel style.
  - Fewer hyperparameters to tune which is time and computation efficient
  - Provides feature importance for decisioning



Classification Report on Test Set

	precision	recall	f1-score	support
1	0.88	0.95	0.91	16007
2	0.77	0.72	0.75	10152
3	0.65	0.58	0.61	10040
4	0.67	0.76	0.71	13979
5	0.77	0.63	0.69	8412
accuracy			0.76	58590
macro avg	0.75	0.73	0.73	58590
weighted avg	0.75	0.76	0.75	58590

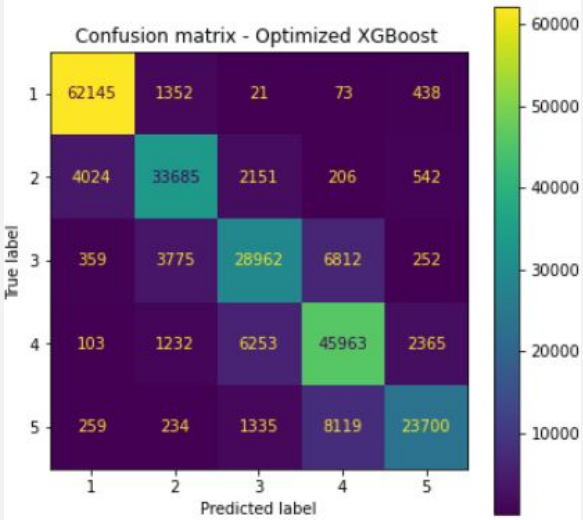


Popularity Component - Modeling (Multi-Classification)

- **XGBoost Classification**
  - It repetitively leverages the patterns in residuals, strengthens the model with week learners and optimize. The method combines the advantages of Random Forest and Gradient Boost.
  - The hyperparameter tuning is costly. With GridSearchCV, training model tool approximately more than 3 hours for 12 sets of parameters.
  - Better performance on Training (~8%) but very close on the Test set with Random Forest.
  - More likely to overfit than Bagging methods and more sensitive parameters to tune (e.g. learning rate, gamma, etc). A better method on complex data set with powerful GPU.

Classification Report on Test Set

	precision	recall	f1-score	support
1	0.89	0.95	0.92	16007
2	0.75	0.74	0.75	10152
3	0.63	0.58	0.61	10040
4	0.67	0.75	0.70	13979
5	0.79	0.63	0.70	8412
accuracy			0.75	58590
macro avg	0.75	0.73	0.73	58590
weighted avg	0.75	0.75	0.75	58590



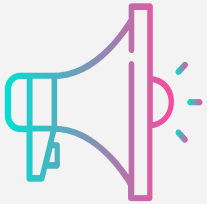
Popularity Component - Ensemble Learning

- **Purpose of Ensemble Learning**
  - Performance: An ensemble can make better average predictions than any single contributing model.
  - Robustness: An ensemble reduces the spread or dispersion of the predictions and model performance.
  - In our case, combine the power of all the three effective methods obtaining similar scores.

- **Performance**
  - Balanced Variance-Bias
  - Keep good training set score with a better one on test set
  - Improve accuracy scores on the difficult-to-predict levels
  - More robust performance can adapt to new data

	Overall	**Level 3/4**	**Train**	**Test**
GB Regression	71%		72%	69%
Random Forest	81%	76%/72%	82%	76%
XGBoost	83%	75%/75%	86%	77%
Ensemble	82%	77%/78%	84%	78%

Application - Case Study



Artist Input

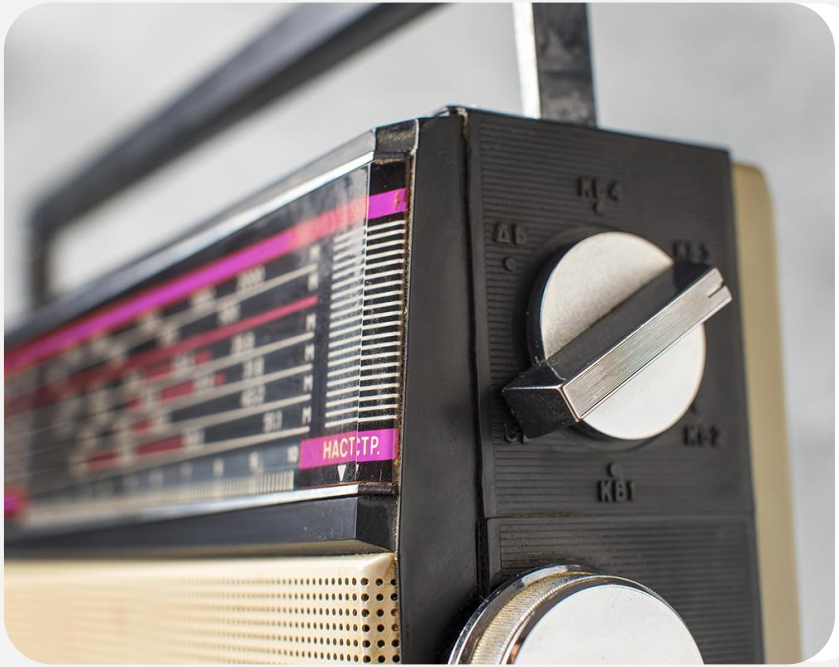
Clustering

Popularity Prediction

Artist & Song Recommendation

Application - Example Input: 2NE1

Artists	Songs
Dailey & Vincent	Cumberland River
Funkadelic	Maggot Brain
Jwonder	Dirty S Hwy



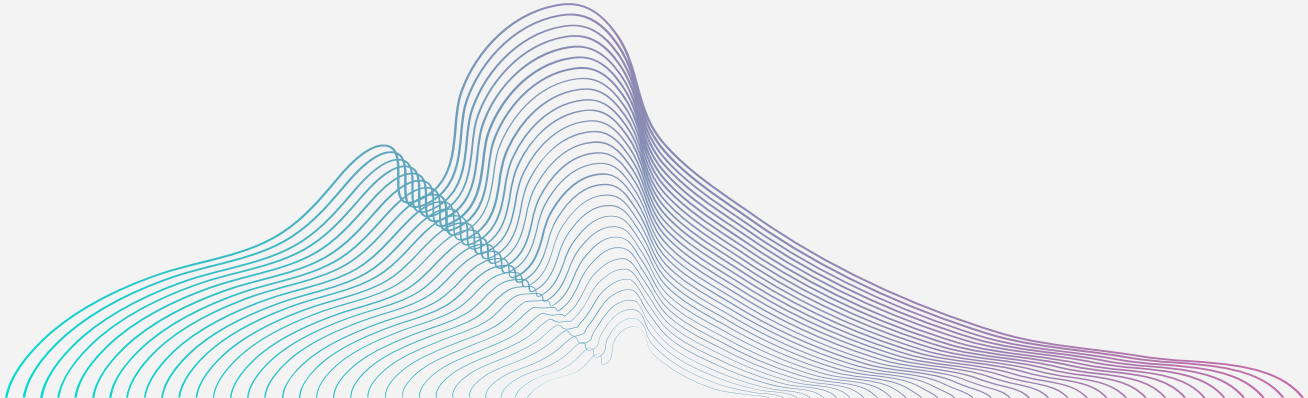
Conclusion

Analysis on the current solution and the next step

Conclusion & Next Step

- **Key Findings**
  - K-Means performs the best in genre clustering. Generated 6 genres according to musical features: R&B, rock, classical, electronic, rap, and pop.
  - Bagging can handle overfitting issue better
- **Challenges**
  - Computation power limitation
- **Next Steps**
  - Further tuning on hyperparameters
  - Deal with imbalanced data for multi-classification problem
  - Find more external data related to the songs and artists

Thanks!



Appendix -- Classification

```python
def clf_fit(X, y, clf):

    clfs = {}
    unique_class = np.sort(np.unique(y))

    if unique_class.shape[0] > 2:
        for i in range(unique_class.shape[0]-1):
            # for each k-1 ordinal value we fit a binary classification problem
            binary_y = (y > unique_class[i]).astype(np.uint8) # transform to a binary label
            clfi = clone(clf) # create a new classifier for this label
            clfi.fit(X, y)    # fit on it
            clfs[unique_class[i]] = clfi    # store into the dictionary
    return clfs

def mdl_predict_proba(X, y, clfs):

    unique_class = np.sort(np.unique(y))
    clfs_predict = {k:clfs[k].predict_proba(X) for k in clfs}

    predicted = []
    for i, y in enumerate(unique_class):
        if i==0:
            # v1 = 1 - Pr(y > V1)
            predicted.append(1 - clfs_predict[y][:,1])
        elif y in clfs_predict:
            # Vi = Pr(y > Vi-1) - Pr(y > Vi)
            predicted.append(clfs_predict[y-1][:,1] - clfs_predict[y][:,1])
        else:
            # Vk = Pr(y > Vk - 1)
            predicted.append(clfs_predict[y-1][:,1])
    return np.vstack(predicted).T

def mdl_predict(vstack):
    return np.argmax(vstack, axis=1)
```

Appendix -- Non-English Detection

Song name: any non-English?

```
new_name = [nm.translate(str.maketrans('', '', string.punctuation)) for nm in data_all['name']]
len(new_name)
```

```
174389
```

```
# I tried to loop through the name word by word but the detection didn't work well and it runs foever
# So now I'm trying to loop just by the name
from langdetect import detect
from langdetect.lang_detect_exception import LangDetectException

nonen_flag = []
for item in new_name:
    try:
        if detect(item) != 'en':
            nonen_flag.append(1)
        else:
            nonen_flag.append(0)

    except LangDetectException: # Pass if there is a line break
        #print('line break')
        nonen_flag.append(0)
        #break
```

Appendix -- Application Use Case

```
input_data['cluster_labels'][input_data.artists == '2NE1']
```

```
29    5
Name: cluster_labels, dtype: int64
```

```
In [56]: artist_popularity.sort_values(by=['predicted_popularity'], ascending = False).head()
```

Out[56]:

artists_x	name	predicted_popularity	popularity_group
Dailey & Vincent	Cumberland River	5.747432	6.0
Funkadelic	Maggot Brain	5.649359	6.0
Jwonder	Dirty S Hwy	5.592359	6.0
Interpol	C'mere	5.563474	6.0
Michael Jackson	Liberian Girl - 2012 Remastered Version	5.560343	6.0

Appendix

	Independent Features	VIF Scores
0	acousticness	6.088960
1	danceability	18.434931
2	energy	18.693486
3	instrumentalness	2.043525
4	liveness	4.578157
5	loudness	12.514755
6	speechiness	2.872620
7	tempo	30.307536
8	valence	8.542654
9	popularity	2.611204
10	key	3.489770
11	mode	3.973770