Northeastern University
College of Professional Studies

Spotify Track Popularity – Analysis and Prediction

GROUP 04

Ashutosh Singh

College of Professional Studies, Northeastern University

ALY 6040: Data Mining

Prof. Joseph M. Reilly

Oct 25, 2022

Contents

Abstract

Spotify is a revolution in the music industry since it changed how people listen to music forever. Spotify provides its customers with free music streaming, while a premium subscription may be purchased for extra benefits like no commercials and offline music listening. They may upgrade to Spotify Premium to enjoy exclusive music features like enhanced sound quality and an on-demand, offline, and ad-free music listening experience, as well as manage and share over 80 million tracks, including more than 4 million podcast titles, for free. With 433 million users across 183 regions, including 188 million customers, Spotify is now the most widely used audio streaming subscription service (*Spotify &Mdash; About Spotify*, 2022). Due to the sheer volume of users and material that Spotify offers, it builds up enormous databases of users and the songs those users have listened to, which may include useful trends and information for other businesses like Spotify, record labels, or radio stations. This is the reason for our selection of this data collection. In this report, the analysis looks at the business strategy, business questions, corporate objectives, Data visualization, and Data mining techniques to draw insight from the data. We attempt to evaluate the value of the Spotify data for both Spotify and for scientific reasons by running basic statistics on the complete dataset. To provide the first response to queries like "What are the determinants of popularity?" special focus is paid to song popularity. For those attempting to forecast the success of new songs, the discovery of a model that can capture this relationship and the identification of the traits that are seen to be most crucial in popularizing a song are both extremely intriguing topics. In this report, we will mainly aim to develop a solid model that can be applied to this data, as well as address a few business questions that are more business-related considering the data's noteworthy patterns. Additionally, we would like to thank our lecturer Joseph M. Reilly for his introduction to the various data mining techniques as well as for all his advice and assistance with this research.

Introduction

With over 158 million paid subscribers in 178 countries, Spotify is the industry leader in the online music business. Spotify has gathered a big database of users and the songs they have listened to, which can include trends and relevant information for businesses, due to its large user base and excellent content (*Spotify &Mdash; About Spotify*, 2022). Spotify provides APIs for accessing information from its music database. Spotify is a well-known dataset in the data science industry for learning predictive modeling. We used over 600,000 Spotify tracks with their attributes in our project. Each song has 20 columns, each of which describes a different component of the song. We are attempting to optimize several business challenges and expect that our inquiry will reveal some business-related insights. The objective of this project is to develop a machine learning model that accurately predicts the popularity of songs on the Spotify platform. Spotify is a popular music streaming service with a vast library of songs from various genres and artists. The popularity of a song is a crucial factor for both artists and users, as it determines the visibility, reach, and potential success of a song. So, here are some business questions we would like to have answered:

1. What is the overall trend and impact of notable events on the music industry?
2. Who are the most popular artists in the dataset?
3. What are the most popular songs in the dataset?
4. What types of machine learning models do well on this dataset?
5. What characteristics most clearly characterize a popular song?

This paper will attempt to assess the value of Spotify data. This will be accomplished in the following manner: Data Exploration, Data Visualization, Feature Selection, Data Split: Resampling, Pipeline Building: Data Cleaning, Data Transformation, Feature Scaling and Model Building: Hyperparameter Tuning. We employed logistic regression, random forest, and XGBoost and conducted in-depth research and hyperparameter tunning to address a few business challenges. Additionally, our fine-tuned model may be used to optimize the numerous Spotify issues.

Methods

1. Data Exploration

This collection includes audio metadata for over 600,000 Spotify tracks. Each column focuses on a different facet of the song. The following is a description of them.

- **id**: Spotify's unique identifier for each track (randomly generated alphanumeric string).
- **popularity**: The popularity of a song is measured on a normalized scale of [0-100], with 100 being the most popular.
- **duration_ms**: track duration in milliseconds.
- **explicit**: whether or not the song contains explicit content.
- **artists**: the artist's name.
- **id_artists**: Spotify's unique identification for each artist.
- **release_date**: When the album was released (date format: yyyy/mm/dd).
- **danceability**: indicates a track's suitability for dancing based on a variety of musical aspects such as tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 indicates that it is the least danceable, while 1.0 indicates that it is the most danceable.
- **energy**: Energy is a perceptual measure of intensity and activity that ranges from 0.0 to 1.0. Typically, energetic tracks have a quick, loud, and boisterous vibe to them.
- **key**: The estimated overall key of the track. Integers are assigned to pitches using conventional Pitch Class notation. For example, 0 equals C, 1 equals C/D, 2 equals D, and so on. If no key was found, the value is set to -1.
- **loudness**: The overall volume of a track measured in decibels (dB). Typical values vary between -60 and 0 dB.
- **mode**: The mode of a recording (major=1 or minor=0) denotes the type of scale from which its melodic content is formed.

- **speechiness**: Speechiness is a value from 0.0 to 1.0 that shows how many spoken words are in a track. If a song's Speechiness is above 0.66, it probably has spoken words. If it is between 0.33 and 0.66, it could have both music and words. If it is below 0.33, it does not have any words.
- **acousticness**: a confidence scale ranging from 0.0 to 1.0 indicating whether or not the track is acoustic. 1.0 indicates a high degree of certainty that the track is acoustic.
- **instrumentalness:** The number of vocals in the song is represented by the instrumentalness, which ranges from 0.0 to 1.0. The closer to 1.0 it is, the more instrumental the music.
- **liveness**: The possibility of an audience being present in the recording ranges from 0.0 to 1.0. Higher liveness numbers indicate a greater likelihood that the track was performed live.
- **valence**: Valence is a scale from 0.0 to 1.0 that describes the melodic positivity expressed by a song.
- **tempo**: A track's overall approximate tempo in beats per minute (BPM)
- **time_signature**: A track's estimated total time signature. The time signature (meter) is a notational convention that specifies the number of beats in each bar (or measure).

First, we examine the dataset's content using the Pandas library:

Table 1

Some of the dataset's initial songs

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	1922-02-22	0.645
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	1922-06-01	0.695
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.434
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	1922-03-21	0.321
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	1922	0.402

energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature
0.445	0	-13.338	1	0.451	0.674	0.744	0.151	0.127	104.85	3
0.263	0	-22.136	1	0.957	0.797	0	0.148	0.655	102.01	1
0.177	1	-21.18	1	0.0512	0.994	0.0218	0.212	0.457	130.42	5
0.095	7	-27.961	1	0.0504	0.995	0.918	0.104	0.397	169.98	3
0.158	3	-16.9	0	0.039	0.989	0.13	0.311	0.196	103.22	4

Key Findings:

As we can see, the Spotify company defines the terms "id" and "id_artist" in an extremely specific way, so we may delete them later because we cannot use them in our machine learning models. The release_date field has inconsistent data, implying that the formatting is erratic. Some columns, for example, have dates in the format YYYY-MM-DD, whereas others only have the format YYYY. As a result, we must concentrate on the consistency of this data; formatting will follow later. While most qualities have a limited range of values, the duration_ms has a much wider range, extending up to almost 300,000. We must scale the duration_ms afterwards to guarantee that our machine learning models work properly. The loudness must also be scaled because it contains negative numbers.

Table 2

Structure of the data

#	Column	Non-Null Count	Dtype
0	id	586672	object
1	name	586601	object
2	popularity	586672	int64
3	duration_ms	586672	int64
4	explicit	586672	int64
5	artists	586672	object
6	id_artists	586672	object
7	release_date	586672	object
8	danceability	586672	float64
9	energy	586672	float64
10	key	586672	int64
11	loudness	586672	float64
12	mode	586672	int64
13	speechiness	586672	float64
14	acousticness	586672	float64
15	instrumentalness	586672	float64
16	liveness	586672	float64
17	valence	586672	float64
18	tempo	586672	float64
19	time_signature	586672	int64
dtypes: float64(9), int64(6), object(5)			

Key Findings:

According to the data structure, there are six features with integer values, nine features with floating values, and five features with categorical data. In total, we have 15 numerical characteristics and 5 category attributes. The 'name' feature appears to be lacking some information as well. Because the "name" characteristic is different between songs and the number of missing values is small, we may gradually eliminate those missing values.

Table 3

Summary statistics

	count	mean	std	min	25%	50%	75%	max
popularity	586672	27.570053	18.370642	0	13	27	41	100
duration_ms	586672	230051.167	126526.087	3344	175093	214893	263867	5621218
explicit	586672	0.044086	0.205286	0	0	0	0	1
danceability	586672	0.563594	0.166103	0	0.453	0.577	0.686	0.991
energy	586672	0.542036	0.251923	0	0.343	0.549	0.748	1
key	586672	5.221603	3.519423	0	2	5	8	11
loudness	586672	-10.206067	5.089328	-60	-12.89	-9.243	-6.482	5.376
mode	586672	0.658797	0.474114	0	0	1	1	1
speechiness	586672	0.104864	0.179893	0	0.034	0.0443	0.0763	0.971
acousticness	586672	0.449863	0.348837	0	0.0969	0.422	0.785	0.996
instrumentalness	586672	0.113451	0.266868	0	0	2.4E-05	0.0096	1
liveness	586672	0.213935	0.184326	0	0.0983	0.139	0.278	1
valence	586672	0.552292	0.257671	0	0.346	0.564	0.769	1
tempo	586672	118.464857	29.764108	0	95.6	117.384	136.32	246.381
time_signature	586672	3.873382	0.473162	0	4	4	4	5

Key Findings:

The popularity column runs from 0 to 100, with an average of 27, showing that the majority of songs are not popular and that the data is skewed to the right. The smallest value in the explicit column is 0, the maximum value is 1, and the average value is 0.04, showing that the majority of the songs are not explicit and that the data is clearly skewed to the right. Speechiness, instrumentality, and liveness numbers are skewed to the left.

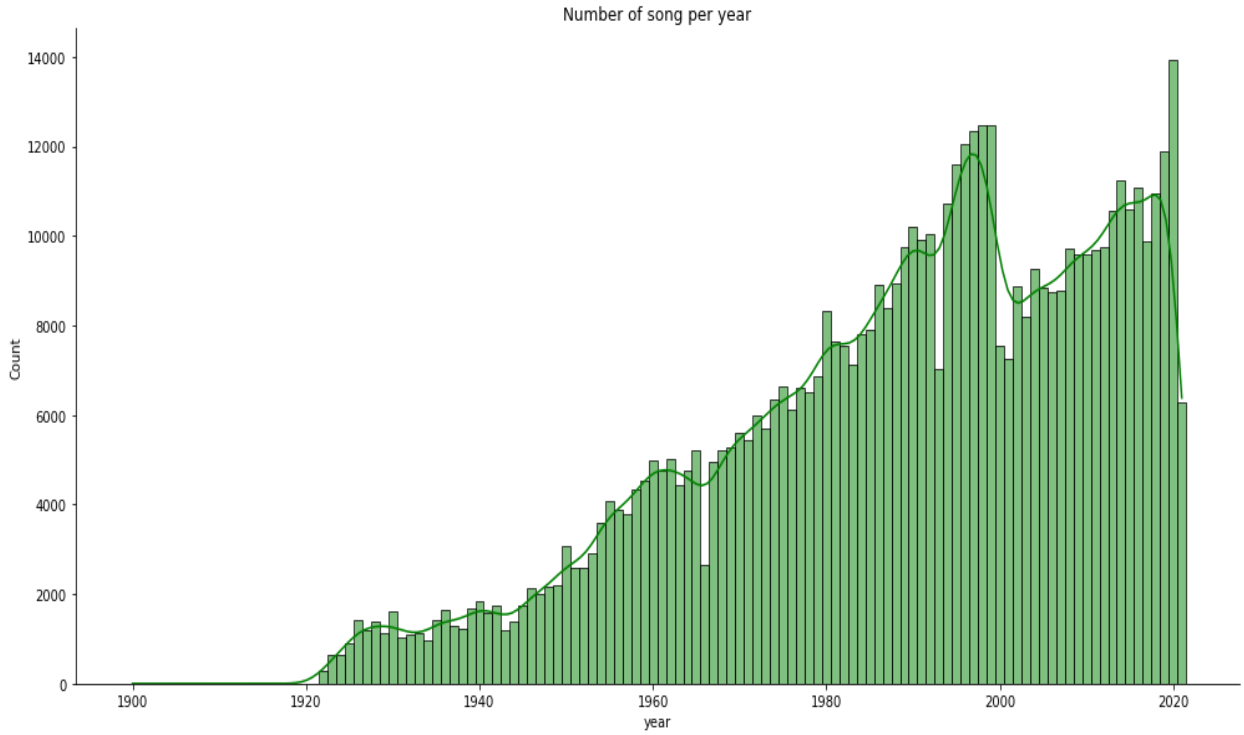
2. Data Visualization

A. What is the overall trend and impact of notable events on the music industry?

We use Pandas to calculate the number of tracks in the dataset by year based on the release_date attribute. The trend is then visualized using the Seaborn library.

Figure 1

Music trend by year



Key Findings:

This graph shows how the popularity of music has grown over time, indicating how the current generation is more inclined to listen to music and how digitization has benefited the music industry. We can also conclude that the terrorist attacks in 2001 had a substantial detrimental impact on the music industry because the graph shows a steep fall in 2021. The entertainment industry thrives during the covid epidemic, and people want to listen to music more, so the covid pandemic benefits the music industry.

B. Who are the most popular artists?

To find the most well-known artists, we first used Pandas' library to sort the number of tracks each artist had. Then we used two visualization techniques: a bar plot and a word cloud. These two visualization styles are great for delivering such imagery to audiences.

Figure 2

Bar Plot of the most popular artists

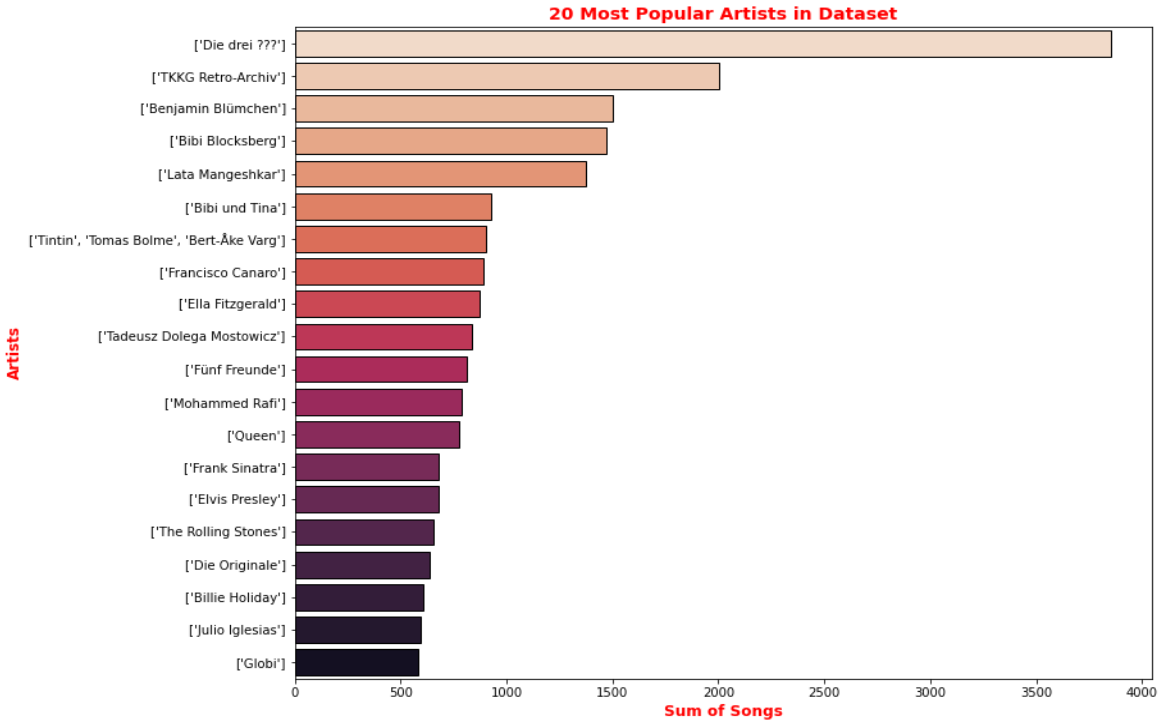


Figure 3

Word Cloud for the most popular artists

Key Findings:

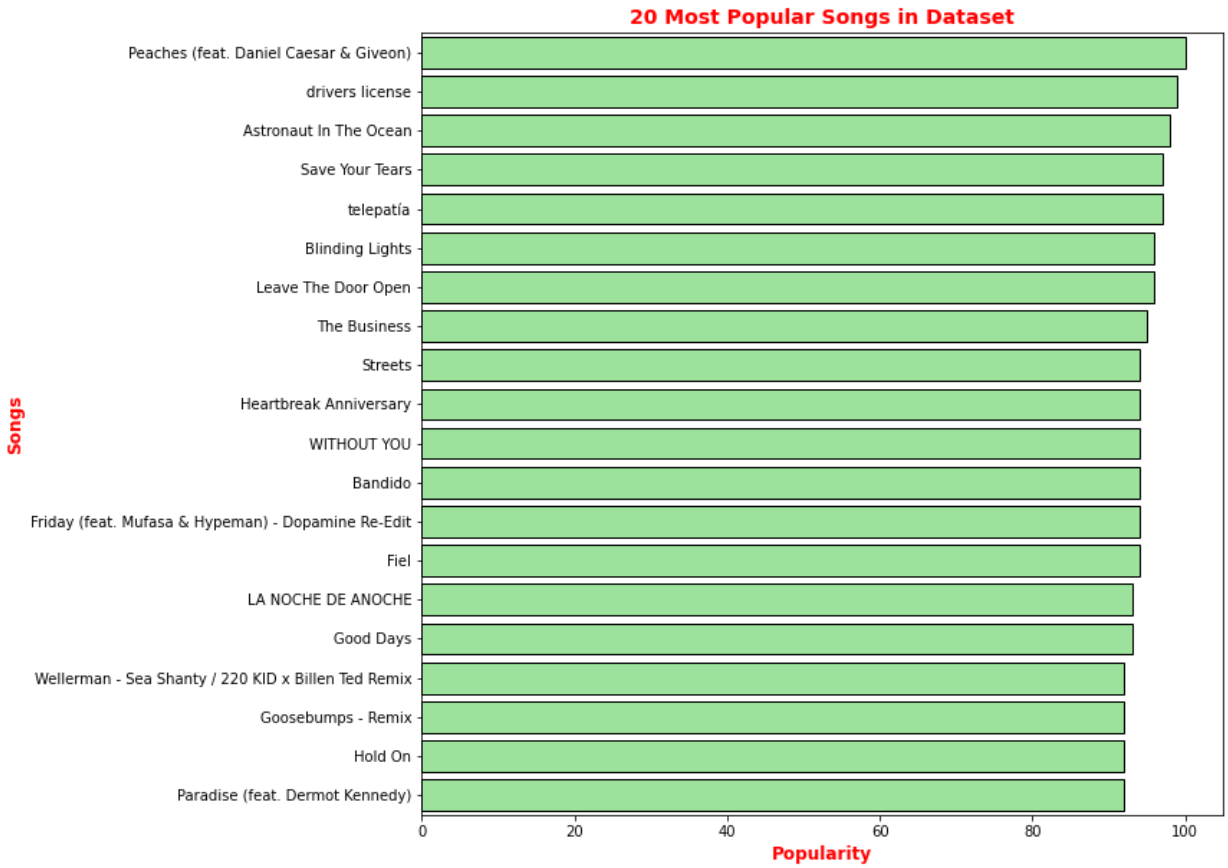
In this graphic, the size of the word represents how frequently it appears. This contains information about well-known musicians. The magnitude of the name "Francisco Canaro" in the upper left corner, for example, indicates popularity. The song "Die Drei???" has the largest size, indicating that it, along with "TKKG Retro-Archiv" and "Benjamin Blümchen," is the most frequently replayed. We are also familiar with well-known musicians such as Frank Sinatra and Elvis Presley.

C. Which are the most popular songs in the dataset?

We used Pandas to sort the most popular songs in the dataset based on their popularity. Then we used Matplotlib to visualize them.

Figure 4

The most popular songs in the dataset



Key Findings:

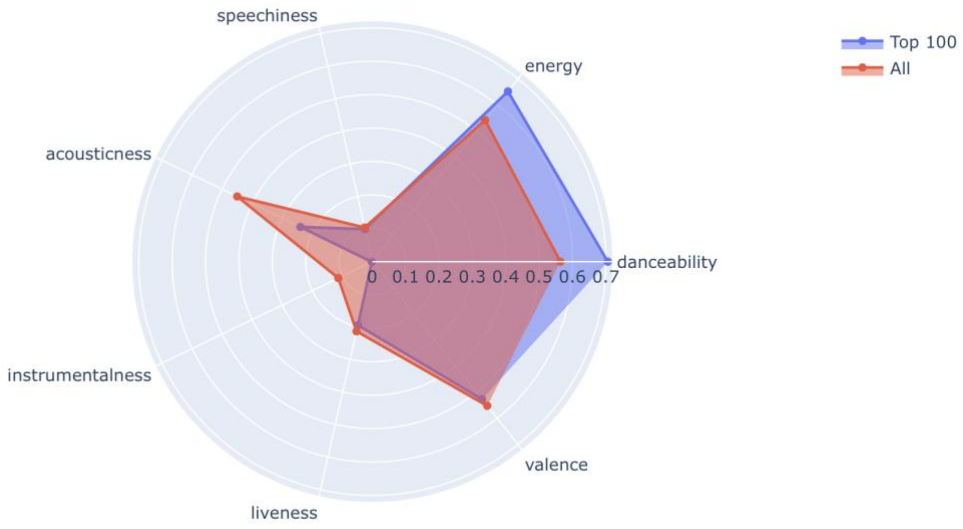
Based on the results, the most popular songs are “Peaches” of Justin Bieber (2021). This song received the MTV Video Music Award for Best Pop and was nominated for the Grammy Award for Song of the Year. Other songs, such as "Astronaut in the Oceans" and "Binding Lights," are well-known worldwide. Talented performers such as Masked Wolf and The Weekend perform them.

D. What characteristics most clearly characterize a popular song?

We used the plotly library to compare the general aspects of normal songs and popular songs based on the most popular songs we discovered above.

Figure 5

Features impacting the popularity most



Key Findings:

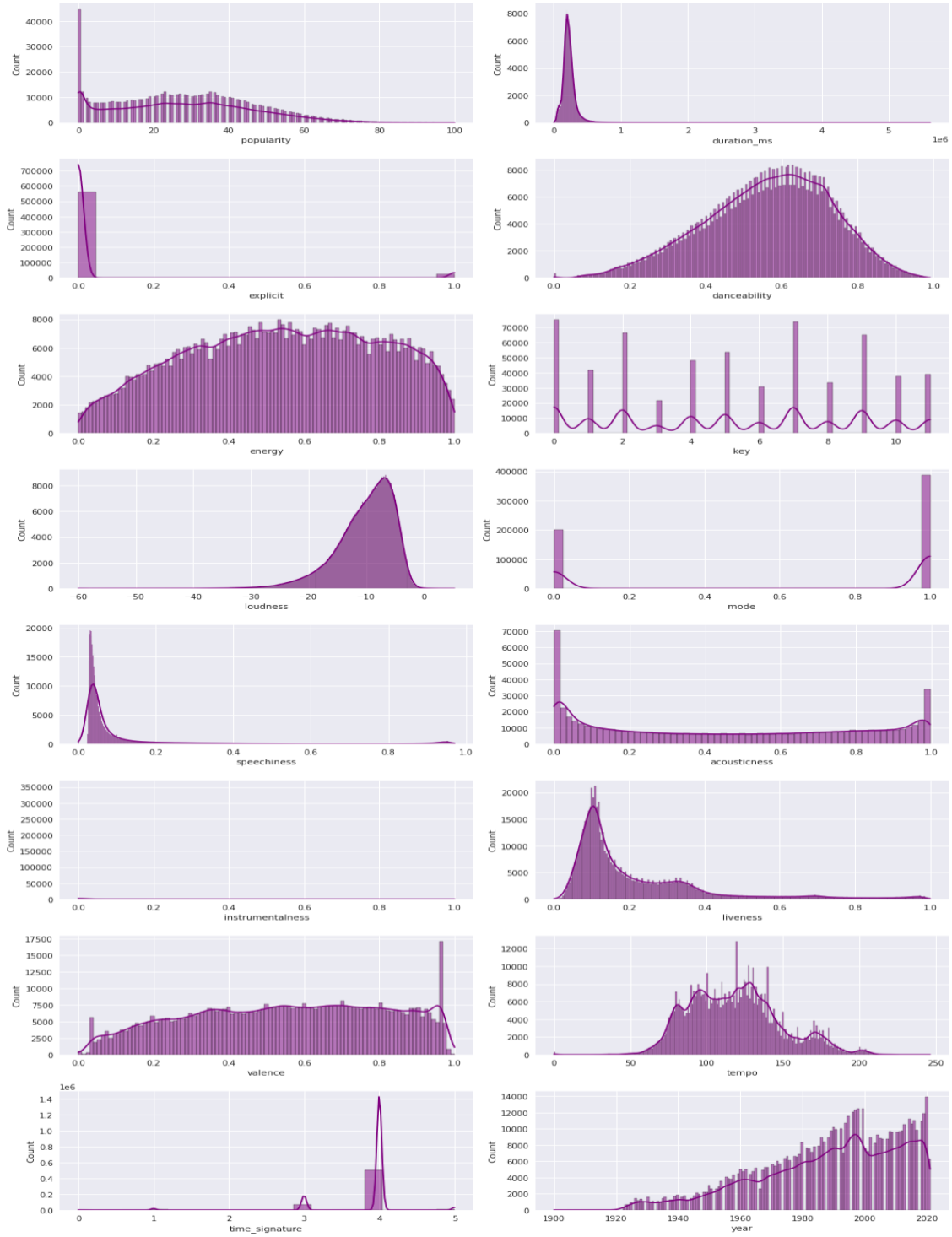
When we compared the most popular songs to all of the songs in the dataset, we discovered that songs with higher energy and danceability tend to become more popular. However, when compared to standard songs, such popular songs lack acousticness and instrumentalness. Tracks released in recent years tend to be more popular.

E. Univariate Analysis

Now we analyze each variable in the dataset and visualize them to see the distribution as well as the trends.

Figure 6

Histograms of variables



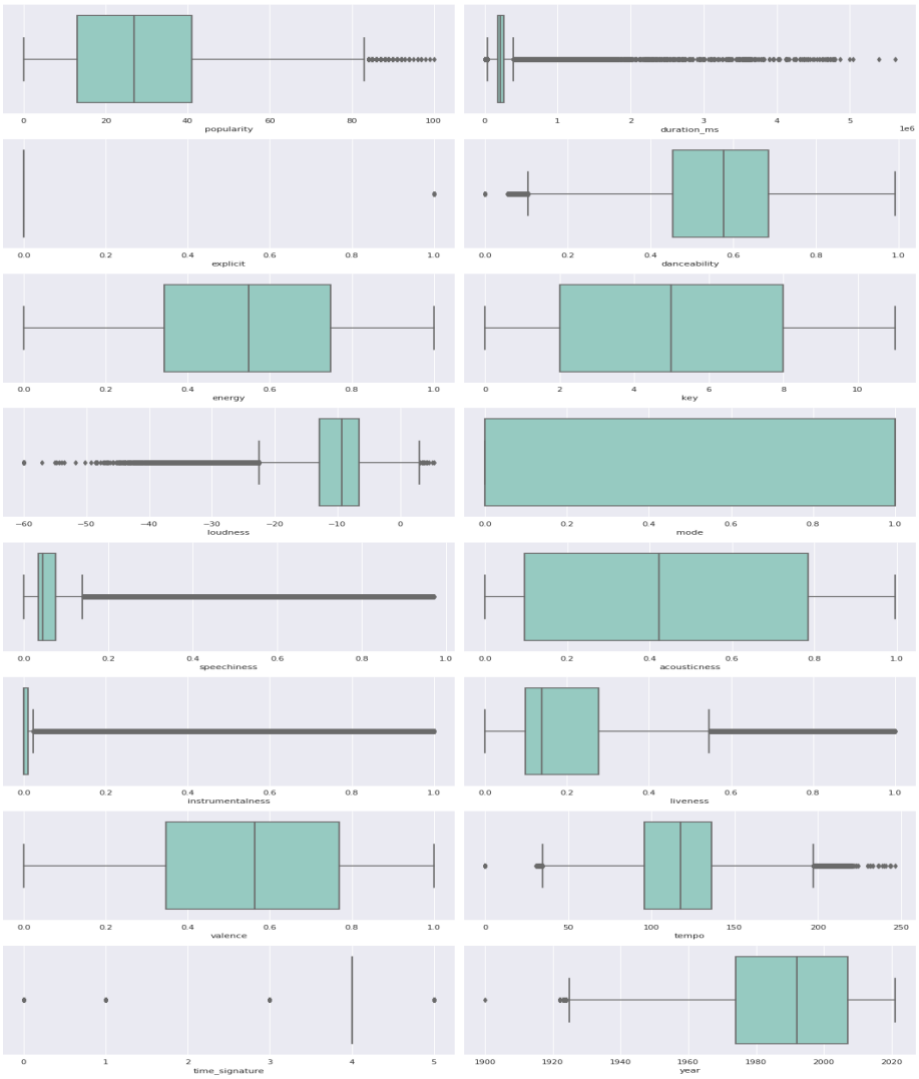
Key Findings:

We can see that popularity, duration, explicit, speechiness, instrumentalness, liveness is skewed right. loudness, mode, time_signature are skewed left. Only danceability, energy, valence, tempo have normal distribution. Before we can use the machine learning models, we must transform the left and right skewed features have more symmetrical and bell-shaped distributions.

F. Outliers Detection

Figure 7

Box Plot for detecting outliers.



Key Findings:

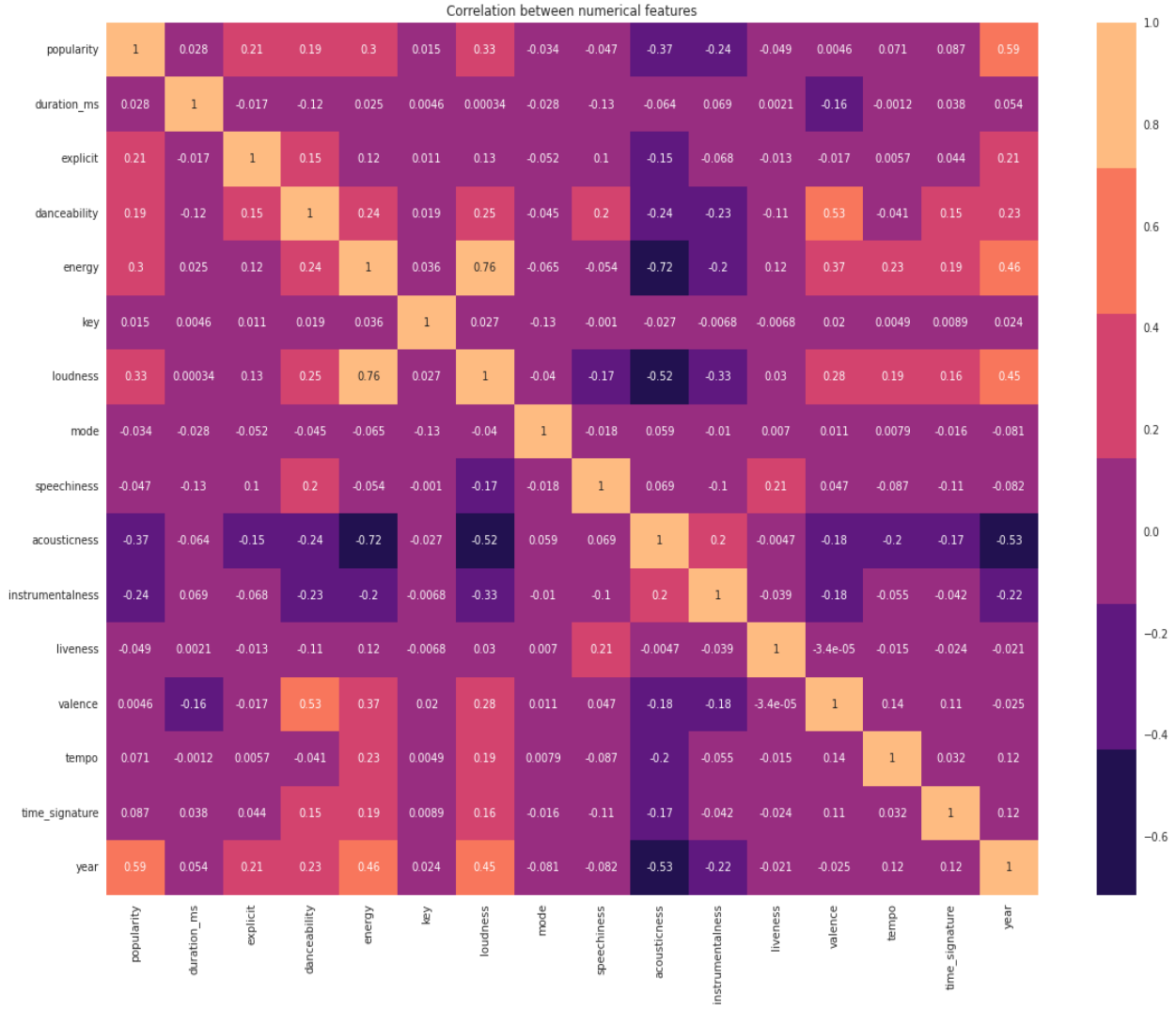
We can see that there are various features that have a lot of outliers from the boxplots of the variables. They are loudness, speechiness, instrumentality, liveness, duration ms, and pace. By utilizing IQR Ranges to reduce outliers, we can handle them. Or, we may handle them by using Robust Scaler. This scaler additionally employs IQR Ranges, making it extremely resistant to outliers.

G. Bivariate Analysis:

We compare the relationship among variables to see their correlation. We used the heatmap seaborn library to produce the correlation scores of variables. From that we can select useful features.

Figure 10

Correlation Matrix



Key Findings:

We may deduce the following facts from the correlation matrix. According to the correlation matrix, there is a strong positive link between energy and loudness, with a score of 0.76. With a score of -0.72, energy and acousticness have a strong negative association. We can utilize the remaining variables to train the machine learning models because they have a good connection.

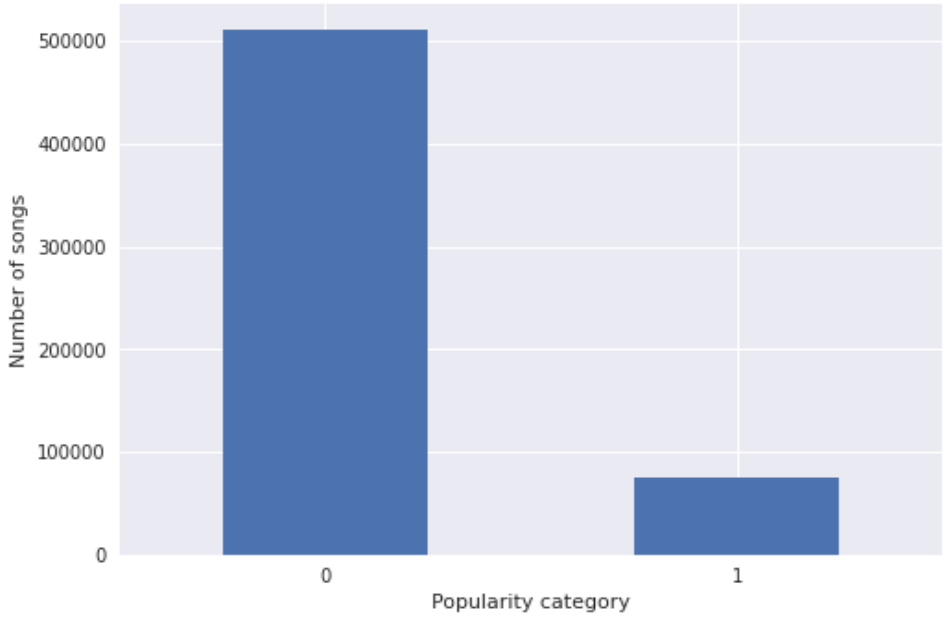
3. Feature Selection

Based on the column 'popularity,' we establish a new variable named 'highly_popular' with a threshold of 50. Songs with a popularity rating of more than 50 are highly popular. This will be our target variable for predicting a song's popularity.

After creating this variable, we drop the column ‘popularity’ and check the number of unique values the column ‘highly_popular’ has.

Figure 11

Number of observations per class



Key Findings:

We have 510,758 unique values of class 0 and 75,843 unique values of class 1. We can see that the number of class 0 is 5 times more than the number of class 1. This is unbalanced data. To balance

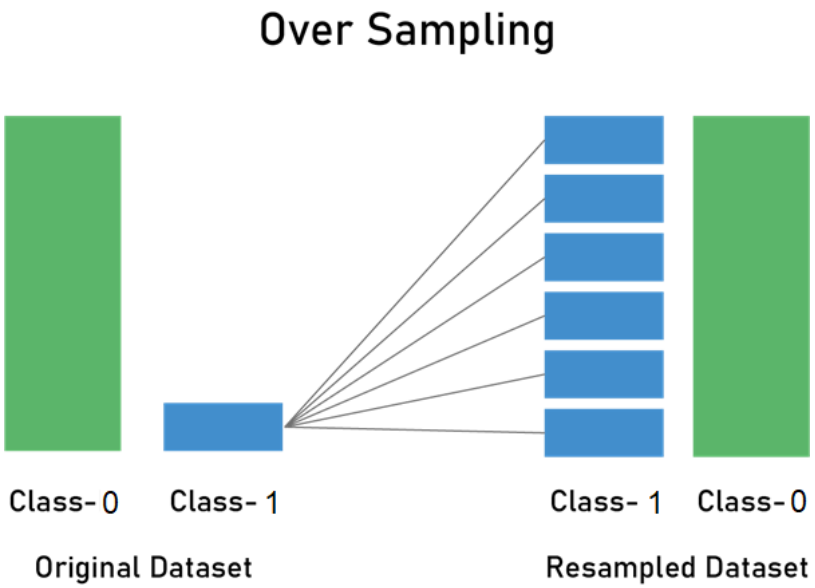
them, we must employ the Resampling approach. But before that, we need to split the target variable from predictors.

Resampling

We first split the target variable ('highly_popular') into X and y matrices. Then the RandomOverSampler module is used to select observations from the dataset at random as long as they have the same number of classes (Peixeiro, 2020). Therefore, we have 75,843 observations of class 0 and 75,843 observations of class 1. They are transformed into balanced data.

Figure 12

Oversampling technique



Next, we split the training set and the test set with the ratio of 8:2.

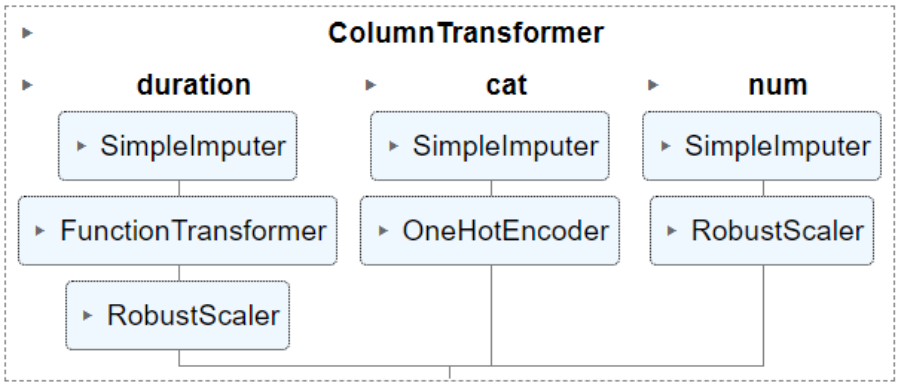
4. Pipeline Building

Pipeline is an effective way to change and manipulate a lot of data. Pipelines are a series of ways to process data. With Pandas pipeline feature, we can string together different user-defined Python functions to make a pipeline for processing data.

Here we have created 3 small pipelines. One for 'duration', one for categorical variables and one for numerical variables.

Figure 13

Preprocessing Pipeline



The first pipeline transforms the 'duration' feature. Checking and replacing the missing values of this characteristic with the median value is the initial step. We use SimpleImputer for this purpose. Then, since this characteristic is initially recorded in milliseconds in the dataset. As a result, it has a vast range of possible values, up to 5 million. Therefore, we attempt to translate this characteristic to 'minute' by dividing each value by (1000 * 60). Finally, we scale this feature using sklearn's RobustScaler(). This scaler is extremely tolerant of outliers.

The second pipeline consists of converting categorical variables to numerical variables using SimpleImputer for missing values and OneHotEncoder. The most frequent value is used to replace any missing data. OneHotEncoder is then utilized to encode them into spare matrices. However, because there are no categorical columns in the dataset that can be used with this pipeline, we are unable to use it at this time.

The third pipeline is responsible for transforming numeric variables. First, missing data is replaced with the median value. Then, we utilize the RobustScaler() function to scale these attributes. Here, we employ all numerical characteristics because they are song qualities and, after analysis, they are suitable for this categorization problem.

5. Model Building

First, we create an evaluation function to analyze the performance of each model. The findings are obtained by running the training and testing data through the Machine Learning models.

Logistic Regression

Logistic Regressions are the original machine learning algorithm, and they are commonly employed as binary classifiers. This algorithm functions similarly to linear regression, except it produces results based on a sigmoid function with values ranging from 0 to 1.

First, we construct a basic Logistic Regression model and assess it using several metrics such as accuracy, precision, recall, F1-score, and ROC AUC score. Then we try different combinations of hyperparameters to fine-tune it.

Figure 14

Logistic Regression Pipeline

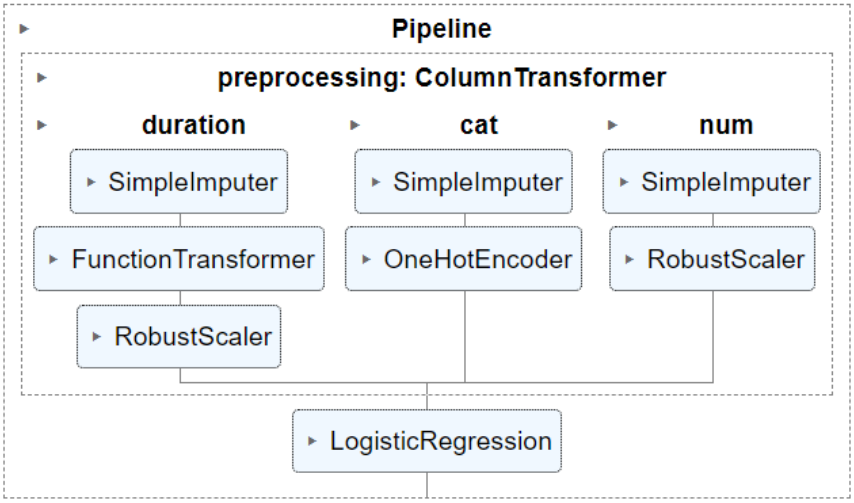


Figure 15

Results of the Logistic Regression model

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.7019	0.6678	0.6844	101950
1	0.6845	0.7175	0.7006	102382
accuracy			0.6927	204332
macro avg	0.6932	0.6927	0.6925	204332
weighted avg	0.6931	0.6927	0.6925	204332

ROC AUC Score: 0.6926730354079126

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.7012	0.6687	0.6846	101950
1	0.6847	0.7163	0.7001	102382
accuracy			0.6926	204332
macro avg	0.6930	0.6925	0.6924	204332
weighted avg	0.6929	0.6926	0.6924	204332

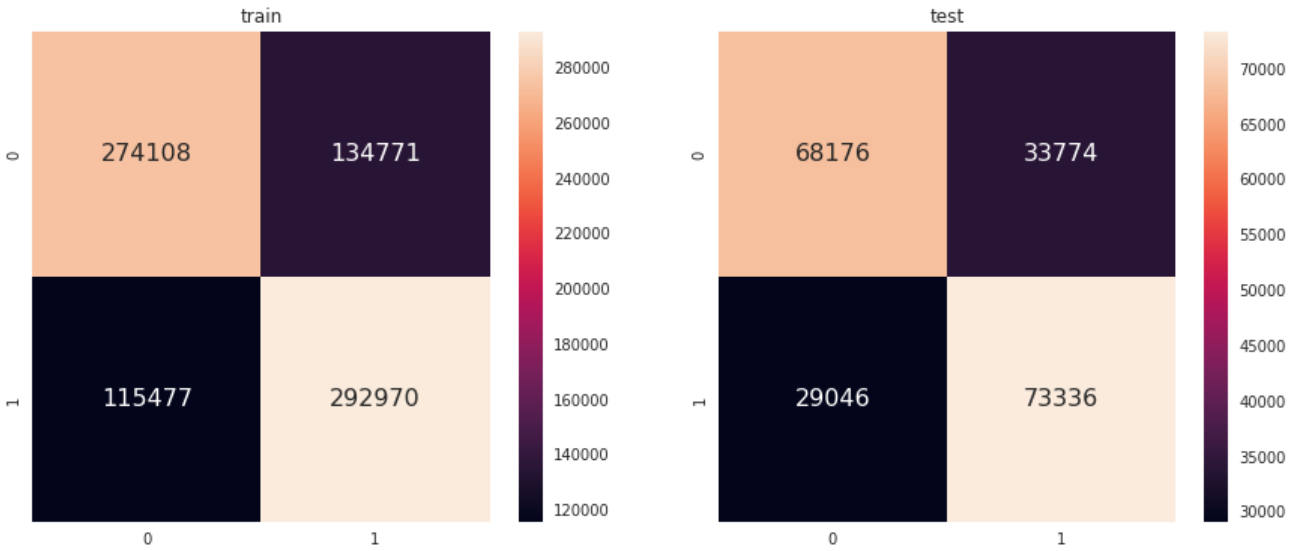
ROC AUC Score: 0.6925088737427014

a) Logistic Regression model *b) Hyperparameter-Tuned Logistic Regression model*

The basic logistic regression model gives fairly reliable results on the test set (accuracy = 0.6927). We utilize Sklearn's GridsearchCV module to try and find the best parameter for our model. We determined that the optimum penalty function is l2, which aids in avoiding overfitting (regulization), and that the C parameter, which regulates the penality strength, is c=100. 'lbfgs' is the best solver we get. So, after feeding this parameter into the logistic model and running the model again, it still produced the same accuracy (accuracy = 0.6926), implying that this is the best achievable accuracy for this dataset with the logistic model.

Figure 16

Confusion matrices of the Logistic Regression model



We will do some calculations here to analyze the confusion matrix's results: False Positive (FP): 33774, True Positive (TP): 73336, True Negative (TN): 68176, False Negative (FN): 29046.

Accuracy is a metric used to determine how frequently a model is accurate. Accuracy = (TP + TN)/total = (73336+68176)/204332 = 0.6925

Misclassification Rate: The rate at which the model makes mistakes is known as the misclassification rate. Accuracy and Misclassification Rates are in direct opposition to one another. Accuracy minus one equals Misclassification. Error Rate is another name for the misclassification rate. Misclassification Rate = (FP + FN)/Total = (33774 + 29046)/204332 = 0.3074

True Positive Rate/Recall/Sensitivity: How frequently does the model correctly anticipate a yes (popular) outcome? 73336/(73336+29046) = TP/(TP+FN) = 0.7163

False Positive Rate: How frequently does the model correctly predict yes (popular) when the true answer is no (not-popular)? 33774/(33774+68176) / (FP+TN) = 0.3312

True Negative Rate/Specificity: How frequently does the model correctly forecast that a value of no (not-popular) actually occurs?

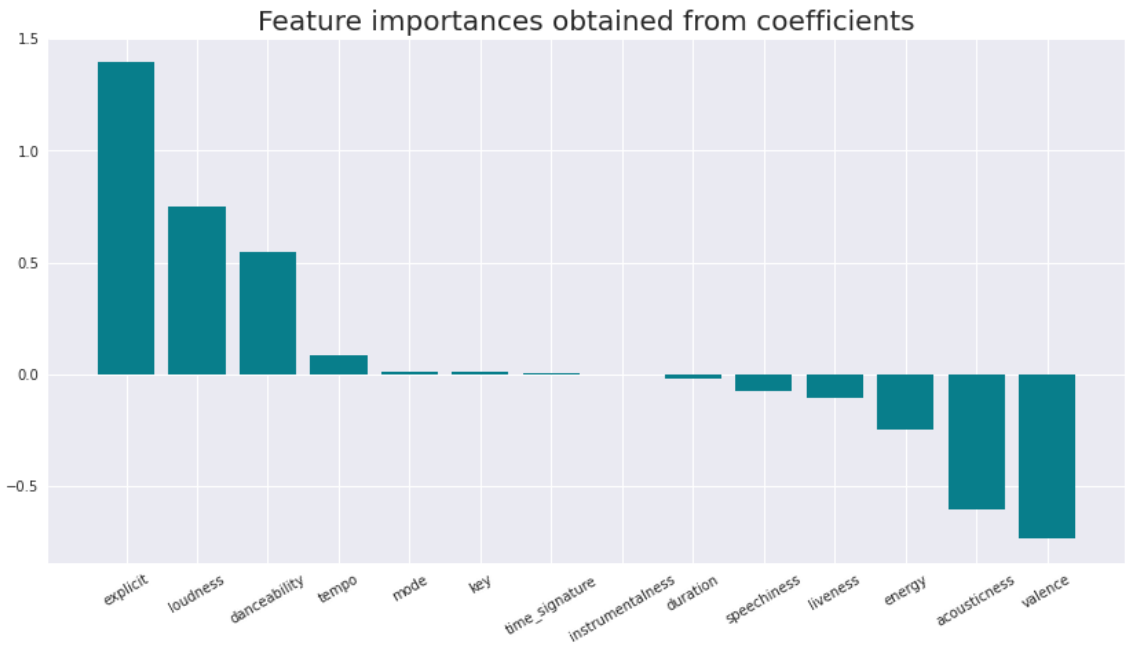
False Positive Rate: divided by True Negative Rate equals one. The formula is TN/(TN+FP) = 68176/(68176+33774) = 0.6687

Precision: How frequently is the model's yes prediction accurate? TP/(TP+FP) = 73336/(73336+33774) = 0.6846.

So all the results here are similar to what we have in the above evaluation.

Figure 17

Feature Importance of Logistic Regression



The **explicit, loudness, and danceability** of the songs contribute most to their popularity. At the same time, **valence, acousticness, and enery** are qualities that make the music less popular. So, in order for a song to be successful, it must have a good combination of these traits. This outcome further validates our findings, as illustrated in Figure 5.

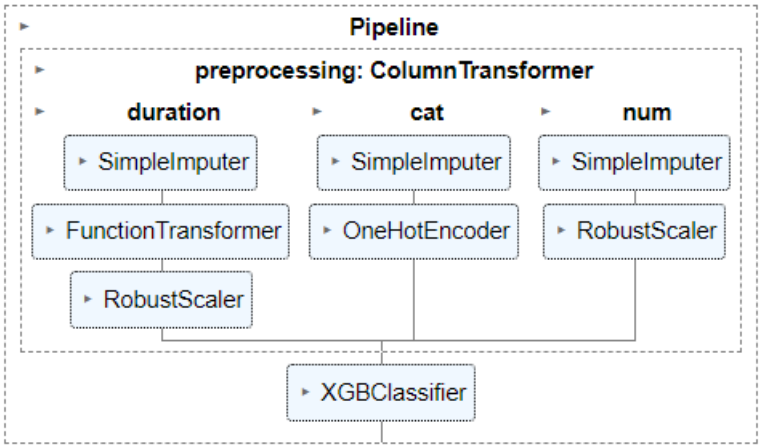
XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning framework. It is the top machine learning package for regression, classification, and ranking tasks, and it supports parallel tree boosting (Harode, 2020).

We begin by creating a basic XGBoost model and training it on the dataset. Then we assess it on the test set using various metrics. The model is then fine-tuned using RandomizedSearchCV. This function allows us to attempt various hyperparameter combinations at random. It saved us a lot of time and was also compatible with the capacity of our laptop.

Figure 18

XGBoost model pipeline



Hyperparameter Tunning

Figure 19

Results of the XGBoost model

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.7519	0.7079	0.7293	101950
1	0.7252	0.7674	0.7457	102382
accuracy			0.7377	204332
macro avg	0.7385	0.7377	0.7375	204332
weighted avg	0.7385	0.7377	0.7375	204332

ROC AUC Score: 0.7376630073320979

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.7352	0.6988	0.7165	101950
1	0.7142	0.7494	0.7314	102382
accuracy			0.7241	204332
macro avg	0.7247	0.7241	0.7240	204332
weighted avg	0.7247	0.7241	0.7240	204332

ROC AUC Score: 0.7240964380491992

b) XGBoost model b) Hyperparameter-Tuned XGBoost model

The basic XGBoost model gives satisfactory results on the test set (accuracy = 0.7377). To select the optimum parameter for our model, we use Sklearn's RandomizedSearchCV module. We discovered that the best parameters are as follows: n_estimators = 100, gamma = 0.5, max_depth = 6, learning_rate = 0.1, min_child_weight = 1, subsample = 1, colsample_bytree = 1, objective = 'binary:logistic', random_state = 42. However, the XGBoost model did not change its output much

after being fed this parameter and rerun (accuracy = 0.7241). We think that if we had a more robust computer, we could test more variants of the hyperparameters and hence get better results.

Figure 20

Confusion matrices of the XGBoost model

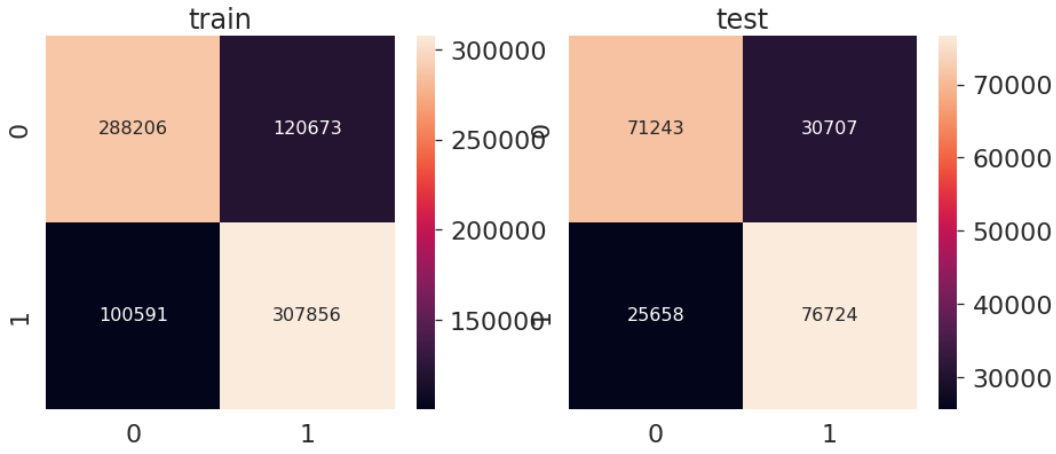
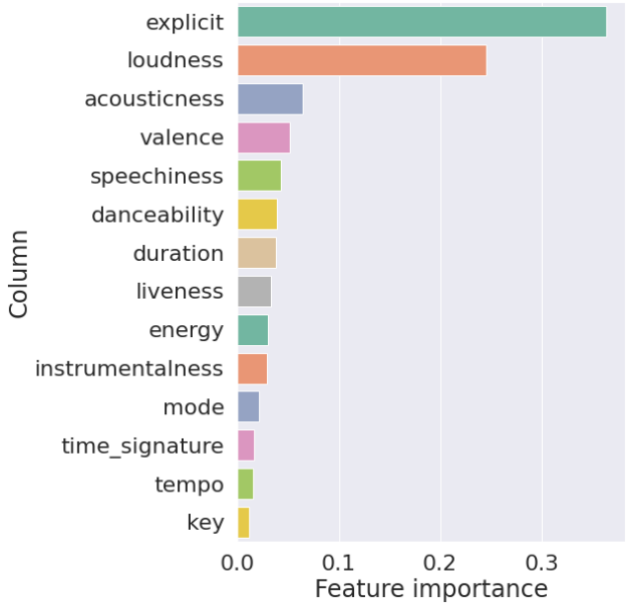


Figure 21

Feature Importance of the XGBoost model



According to the XGBoost model, the most critical factors that determine whether a song is popular or not are **explicit, loudness, acousticness, and valence**. This outcome is remarkably similar to our previous findings.

Random Forest

The random forest algorithm is a classification system made up of numerous decision trees. When building each individual tree, it employs bagging and feature randomness in an attempt to produce an uncorrelated forest of trees whose forecast by committee is more accurate than that of any individual tree (Education, 2020).

First, we build and train a simple Random Forest model using the data. Then, we use several metrics to evaluate it on the test set. RandomizedSearchCV is then used to fine-tune the model. Using this method, we can randomly explore a wide range of hyperparameter settings. It helped us save a lot of time and worked well with our laptop's specs.

Figure 22

Random Forest pipeline

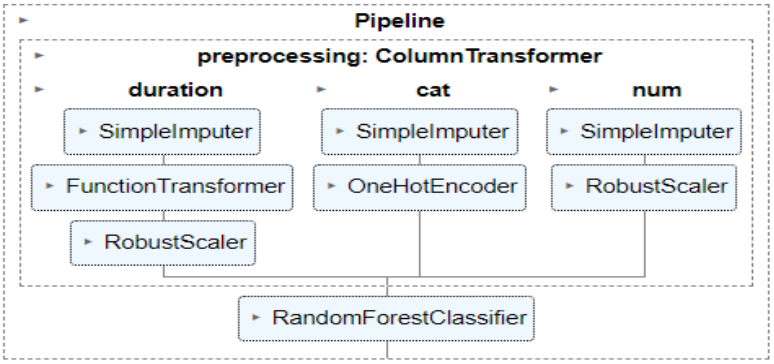


Figure 23

Results of the Random Forest model

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.9972	0.9553	0.9758	101950
1	0.9573	0.9973	0.9769	102382
accuracy			0.9764	204332
macro avg	0.9772	0.9763	0.9763	204332
weighted avg	0.9772	0.9764	0.9763	204332

ROC AUC Score: 0.9763077376250341

TEST RESULTS:

Classification Report:

	precision	recall	f1-score	support
0	0.9908	0.9359	0.9626	101950
1	0.9395	0.9914	0.9648	102382
accuracy			0.9637	204332
macro avg	0.9652	0.9636	0.9637	204332
weighted avg	0.9651	0.9637	0.9637	204332

ROC AUC Score: 0.9636425776934088

c) Random Forest model b) Hyperparameter-Tuned Random Forest model

The basic Random Forest model gives extremely superior results on the test set (accuracy = 0.9764). To select the optimum parameter for our model, we use Sklearn's RandomizedSearchCV module. We discovered that the best parameters are as follows: n_estimators = 300, max_features = 'sqrt', max_depth = 50, min_samples_leaf = 3, min_samples_split = 2, criterion = 'entropy', bootstrap = True, random_state = 42. With these new hyperparameters, the Random Forest model was run again, and it yielded similar findings (accuracy = 0.9637). This is the best results of the 3 models that we tried.

Figure 24

Confusion matrices of the Random Forest model

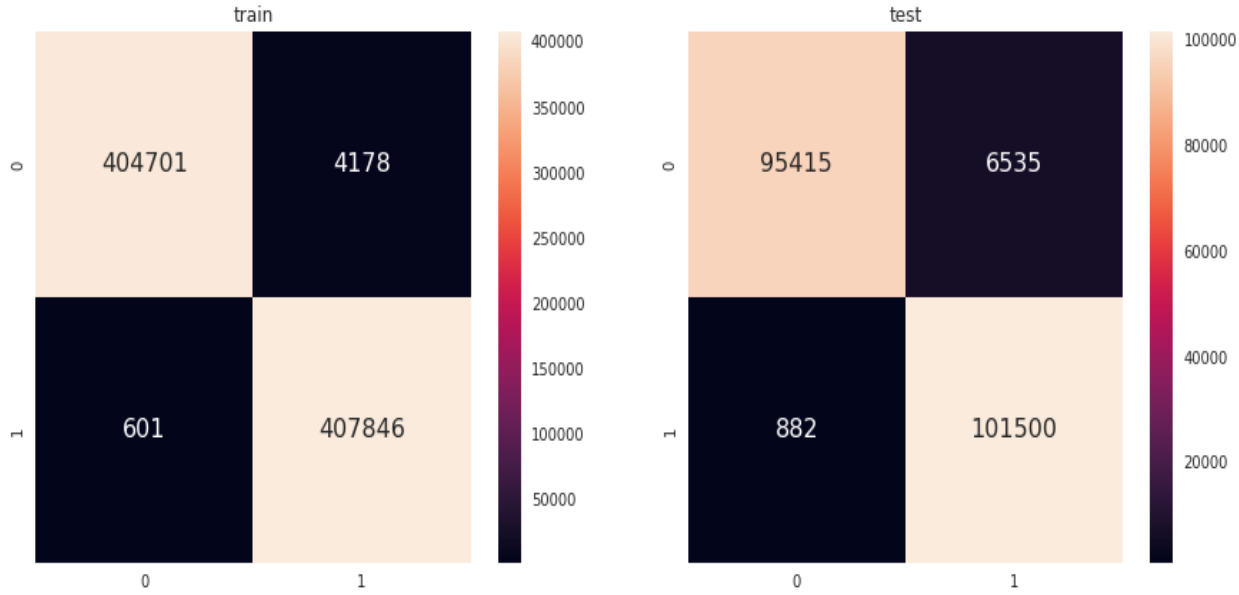
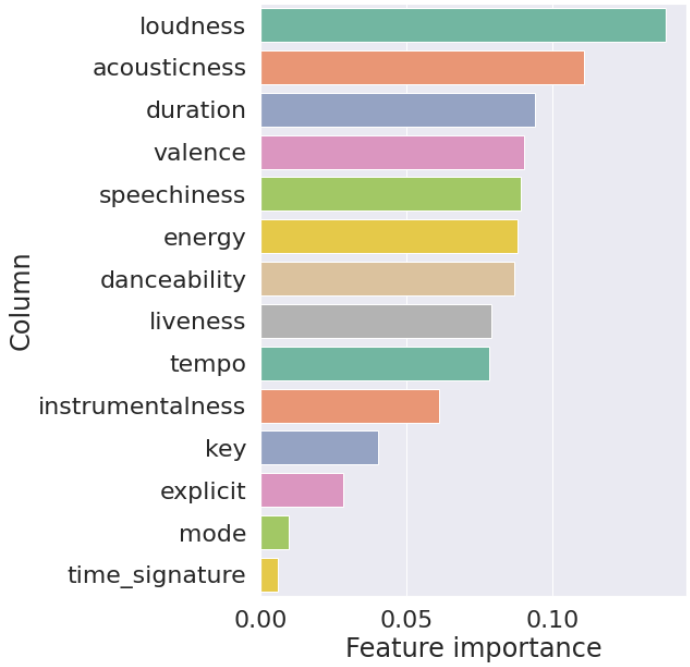


Figure 25

Feature Importance of the Random Forest model



Based on the results of the Random Forest model, we can infer whether a song is popular or not based on its **loudness, acousticness, duration and valence**. These are the characteristics that we discovered in our earlier modeling and visualizing efforts.

Conclusion

As part of this project, we completed all necessary data science stages and were able to get some fascinating insights from the data. First, we performed the visualization and drew several intriguing insights from the data, such as the overall trend of the data, the events that had the greatest impact on the music business, the most well-known musicians, and the most well-liked songs. Three pipelines have been put in place to transform features in our models. Transforming duration from a millisecond to a minute is the first pipeline. The second pipeline uses a single hot encoder to convert category data to numerical variables, while the third pipeline scales the features of numerical variables. For this scenario, we employed 3 models (Logistic Regression, Random Forest and XGBoost) and Random Forest performed best with an accuracy of almost 97%. We discovered that the most crucial features to consider are loudness, energy, valence, acousticness, and explicitness. These features have the greatest influence on popularity. After hyperparameter tuning, we didn't see any appreciable improvements in the model's performance, which may be attributed to the large dataset's size, the lengthy training times associated with each combination, and the laptop's technical limitations. Working with this dataset was really intriguing, and in the future, we'd like to conduct our study utilizing cloud computing on a more powerful system. We'd also like to examine some of Spotify's rivals, such as Apple Music, Wynk Music, YouTube Music, etc. We also want to use Artificial Neural Networks, SVM, and K-nearest Neighbor for our analysis. For our study, we'd also like to make use of aspects like the music's genre, such as pop, rock, jazz, rap, etc.

References

1.) Education, I. C. (2020, December 8). *What is Random Forest?* What Is Random Forest? | IBM. Retrieved October 25, 2022, from https://www.ibm.com/cloud/learn/random-forest

2.) *Spotify — About Spotify*. (2022, October 25). Spotify. Retrieved October 25, 2022, from https://newsroom.spotify.com/company-info/

3.) Peixeiro, M. (2020, May 19). *The Complete Guide to Resampling Methods and Regularization in Python | by Marco Peixeiro | Towards Data Science*. Medium. Retrieved October 25, 2022, from https://towardsdatascience.com/the-complete-guide-to-resampling-methods-and-regularization-in-python-5037f4f8ae23

4.) Harode, R. (2020, April 23). *XGBoost: A Deep Dive into Boosting | by Rohan Harode | SFU Professional Computer Science | Medium*. Medium. Retrieved October 25, 2022, from https://medium.com/sfu-cspmp/xgboost-a-deep-dive-into-boosting-f06c9c41349

5.) *Spotify Datasets*. (n.d.). Spotify Datasets | Kaggle. Retrieved October 25, 2022, from https://www.kaggle.com/datasets/lehaknarnauli/spotify-datasets