Classification of Popular Songs Using Spotify Data

Problem Statement

Spotify has a massive amount of data about songs. Some of those songs are popular and some of them are not. I will try to predict popular songs by using Spotify dataset.

Spotify Dataset

The dataset contains more than 170.000 songs and 19 feature columns collected from Spotify Web API.

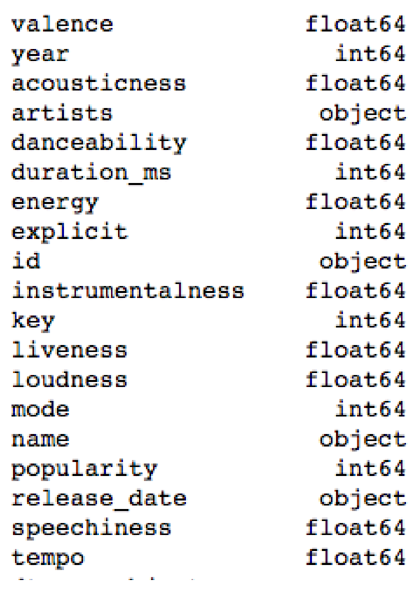
Features

Details of the feature columns are available in appendix.

Summary

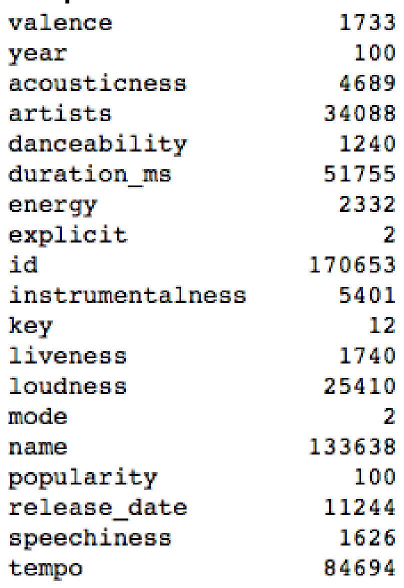
- 4 columns are objects others are numeric(float, int)
- Release date column is very similar to year column but it is more detailed for recent periods
- id and name columns look unnecessary
- Artists column have 34088 unique values. Cardinality is too high
- Except our target columns popularity we have 14 numeric features
- Popularity varies between 0-100
- Explicit and mode columns are boolean
- There are no null values.

Dtypes



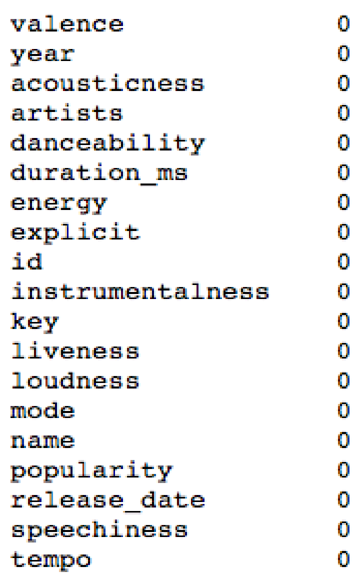
```
valence              float64
year                   int64
acousticness         float64
artists               object
danceability         float64
duration_ms            int64
energy               float64
explicit               int64
id                    object
instrumentalness     float64
key                    int64
liveness             float64
loudness             float64
mode                   int64
name                  object
popularity             int64
release_date          object
speechiness          float64
tempo                float64
```

Unique Values



```
valence               1733
year                   100
acousticness          4689
artists              34088
danceability          1240
duration_ms          51755
energy                2332
explicit                 2
id                  170653
instrumentalness      5401
key                     12
liveness              1740
loudness             25410
mode                     2
name                133638
popularity             100
release_date         11244
speechiness           1626
tempo                84694
```

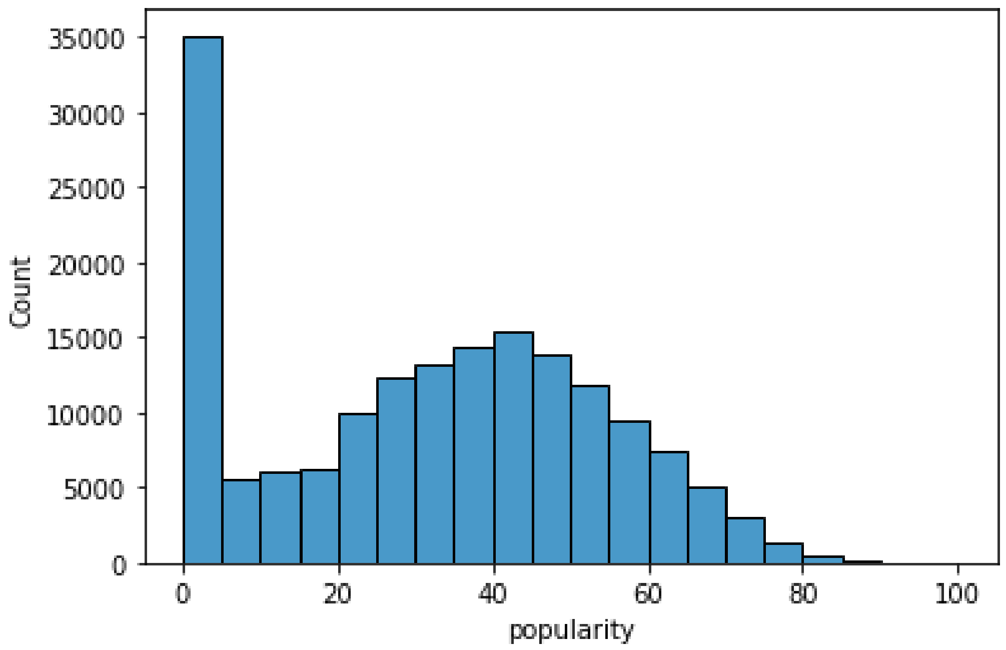
NaN values



```
valence             0
year                0
acousticness        0
artists             0
danceability        0
duration_ms         0
energy              0
explicit            0
id                  0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
name                0
popularity          0
release_date        0
speechiness         0
tempo               0
```

We have several features as type object, the problem is popularity is strictly connected with year. And the year feature is disrupting the model because of high cardinality and lack of causality.
If the idea is creating a model that can predict a new song would be popular or not, year is not beneficiary.

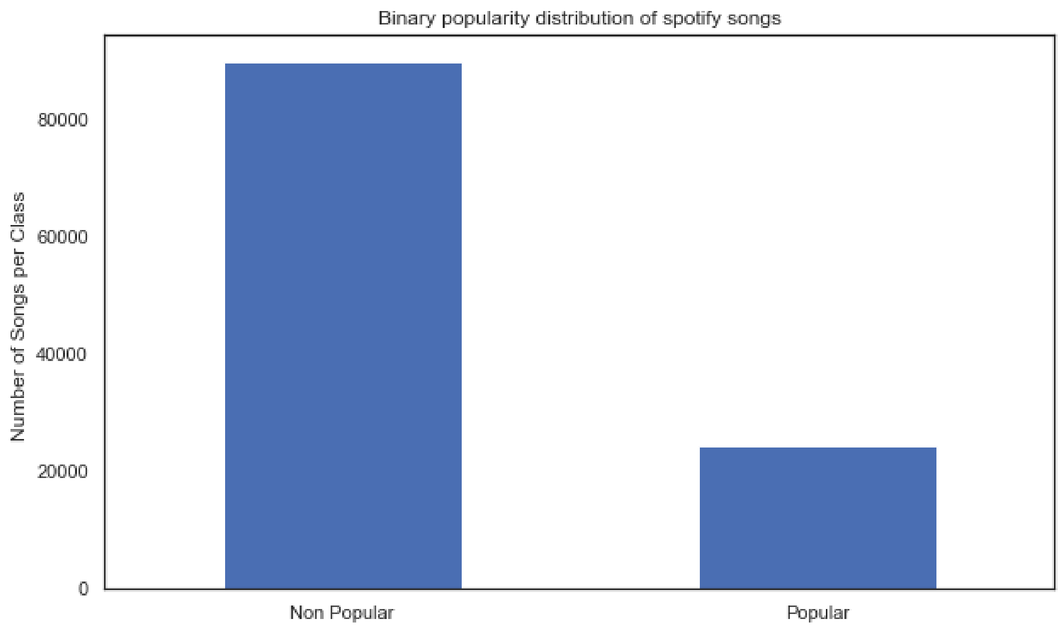
Target Distribution



According to histogram above, our target follows normal distribution after the popularity score of 20. And if we define popularity as the being higher than the average; popularity score of 50 could be the point of discrimination for a binary classification problem.

After transforming our target column into binary form;

```
Percent Distribution of Potential Popular class-
0    78.720799
1    21.279201
```

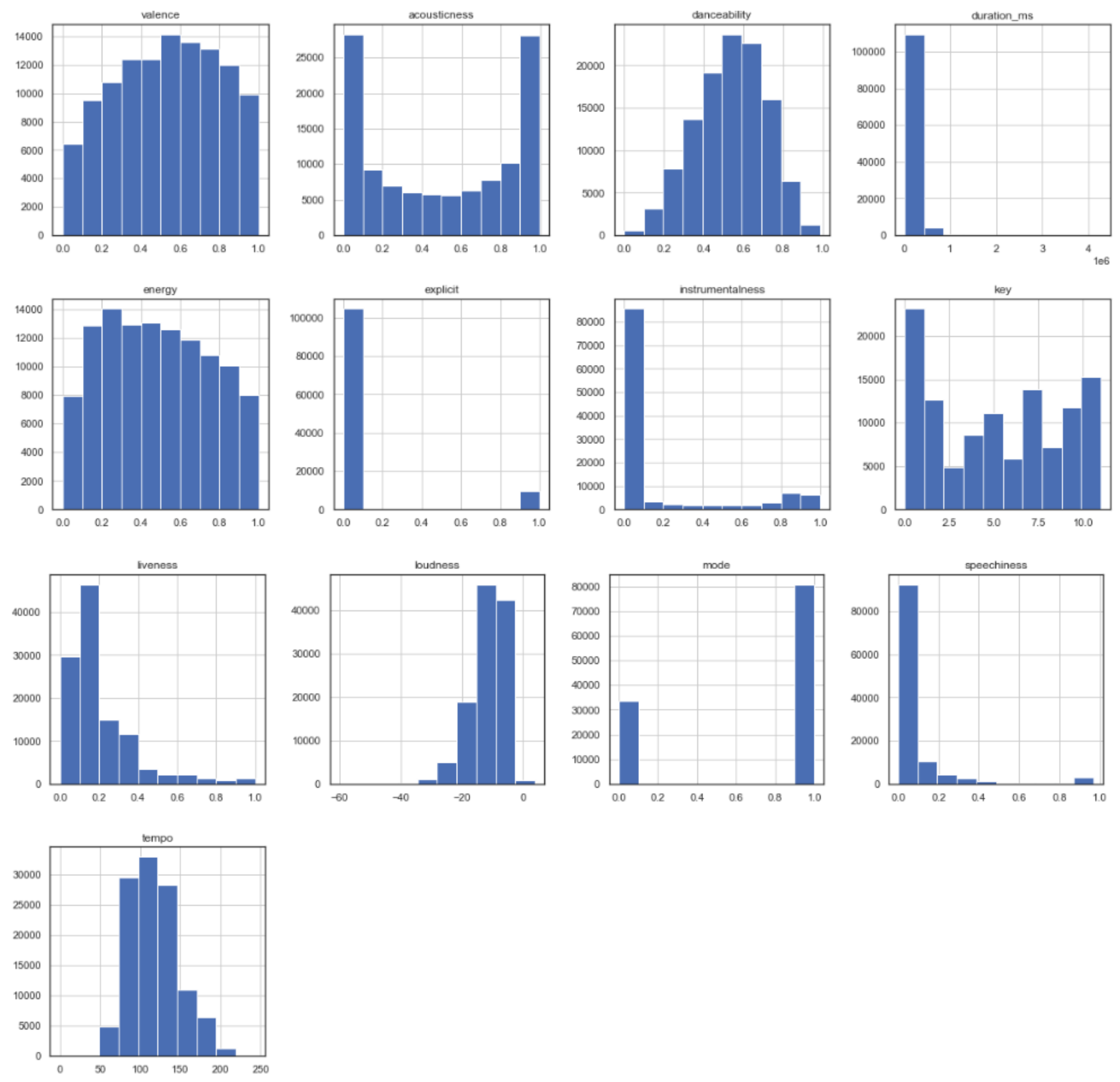


We need to split data before starting EDA because including test data into our analysis creates bias.

EDA (Explanatory Data Analysis)

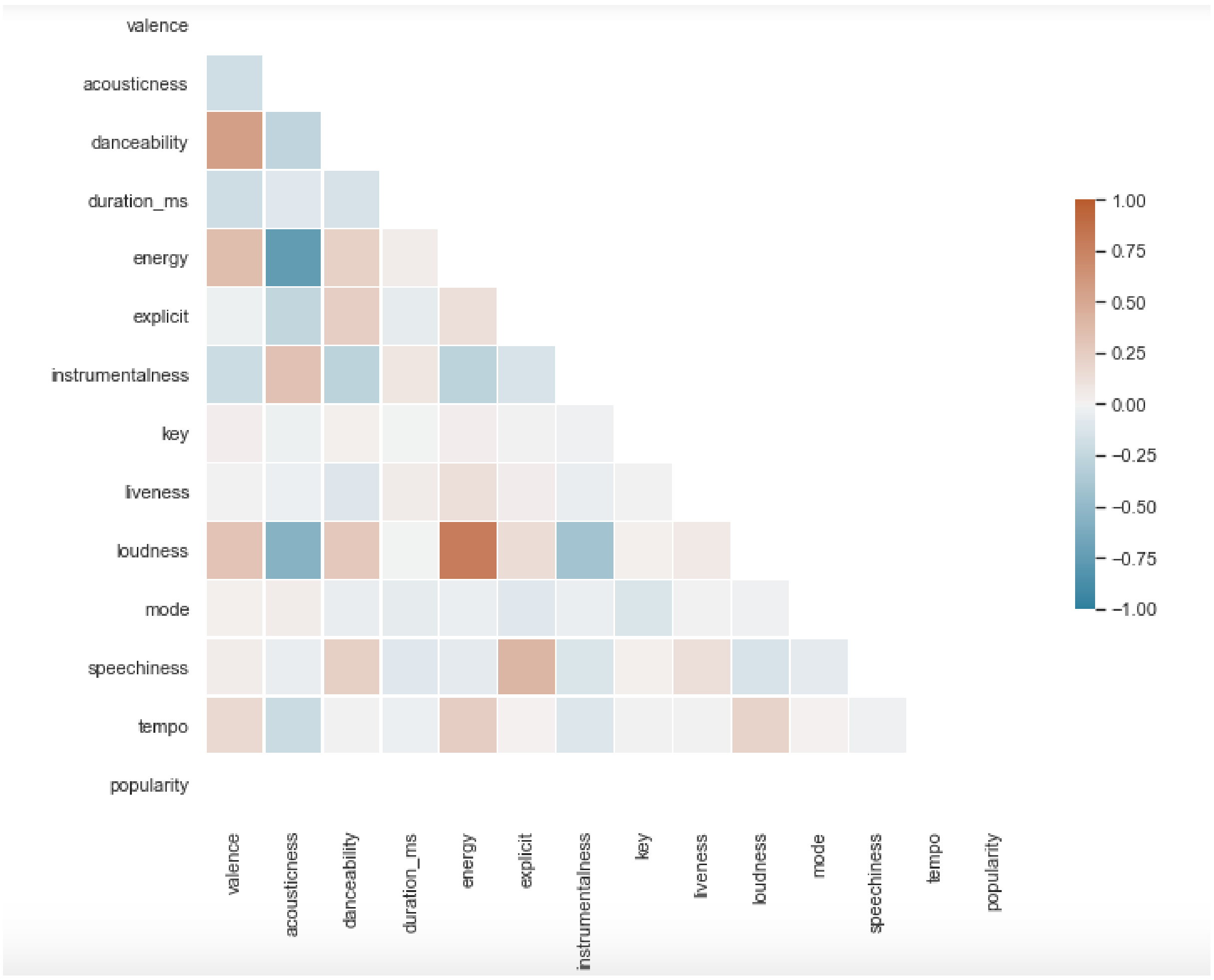
Histograms of Dataset

Histograms help us to understand the distribution shape of a feature.
 - According to distribution shape, we might use the feature in an algorithm that has linearity or normality assumption, Or, we can implement normalization or transformation (log, exp etc.) to use that feature.
 - For example, danceability feature has a normal distribution but speechiness shows kind of log distribution. So, if you are planning to implement a linear regression model, you need to transform second feature and check the linearity assumption.
 - We are working on a classification problem, so we don't have to follow this pipeline. However, implementing these methods could still boost our model performance.



Correlation check with Heatmap

Heatmap of correlations would be beneficial to understand the relations between our features and target.



We can observe the level of correlation between our features and target. However, the problem with correlation is the assumption of linearity. Correlation coefficient measure the linear relationship between features, but our features would have for example quadratic relationship but It is still useful to check the correlation.

Initial Results

Two different algorithms have been used during modeling phase. We will use these two algorithms to compare before and after the feature engineering.

Logistic Regression (Solver: lbfgs)

Train Accuracy : 0.7880651057837795
Train AUC : 0.7496114644341656
Test Accuracy : 0.789313871723844
Test AUC : 0.750144952140415

Logistic regression model doesn't work well compared to gradient boosting. The root cause would be the shape of the optimization function. If we try another solver which is able to handle non-con vex optimization problems, it would perform better.

Logistic Regression (Solver: Newton-cg)

Train Accuracy : 0.8122042733323421
Train AUC : 0.8078983389660184
Test Accuracy : 0.8123623836920236
Test AUC : 0.8071837061259486

Using newton-cg solver, logit is able to achieve better results in terms of both accuracy and AUC scores.

Gradient Boosting

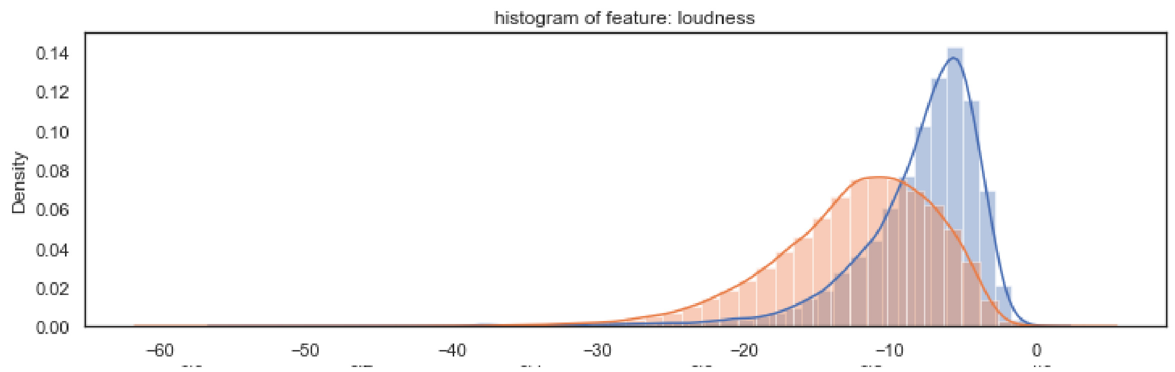
Accuracy : 0.8289180230371621
AUC : 0.8427904700895278
Accuracy : 0.8268698060941828
AUC : 0.8377180013973688

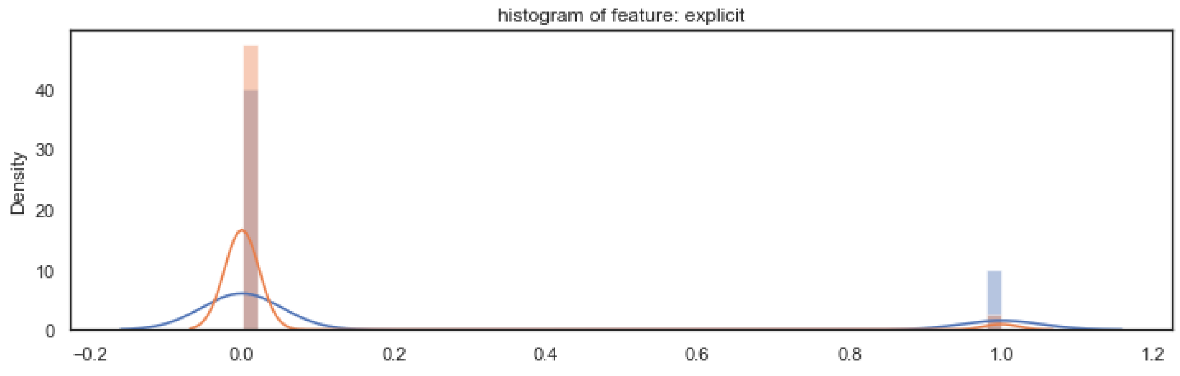
Ensemble algorithms handle different feature structures better.

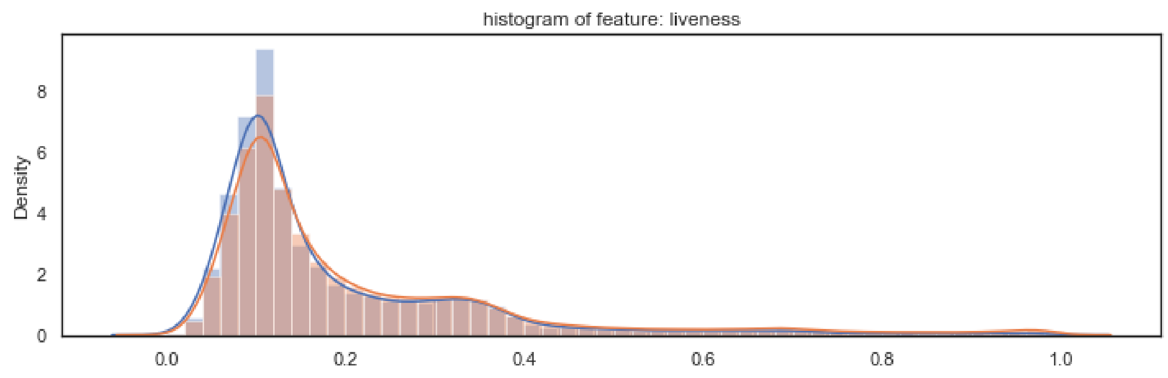
Improvements

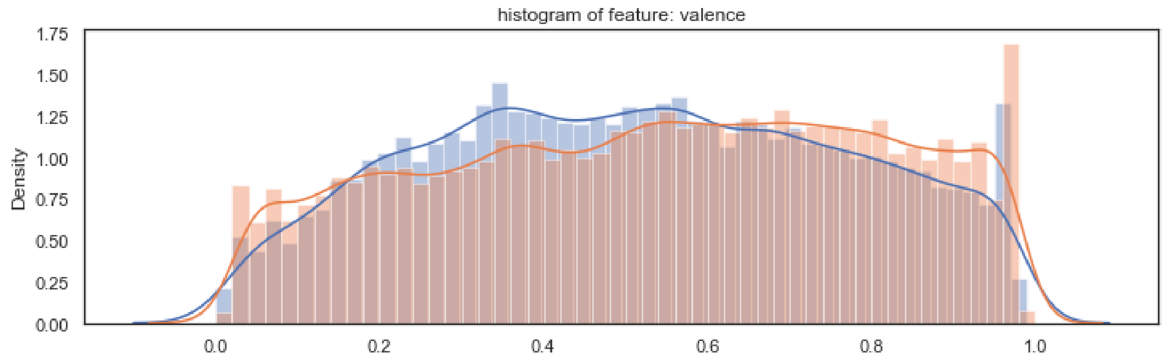
We should analyze each feature individually and implement single factor analysis with respect to target classes to improve model performance.

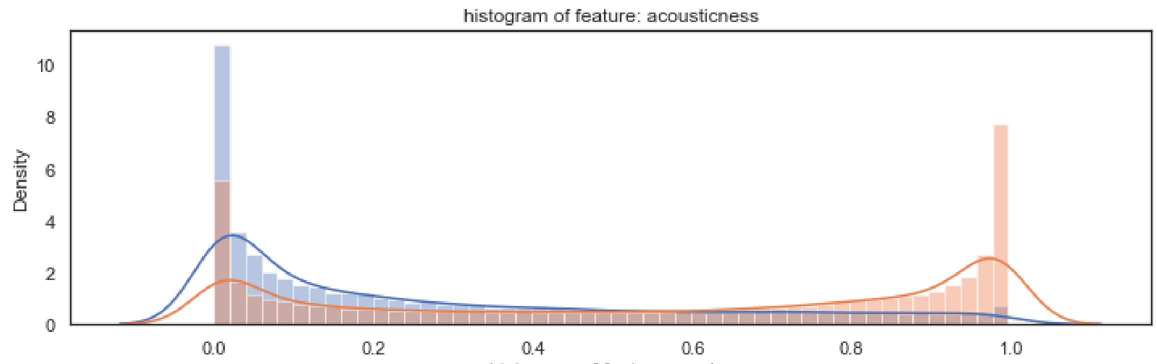
Histograms of each feature regarding target classes

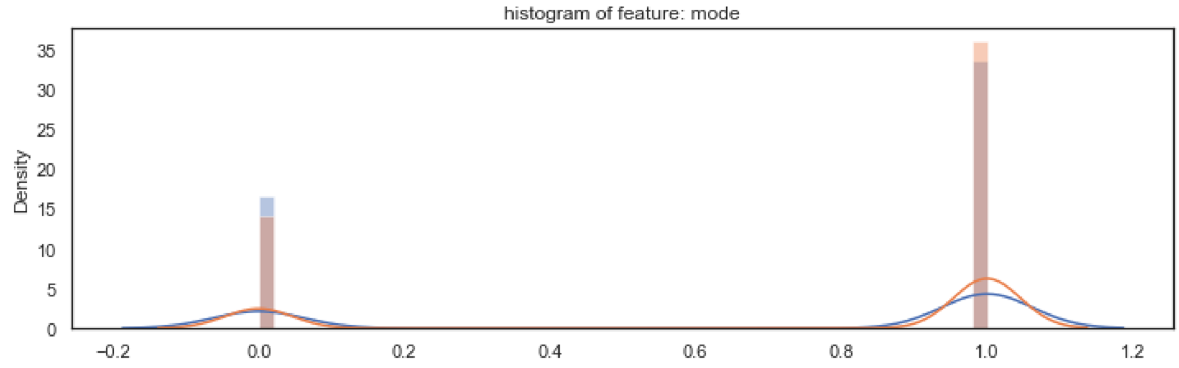


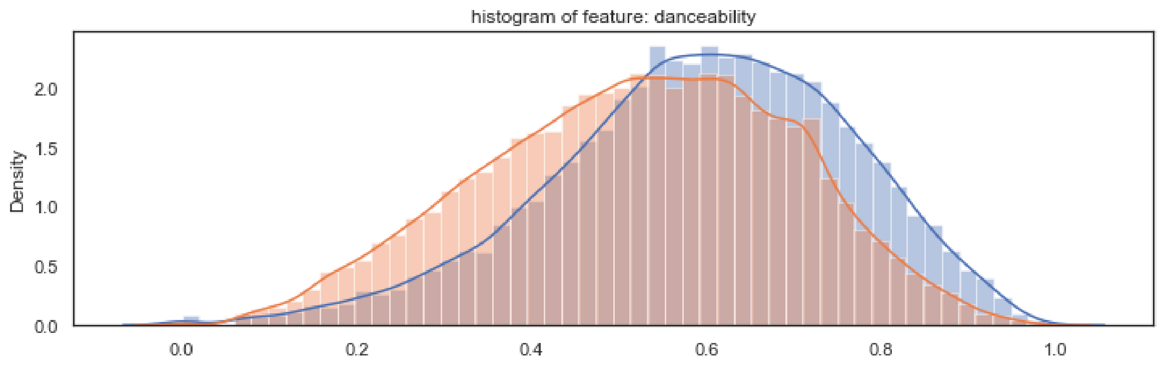


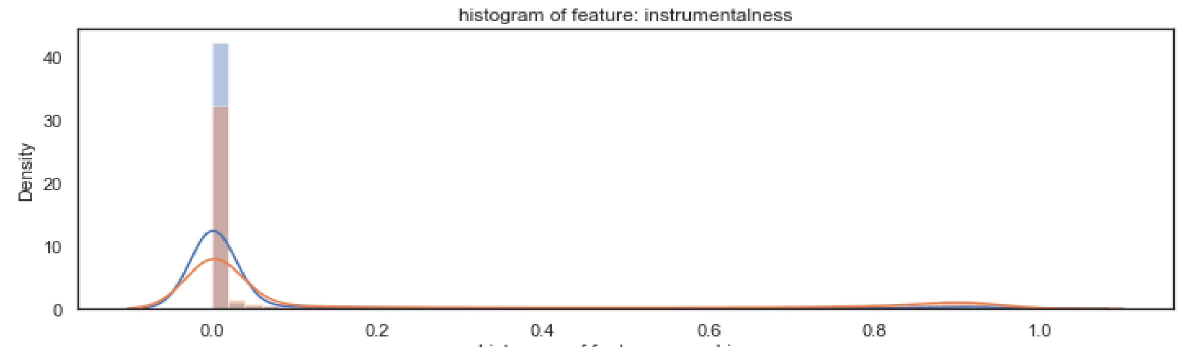

histogram of feature: liveness
Density
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0

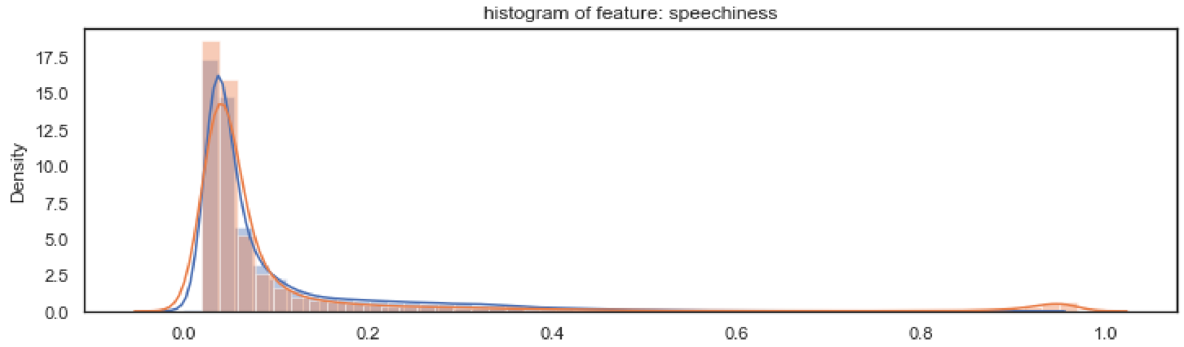

histogram of feature: valence
Density
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.2
0.4
0.6
0.8
1.0

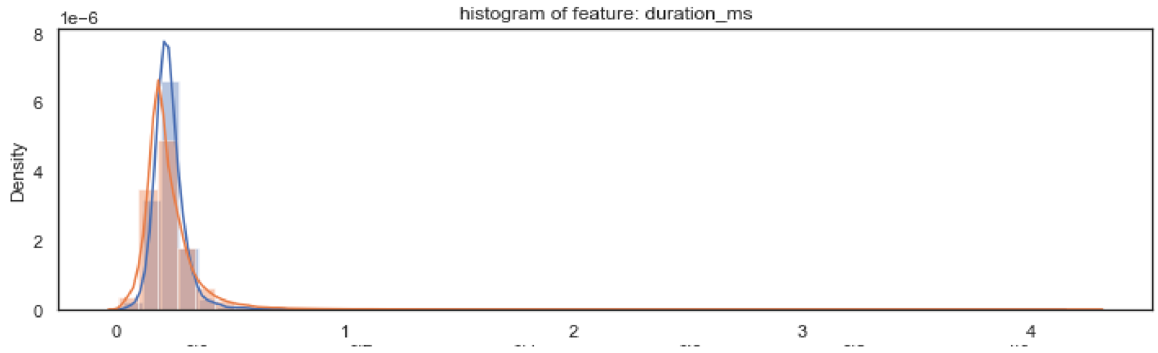

histogram of feature: acousticness
Density
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

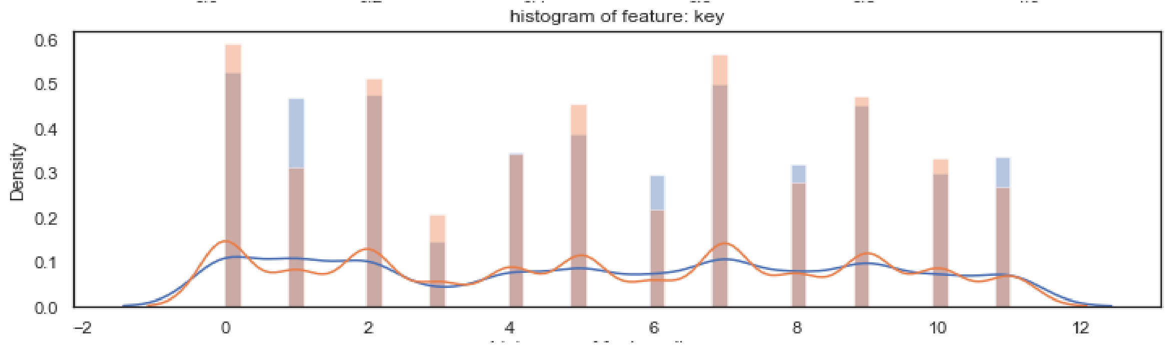

histogram of feature: mode
Density
0
5
10
15
20
25
30
35
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

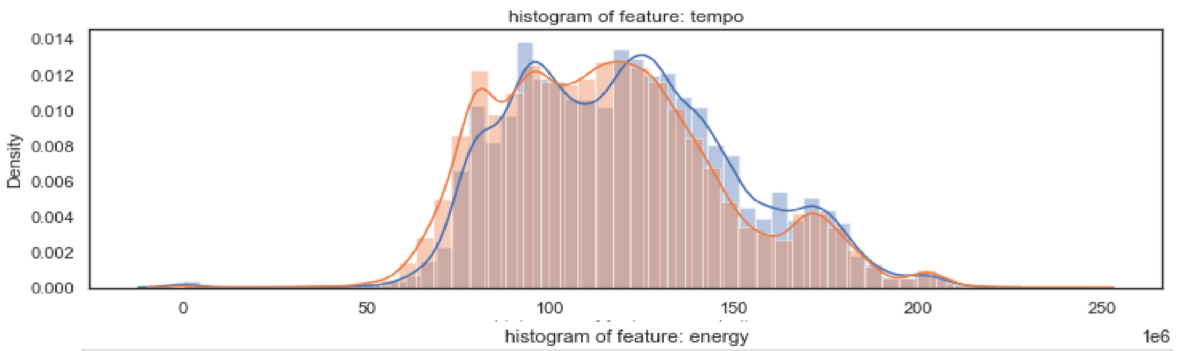

histogram of feature: danceability
Density
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0

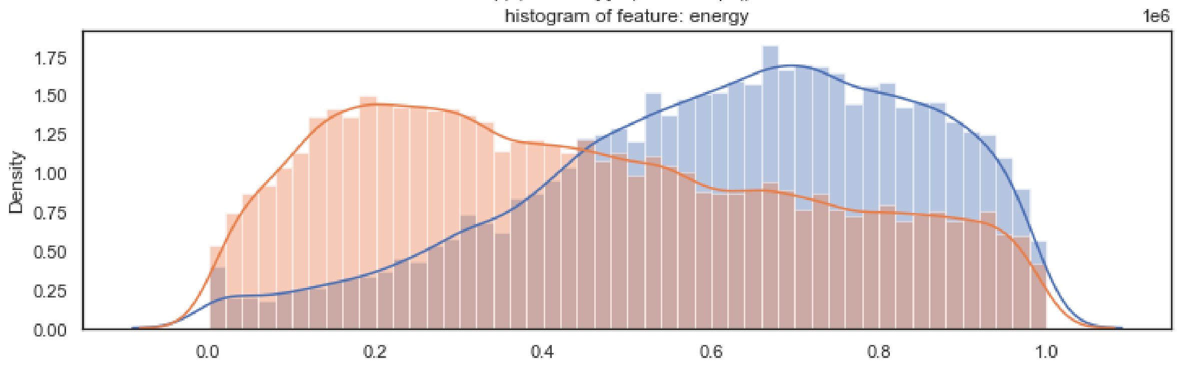

histogram of feature: instrumentalness
Density
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0


histogram of feature: speechiness
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.0
0.2
0.4
0.6
0.8
1.0

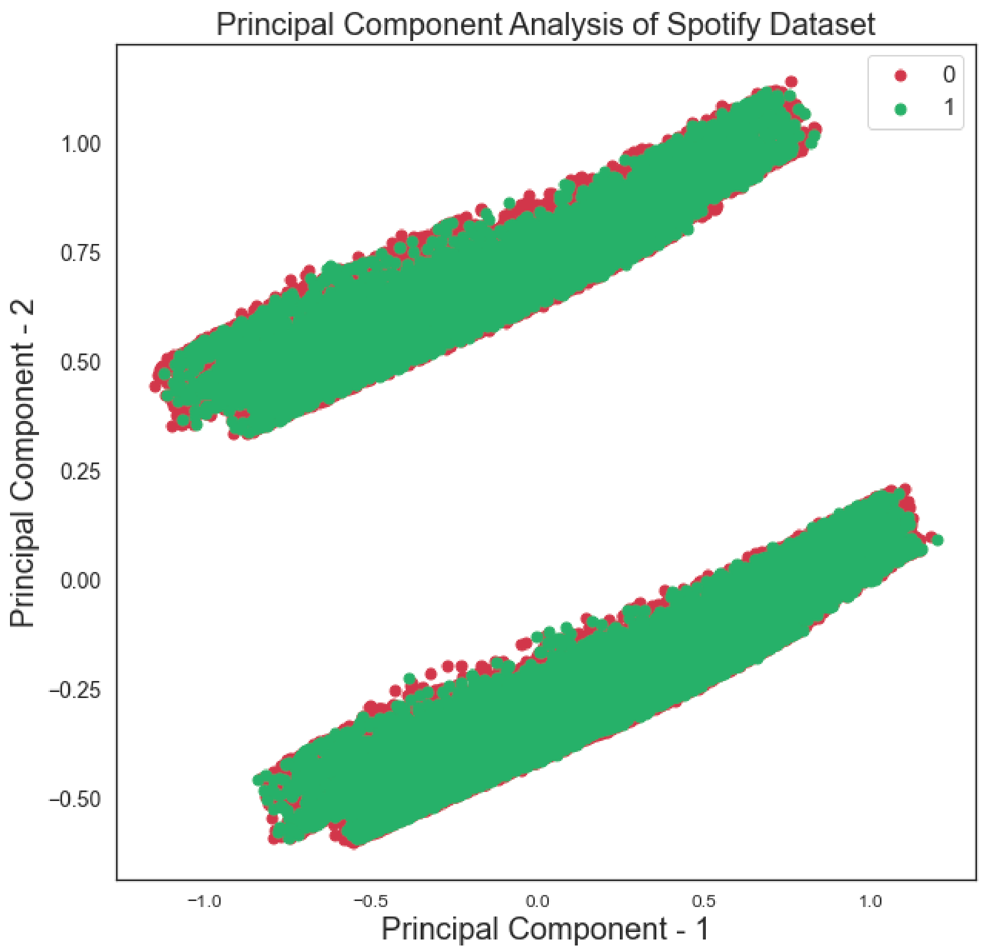

histogram of feature: duration_ms
1e−6
Density
0
2
4
6
8
0
1
2
3
4


histogram of feature: key
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−2
0
2
4
6
8
10
12





Data visualization with 2 Dimensional PCA



PCA graph shows 2 distinct clusters but unfortunately, these 2 clusters do not discriminate our labels. Graph still shows a overlapping structure between labels. We need to implement feature engineering.

Feature Engineering

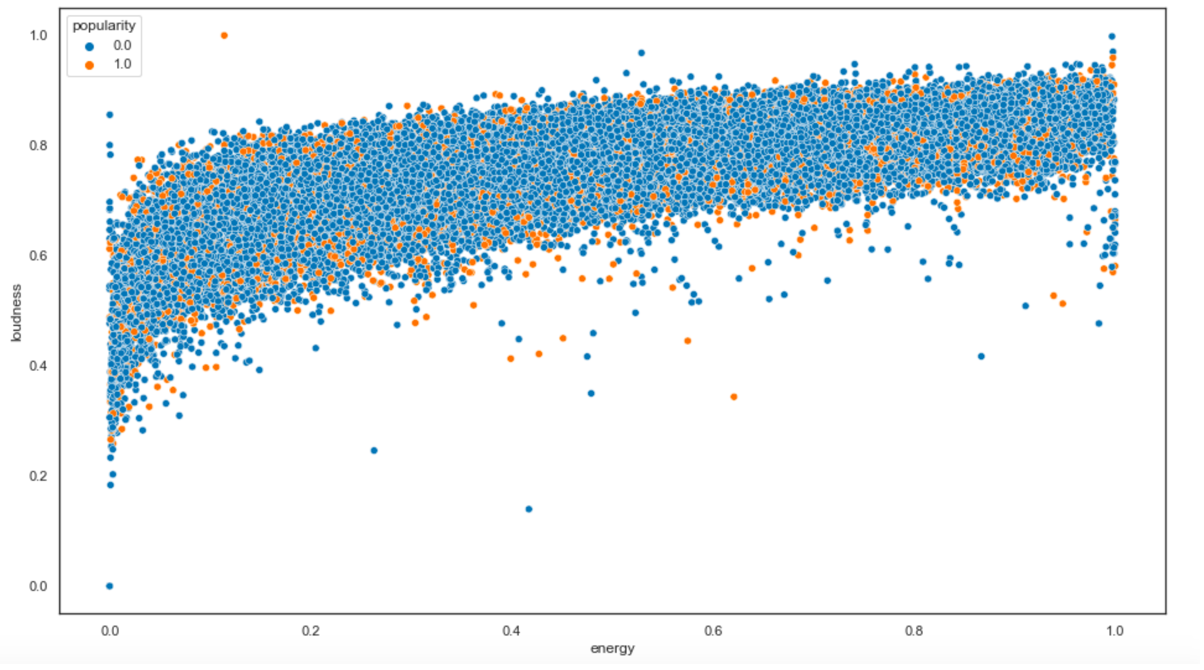
Log transformation and Standardization

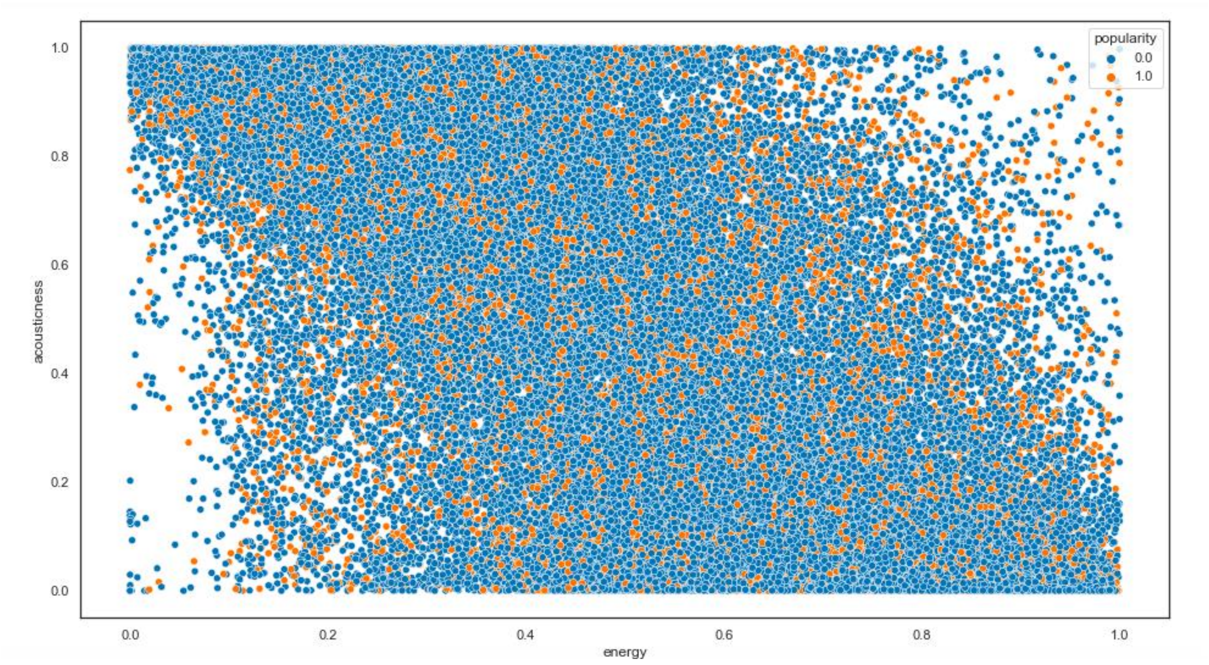
According to histogram graphs, some of our features show high skewness. Normality is not an assumption of our selected algorithms but linear relationships are easy to observe and detect. And scaling operation also creates balance between features in terms of upper and lower boundries and makes it easier to interpret using scaled features coefficients. In addition, it accelerates the model calculation process.
We transformed four features using log transformation: **Acousticness, duration_ms, speechiness, instrumentalness.** Exponential transformation also experimented but there are no fundamental differences in terms of distribution.
After log transformation, all features have been scaled between 0-1 using min-max scaler.

PCA transformation for highly correlated features

We already know that energy and loudness features have high positive correlation.(0.782089)
After log transformation, acousticness and energy features showed high negative corralation(-0.696303)

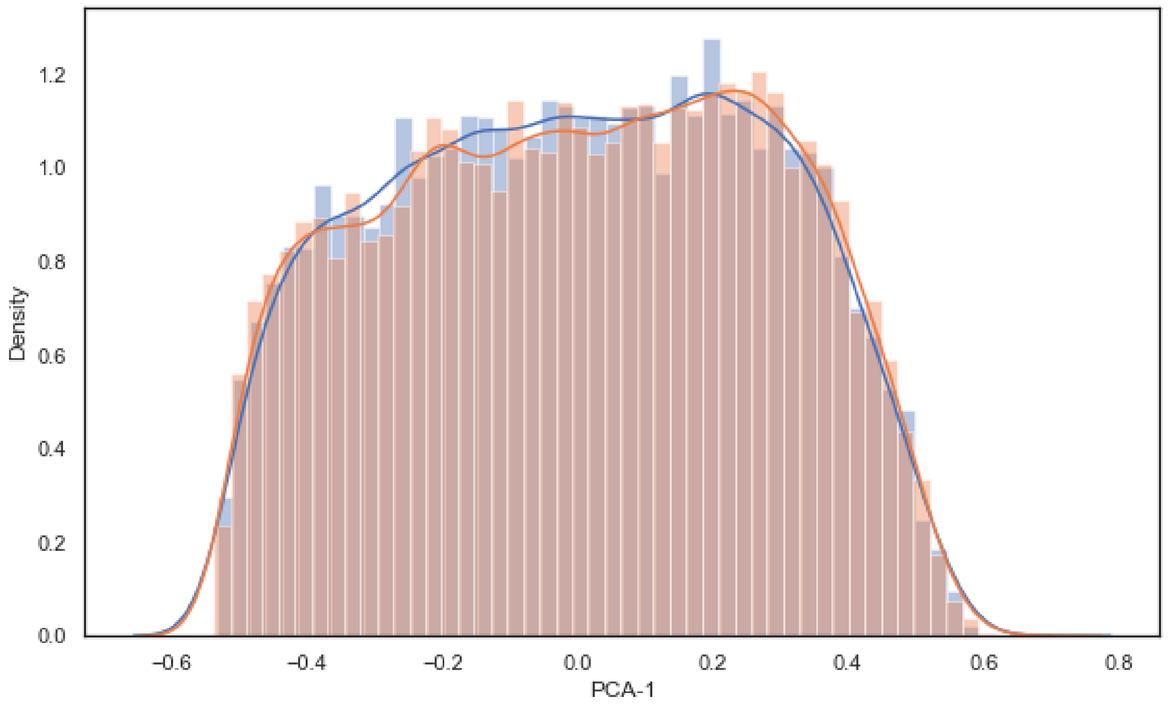
However, our dataset is too big to observe these relationships.





PCA of highly correlated features

I tried to implement a PCA transformation on 2 feature sets: energy & loudness and energy and acousticness. The result is not satisfactory, one of the results is observable below. There is an overlapping between labels in terms PCA-1 merged feature. Of course statistical tests would answer the question of discriminative power but it doesn't look a good feature. And it also decreased the models ROC and Accuracy scores.



Discretization

According to our graphs, there are no highly discrete features. However, energy column looks like it would show a good performance after discretization so I divide column into 2 and gave a new score for each split. The new feature was not performing well because it contains more information while it is in continuous form.

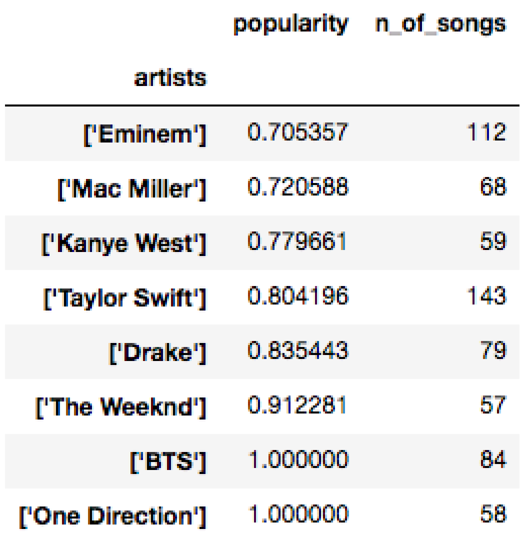
OneHot Encoding

Key feature has no discriminative power when it used as ratio scaled feature. To extract more value from the feature, we can group specific values, or just implement one hot encoding. I experimented both. The column has a low cardinality of 13 unique values. The result of one-hot encoding was not successful; there was no increase in scores and feature importance of encoded columns was almost zero except 3 of them: key_1, key_6 and key_11. So, I created a new binary feature(key_16110) using 3 of these new features. The result has not changed and its feature importance was not significant (0.000714765). The conclusion is key feature does not consist any explanatory power and I dropped it from the dataset.

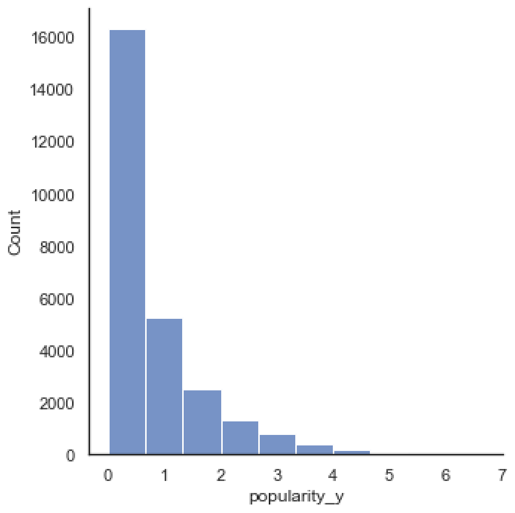
Text mining

Most of the classification algorithms are not able to process string columns. If data consists of raw string columns, it is possible to boost the performance by mining these string columns. Our spotify dataset consists of artists column. We can investigate and try to create a few new features using it. About the column; row values are lists of singers and there are 26896 unique singer value.more info

I have created a summary data frame with 2 columns: total number of songs of a singer and popularity of a singer aggregated by songs. A sample is below. And also the distribution of artist popularity shows a log distribution.



	popularity	n_of_songs
artists		
['Eminem']	0.705357	112
['Mac Miller']	0.720588	68
['Kanye West']	0.779661	59
['Taylor Swift']	0.804196	143
['Drake']	0.835443	79
['The Weeknd']	0.912281	57
['BTS']	1.000000	84
['One Direction']	1.000000	58



After calculating summary table, I need to find a pattern and implement a new feature. I tried to assign scores for a combination of popularity and number of songs. I also tried creating buckets according to the balance between 2 features. However, after several attempts; I found that using a flag for popular and productive singers is performing well compared to other methods.

I used thresholds of 0.5 for popularity and 10 for number of songs and create a new flag: popular artist flag (pop_artist_flag). After including this feature into algorithms our scores increase significantly both for logistic regression and gradient boosting.

Final Results

Logistic Regression:

```
train Accuracy : 0.8268102189142622
train AUC :  0.8202665316741977
test Accuracy : 0.8257866325733362
test AUC :  0.8172146806474958
```

Gradient Boosting

```
train Accuracy :  0.8398943474115991
train AUC :  0.8535843755057805
test Accuracy : 0.8365295830669792
test AUC :  0.8470367979857615
```

We end up with 11 features with a test roc_auc score of 84.7.

Appendix

Data

https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

Feature Explanations

• **id** (Id of track generated by Spotify)

• **artists** (List of artists mentioned)

• **name** (Name of the song)

• **genres** (Genre of the song)

• **year** (Ranges from 1921 to 2020)

• **duration_ms** (Integer typically ranging from 200k to 300k)

• **mode** (0 = Minor, 1 = Major)

• **explicit** (0 = No explicit content, 1 = Explicit content)

• **key** (All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on…)

• **popularity** : The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are.Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently. Artist and album popularity is derived mathematically from track popularity. Note that the popularity value may lag actual popularity by a few days: the value is not updated in real time.

• **acousticness** : A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

• **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. A value of 0.0 is least danceable and 1.0 is most danceable.

• **energy** : Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

• **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.” The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

• **valence** : A measure from 0.0 to 1.0 describing 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.

• **liveness** : Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

• **loudness** (Float typically ranging from -60 to 0)

• **speechiness** : Speechiness 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. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.