**Exploratory Analysis**

**The Basic Descriptive Statistics**

In total, our dataset has 24 attributes. Among them, 7 are nominal categorical, including trackname, trackID, movie name, movie releasing year, movie genre, etc. The other 17 are numerical, indicating characteristics of the soundtrack and the movie it comes from.

|  |  |
| --- | --- |
| **Nominal Attributes** | Acousticness, danceability, duration\_ms, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time\_signature, valence , popularity, Movie\_gross, Movie\_rate, Movie\_runtime |
| **Categorical Attributes** | Track\_name, Track\_ID, Album\_name, Album\_ID, Movie\_name, Movie\_genre, Movie\_yr |

In this part, we implement basic statistical description on the following 11 attributes:

'acousticness', 'danceability', 'duration\_ms', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'popularity', 'Movie\_rate’

We include these attributes as they reflect the characteristics of both the soundtrack itself and the movie, and they are all continuous. We basically generated the mean, median, and standard deviation for these attributes. The result is shown as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **attributes** | **mean** | **median** | **standard\_deviation** |
| acousticness | 0.500077 | 0.532 | 0.366199 |
| danceability | 0.452396 | 0.47 | 0.21768 |
| duration\_ms | 201255.8 | 192726 | 115187.7 |
| energy | 0.439615 | 0.414 | 0.294995 |
| instrumentalness | 0.386196 | 0.109 | 0.412345 |
| liveness | 0.193272 | 0.12 | 0.170154 |
| loudness | -13.5241 | -12.122 | 7.523836 |
| speechiness | 0.097097 | 0.0436 | 0.155745 |
| tempo | 113.2979 | 111.772 | 32.81185 |
| popularity | 22.69279 | 21 | 17.65572 |
| Movie\_rate | 7.567121 | 7.5 | 0.422295 |

**Outlier Detection**

To ensure that the models we build in further analysis make sense for the whole experimentation, we continue our exploratory analysis by looking into whether there exist any outliers. During Project 1 we have already checked for each attribute for detecting any attributes that are out of range. In this part, two methods are implemented to detect outliers: z-score outlier detection and DBSCAN outlier detection.

Z-score Outlier Detection

First, we will go through the z-score approach for desired columns to check whether there exist any outliers in any attributes. The z-score is a way of describing a data in terms of its relationship to the mean and the standard deviation of a group of points.

The intuition behind the z-score method of outlier detection is that, we center the data and rescale them, anything that is too far from zero (the threshold is set according to requirements. Here, our threshold is a z-score of 5.5 or -5.5) will be considered as an outlier.

The following columns are being considered as might contain outliers:

'acousticness', 'danceability', 'duration\_ms', 'energy', 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'speechiness', 'tempo', 'valence', 'popularity', 'Movie\_gross', 'Movie\_runtime'.

The numbers of outliers in each column listed above are shown as follows:

|  |  |
| --- | --- |
| **Attribute\_Name** | **Outlier\_Number** |
| acousticness | 0 |
| danceability | 0 |
| duration\_ms | 54 |
| energy | 0 |
| instrumentalness | 0 |
| key | 0 |
| liveness | 0 |
| loudness | 0 |
| mode | 0 |
| speechiness | 99 |
| tempo | 0 |
| valence | 0 |
| popularity | 0 |
| movie\_gross | 140 |
| movie\_runtime | 18 |

We see that, in z-score outlier detection, most columns do not get any outliers according to our criteria, while some columns have outliers, specifically, ‘duration\_ms’, ‘speechiness’, ‘Movie\_gross’, ‘Movie\_runtime’ have outliers.

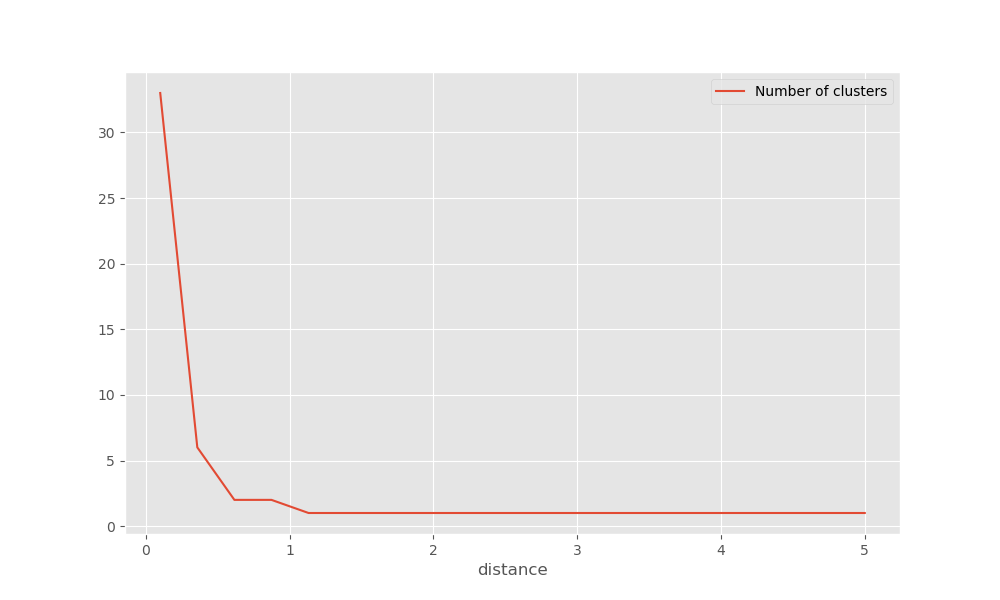
DBSCAN Outlier Detection

We then apply DBSCAN outlier detection approach to our dataset.

DBSCAN is a popular density-based clustering algorithm used often in outlier detection. To be more precisely, this approach is to look for noise data.

In this part, we will not scatter all attributes for DBSCAN algorithm. Based on importance/influence each column has on the predictive models which will be discussed later in this report, we apply the DBSCAN outlier detection on the following attributes: ‘tempo’, ‘loudness’, ‘instrumentalness’. These are columns that prove to have the ‘biggest’ effect on the predictive results of popularity, thus we care about them more for outlier detection.

We need to set the eps parameter in DBSCAN algorithm. In order to get the best distance, we first apply the function ‘getDistance’. The result it generated is exhibited as follows:



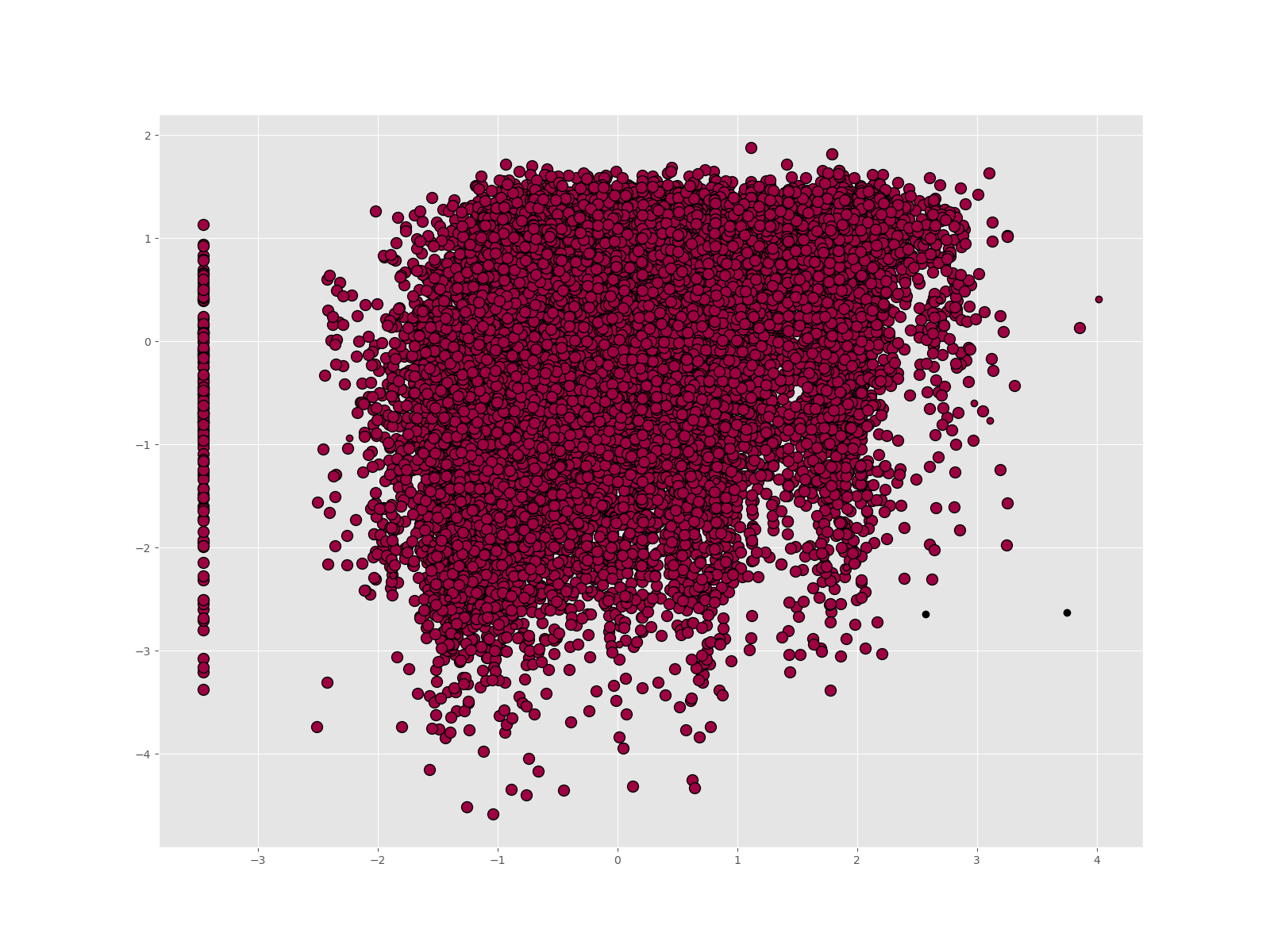
According to the result, we set our eps to be 1. The min samples included in DBSCAN process is 7.

This will return all outliers’ index.

20493

22047

Besides, we have generated a scatter plot showing all the outliers:



The outliers/noise data points are marked black. In total, DBSCAN detected 2 outliers.

**Deal with Outliers**

In summary, we have detected for outliers/noise in two ways: for one column at a time and by scattering method dealt with several columns. In later analysis, we will NOT exclude these outliers detected due to the following reasons:

* Comparing with the whole dataset (26321 records), the outliers are relatively a very small proportion.
* In previous work, specifically, the scope check part, we have already removed those data that are out of range according to the scope listed on Spotify official website. This indicates that all data actually make sense as soundtracks could have very diversified characteristics, and there are no specific or general rules defining a track. So the outliers detected by z-score, which is based on mean and standard deviation, are soundtracks that contain unusual characteristic values, however, this is not something wrong and might reflect interesting findings. It is not necessary to remove them.
* Outliers detected by DBSCAN are actually noise data. Again, since the data have passed the scope check, there is nothing technically wrong with these data. They might have effect on the accuracy of predictive models later to be discussed, however, considering the size is small, there is no need to exclude them.

**Columns Containing Null Values**

In the cleaning phase of Project 1, we examined whether there existed null values in each attribute in our raw dataset.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 4: Feature Quality Check Table** | | | | | | | |
| **Attribute** | Acoustic-  ness | Danceability | Duration  \_ms | Energy | Instrumen- talness | Key | Liveness |
| **Null Value Rate** | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 |
| **Attribute** | Loudness | Mode | Speechi-  ness | Tempo | Time  \_signature | Valence | Popularity |
| **Null Value Rate** | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0 |
| **Attribute** | Analysis  \_url | id | track\_href | type | uri | track\_id |  |
| **Null Value Rate** | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0.001545 | 0 |  |

As a result, we saw there are null values in the following attributes in our raw dataset:

'acousticness', 'danceability', 'duration\_ms', 'energy', 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'speechiness', 'tempo', 'valence', ‘Analysis\_url’, ’id’, ‘track\_href’, ‘type’, ‘uri’.

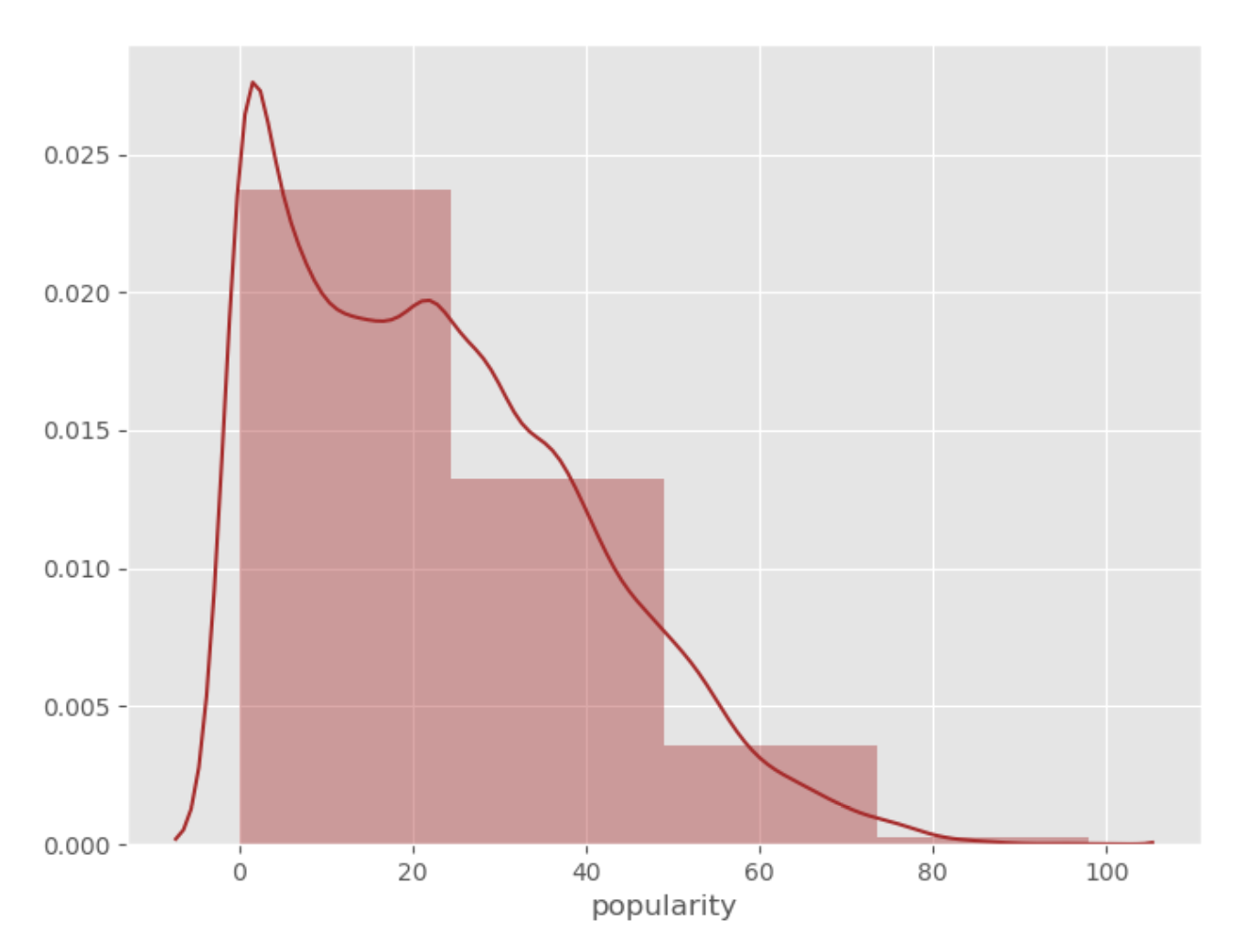
Noticing that the null value rate (records with null values/total size of records) are all identical. By further check the raw data, we found that, these missing values are within the same records. This indicating that there might due to lack of information of these soundtracks that resulting in null values in characteristics. Though the proportion is rather small, we removed those records as they provide us with very little information that is useful to our analysis.

**Binning Attribute of ‘Popularity’**

In this phase, we will bin the attribute ‘popularity’ with equal-width binning method.

The reason we chose to bin this attribute is that, the goal of our project is based on studying the soundtrack’s popularity, we want to bin the attribute ‘popularity’. This is a numeric column with continuous values ranging from 0-100. Spotify use several algorithms to determine the popularity, but in general the more a song is played the higher its popularity. The popularity rating is based on total number of plays compared to other tracks as well as how recent those plays are. Most Popularity views have 12 bars to indicate the popularity.

Let’s first take a look at the distribution of popularity:



We see that in our dataset, most soundtracks got a popularity of 0-40. The distribution of different popularity is not average, with most songs lie in a relatively low area of popularity. We want to analyze whether the soundtrack is popular or not according to their popularity scores, so the best strategy that make sense is to use equal-width method. Other methods, for example, equal-depth method, will result in some category containing songs that are very different in popularity.

Considering our popularity distribution and comparing with other songs outside our dataset, we generally use equal-width binning method for our dataset, the width for each bin in our dataset is 25 except for the last bin (popularity < 20). In conclusion, we set the binning strategy as follows:

|  |  |  |
| --- | --- | --- |
| **Popularity range** | **Category** | **Category number** |
| popularity >= 70 | Very popular | 4 |
| 45<=popularity<70 | Popular | 3 |
| 20<=popularity<45 | Common | 2 |
| popularity<20 | Not popular | 1 |

The popularity binning will create a new column named ‘pop\_level’ in the new dataset generated.

As a result, we got 259 considered as ‘very popular’ tracks, 3119 ‘popular’, 10279 ‘common’, and 12664 ‘not popular’.