CS 6375.002

Machine Learning

Final Project

Predicting Song Popularity

Author:
Lakshmipriya Narayanan

December 10, 2024

Introduction and Potential applications

This project aims to predict how popular a song is using machine learning techniques and algorithms based on a dataset of songs and their features. We utilize a large comprehensive dataset containing information on various song attributes, including song characteristics (energy, danceability, loudness, etc.) and metadata (artist name, year of release, genre, etc.).

Our goal is to develop a machine learning model that can accurately predict a future song's popularity based on the above mentioned features and hence help an artist predict if their song is going to be popular or not and can improve their song by tweaking some of the song attributes to make it a hit.

By gaining insight into what characteristics make a song popular, artists and music producers may be able to improve their songs or data scientists at Spotify may make informed decisions in generating "Spotify Wrapped" for consumers of the app.

Data Description

The chosen dataset for this project was a Spotify Songs Dataset from Kaggle (). It includes 32,828 songs and 23 different song attributes between the years 1957 - 2020. Below you will find description of the attributes present in this dataset (taken from Kaggle Spotify dataset of 30,000 songs)

- track id (Char): A unique ID for each song
- track name (Char): Name of song
- track artist (Char): Artist of song
- track popularity: Popularity of a song on a scale of $0 - 100$. 0 is the least popular and 100 is the most popular.
- track album id (Char): Unique ID of album
- track album name (Char): Name of song album
- track album release date (Char): Date of album release
- playlist name (Char): Name of playlist
- playlist id (Char): Unique ID of playlist
- playlist genre (Char): Genre of playlist
- playlist subgenre (Char): Subgenre of playlist
- Danceability: Represents how appropriate a song is for dancing, determined by various musical factors such as tempo, rhythm consistency, beat intensity, and overall regularity on a scale of $0 - 1$ where 0 refers to least danceable.
- Energy: Represents a measure of song intensity and activity. For example, death metal has high energy whereas a song with low beats may have low energy. This is also measured on a scale of $0 - 1$.
- Key: Estimation of overall key of track. -1 for no key

- Loudness: Overall loudness of a song in dB (deciBels is a measure of sound/noise). Values range from $-60\text{dB} - 0\text{dB}$
- Mode: If track is major then $mode = 1$. if track is minor then $mode = 0$
- Speechiness: Refers to the presence of spoken words in a track. An audiobook or a podcast typically have high speech in them and hence have a value of ≈ 1. Values range from $0 - 1$.
$$\begin{cases} x > 0.66, & \text{spoken words only} \\ 0.33 \leq x \leq 0.66, & \text{music and speech} \\ x < 0.33, & \text{music} \end{cases}$$
- Acousticness: Does the track solely use instruments to produce sounds. Values range from $0 - 1$.
- Instrumentalness: Represents whether a track contains no vocals.
- Liveness: Detects the presence of audience in the recording. Higher values of liveness imply that the track was performed live for an audience. Values range from $0 - 1$
- Valence: Represents musical positiveness in a track. Happy and cheerful tracks sound more positive and have high valence. Whereas, tracks with low valence sound gloomy, angry or, depressed.
- Tempo: Overall estimated tempo of a track measure in BPM (Beats Per Minute).
- Duration ms: Duration of a track measure in milliseconds.

For this project we only make use of quantitative variables to predict song popularity. We drop the rest. We also drop numerical variable 'duration_ms' because the duration of a song does not help in predicting its popularity. Variable 'mode' is also dropped because it is a binary variable.

Therefore, in this project we will consider
Label = track_popularity
Features = {danceability, energy, key, loudness, speechiness, acousticness, instrumentalness, liveliness, valence, tempo}

Data Cleaning

Firstly, finding an appropriate dataset with more than $10,000$ observations was extremely challenging to find. Most datasets were outdated and did not have any desirable attributes to work with.

This dataset was relatively organized and hence the data pre-processing was not very tedious. After taking a look at the variables present in this dataset, we decided to drop irrelevant predictors for regression like track id, track album id, playlist name, playlist id, playlist subgenre, duration ms and mode. These variables are not quantitative (except duration) and just by inspecting it do not help with predicting a hit song. So we choose the numerical predictors and use the forward stepwise selection method to perform variable selection.

Once a method was decided, we proceeded to execute them. First, we examine null values and get rid of them for accurate analysis. There were 5 observations that had missing values and they were from track name, track artist and track album name. If we don't have information on a song then making predictions on it is meaningless. So, they were dropped and the resulting dataset had $32,828$ observations.

Summary		
Variable	Unique	Duplicate
Track name	23,449	9,379
Track artist	10,692	22,136
Track popularity	101	32,727
Track album name	19,743	13,085
Playlist genre	6	32,822
Mode	2	32,826

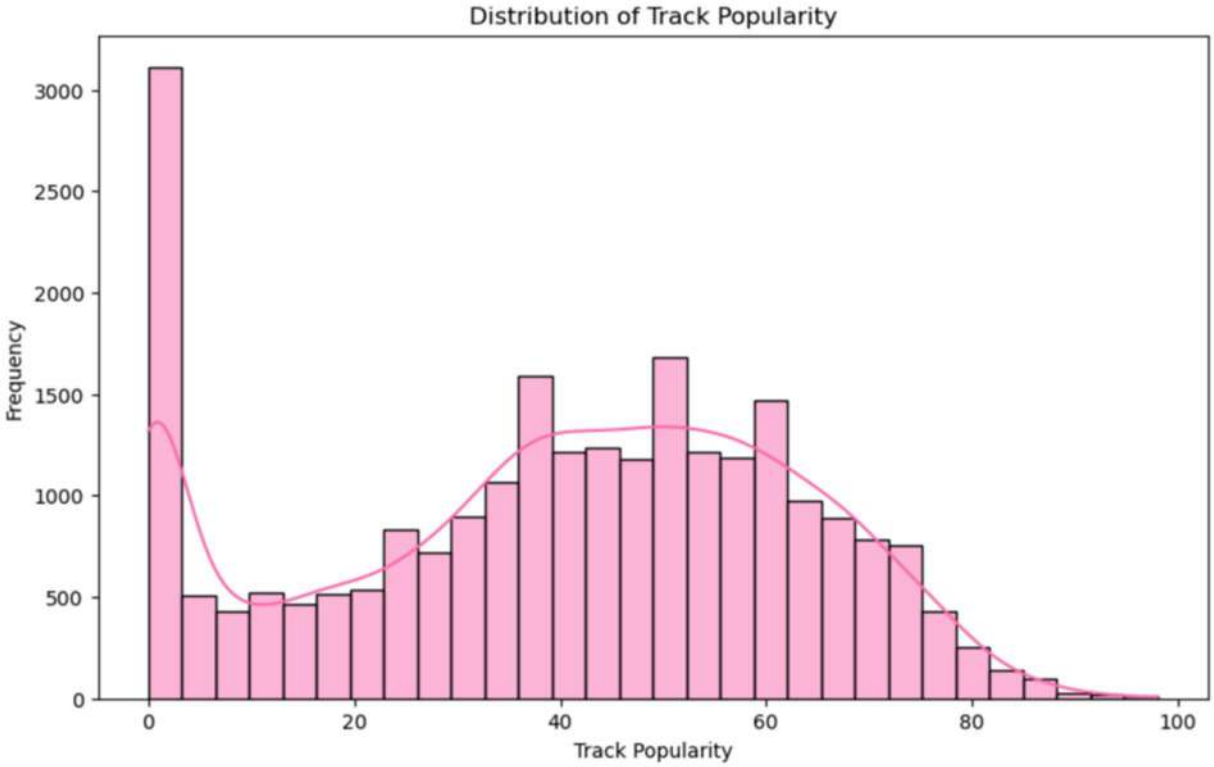
Next, we examined unique values and duplicates. This was done to ensure that features had the values of the range specified in the description. Track names and artist names were lesser than the amount of observations and this is because columns like track name, track artist, track album name and track album release data can have contribution from many different artists over the years. On the other hand, variables like genre and mode have fewer unique values indicating that these are categorical features with a limited set of possible values.

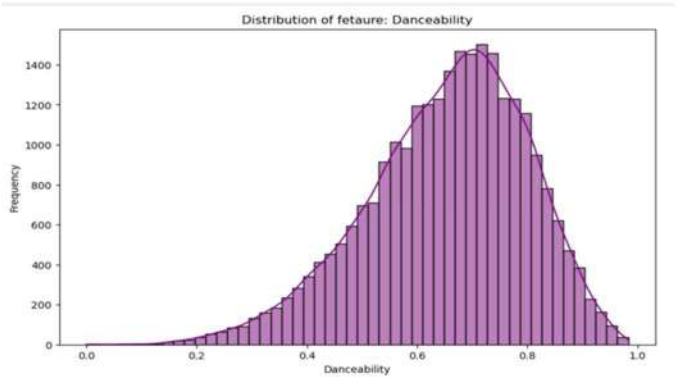
High duplicate counts like track name or track artist are valid because there is a high possibility that there are many songs with the same name, or artists with multiple songs and albums with multiple tracks. Similarly, high duplicate counts of track popularity and playlist genre, etc., indicate that many different songs share the same or similar values for these features.

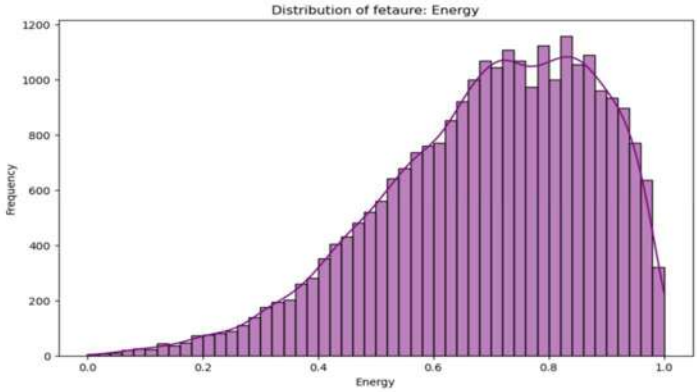
Other than this, any other data cleaning was very minor.

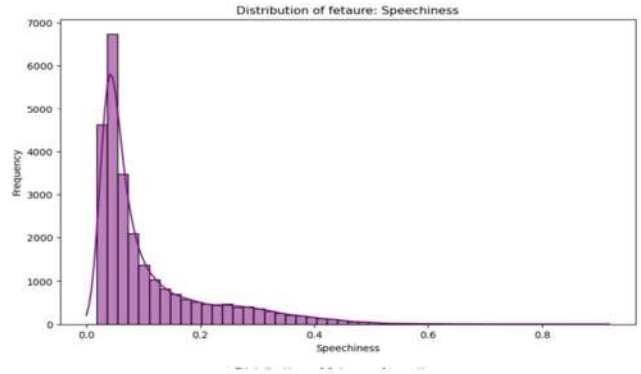
Exploratory Data Analysis

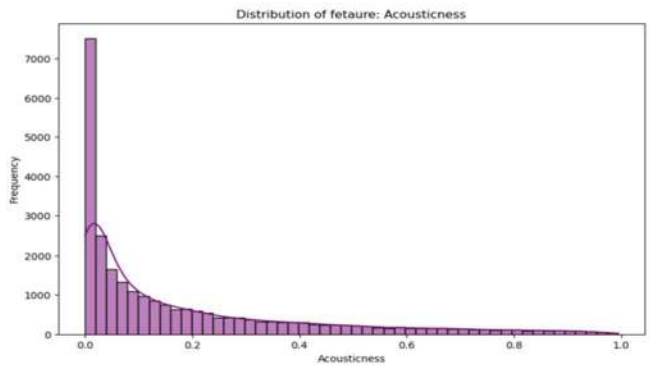
Before doing any sort of analysis, we first wanted to see the distribution of all the features we will be considering and so we proceeded to plot histograms with a density curve to see if any of these features (and response) follow normal distribution.





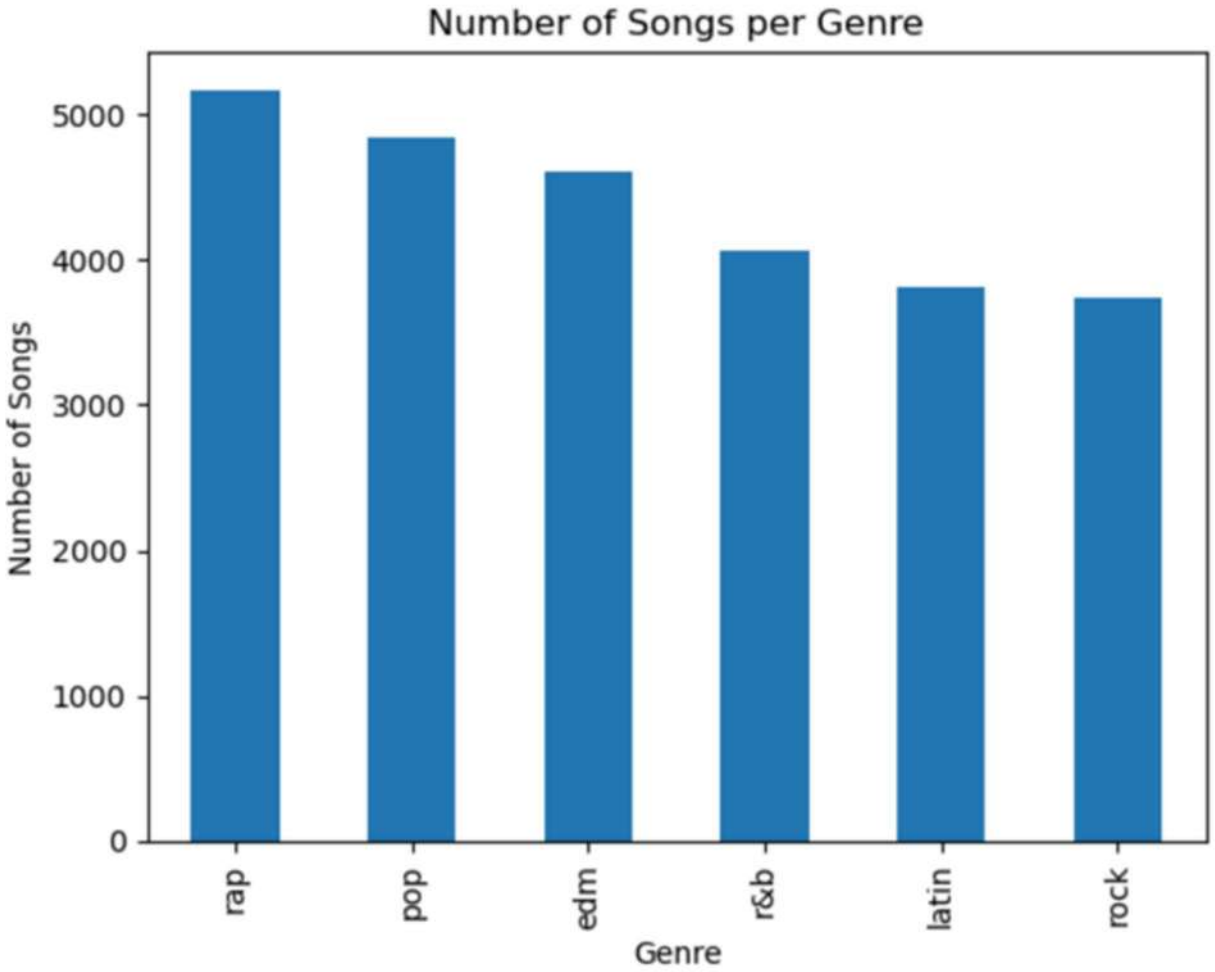






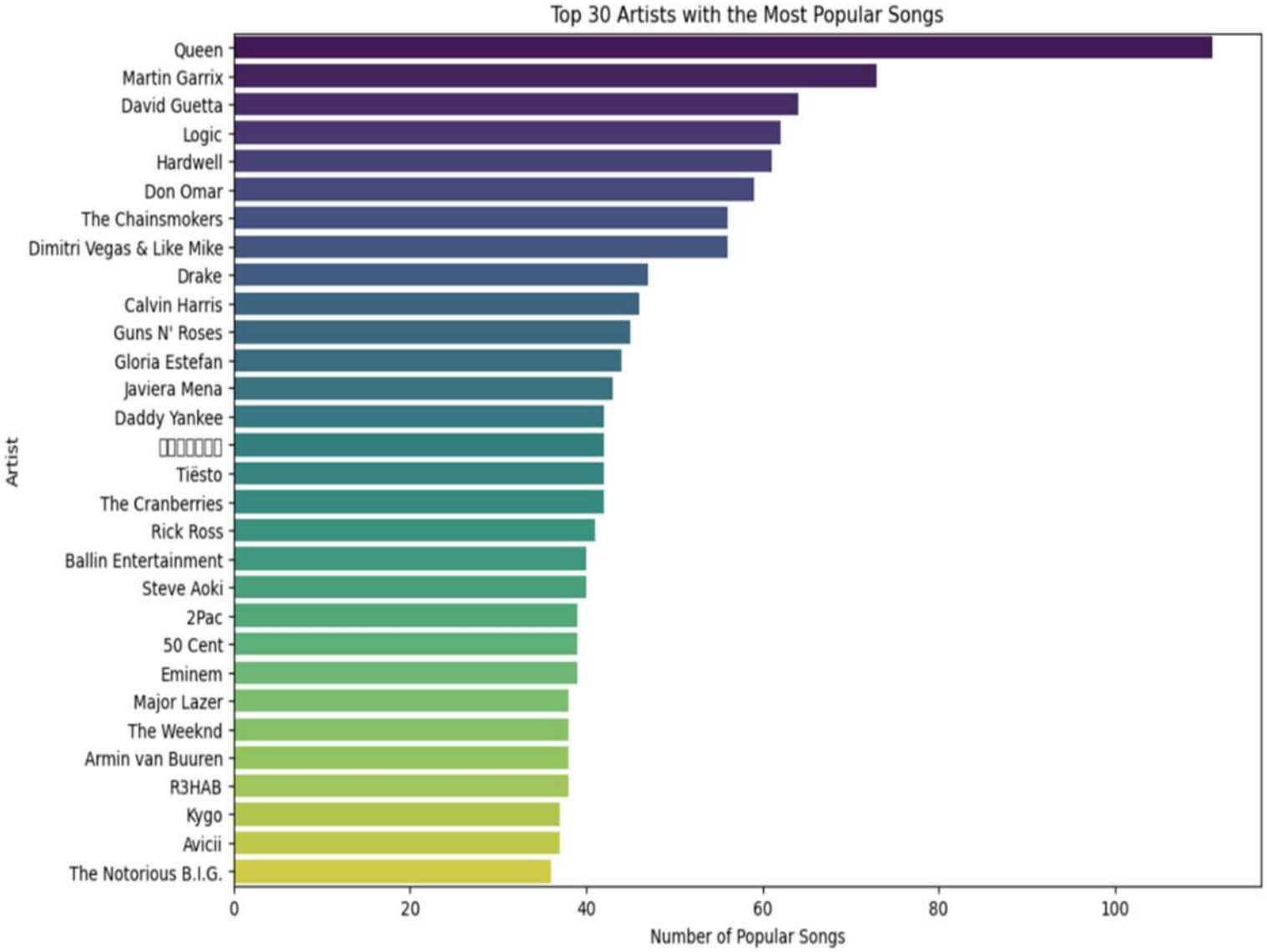
These histograms were not exactly symmetrical and bell-shaped and were skewed. Some results are included below. Consider the plot for response track popularity. It is close to a normal distribution but there is one big outlier. This could represent songs that are not popular and this makes sense because not every single song released is received well by listeners of music around the world. The most common popularity rating is 0 for more than 2500 songs. This confirms are former assumption of not all songs being a hit song. Similar conclusions can be drawn from other feature distributions. Even though most of these features have skewed distributions, we will not be performing any transformations. This is because we will be standardizing them later on.

Rap is the most popular genre and rock is the least popular. This could be because this dataset consists of songs from 1950s - 2020 and rap songs started originating in the 70s and are still very popular. Even songs that are not rap might have a few seconds of a rap portion in them.



Since, our goal is to predict hit songs, it is interesting to take a look at what artists have had popular songs. This plot below shows that 'Queen' has the most popular song and 'The Notorius B.I.G.' (one of my personal favorites) has the least. This is because the latter was murdered in the late '90s at the peak of his career implying that he had hit songs but couldn't make any more because of his tragic demise and he was a very famous rap artist too which sort of reflects on the previous plot. The former also died at the peak of his career but was a part of a rock band that continued to work even after his untimely death. This band was very popular

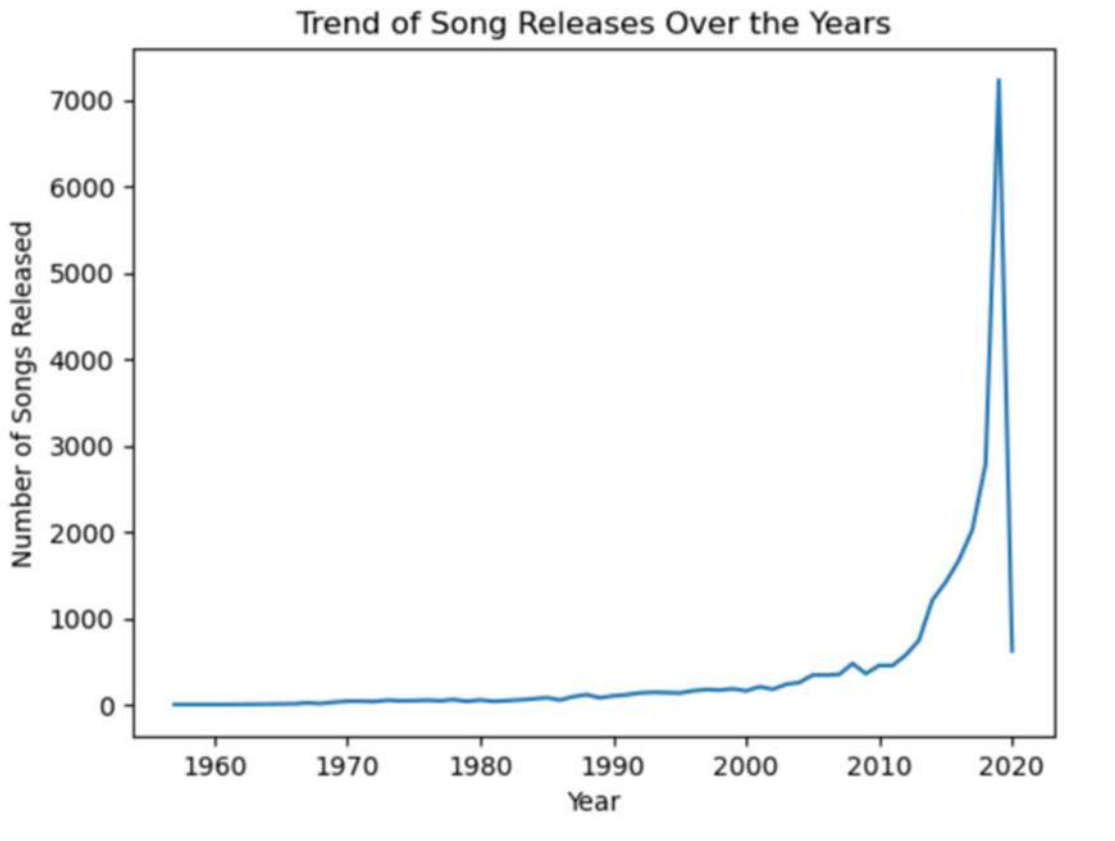
because of their world-renowned stage performances and electrifying costumes. The time that this band was famous was around the 70s - 80s which was also a peak time in the fashion world.



The above popularity of 'Queen' could also be because of the number of songs they have released over the years. They have released over 111 songs and are the artist with the highest amount of songs in this dataset. From this we can infer that artists who have maximum number of songs tend to have more popular sings and hence have more hits.

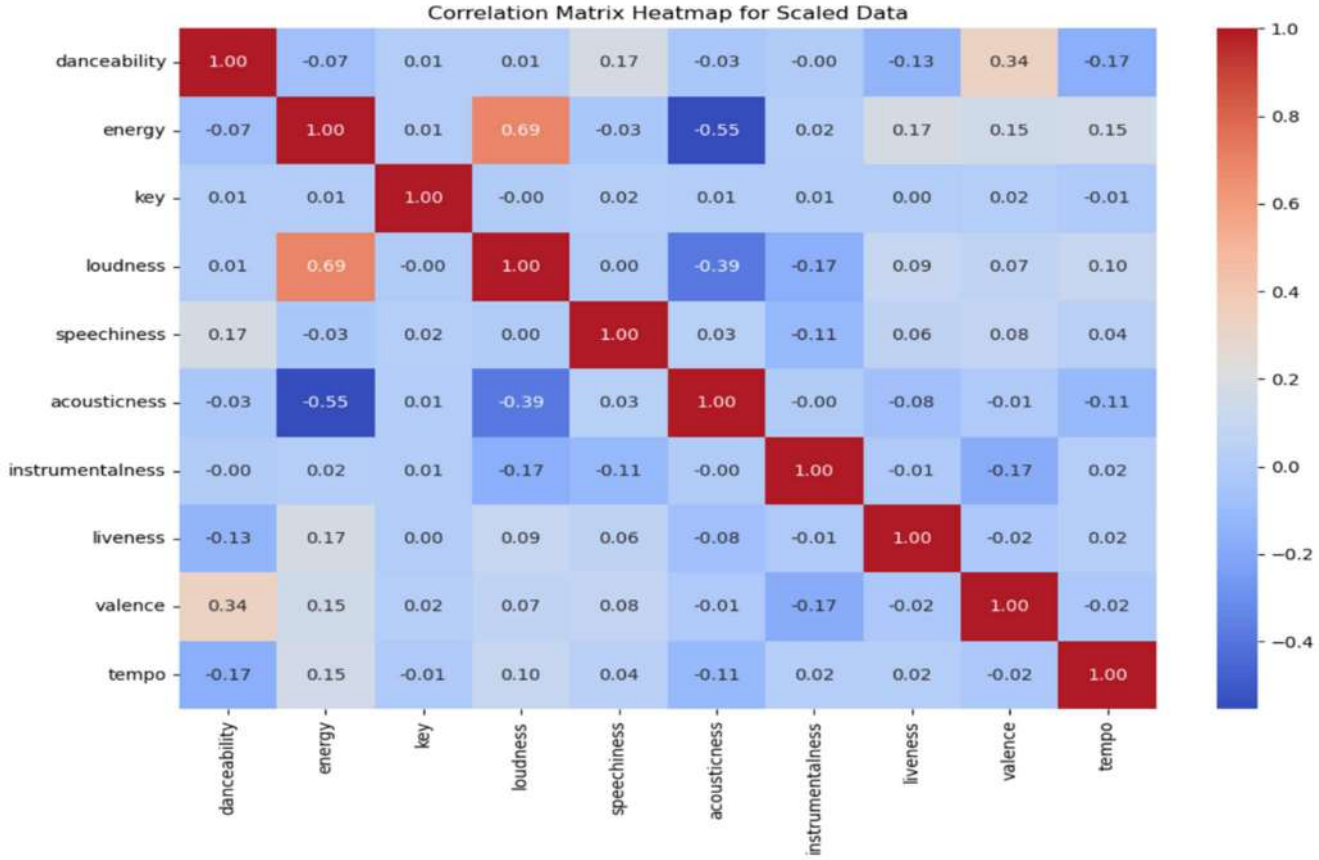
	Artist	Number of Songs
0	Queen	111
1	Martin Garrix	73
2	David Guetta	64
3	Logic	62
4	Hardwell	61
...	...	...
10687	Irvine	1
10688	Andrew W. Boss	1
10689	Scott Krokoff	1
10690	John Belthoff	1
10691	Mat Zo	1

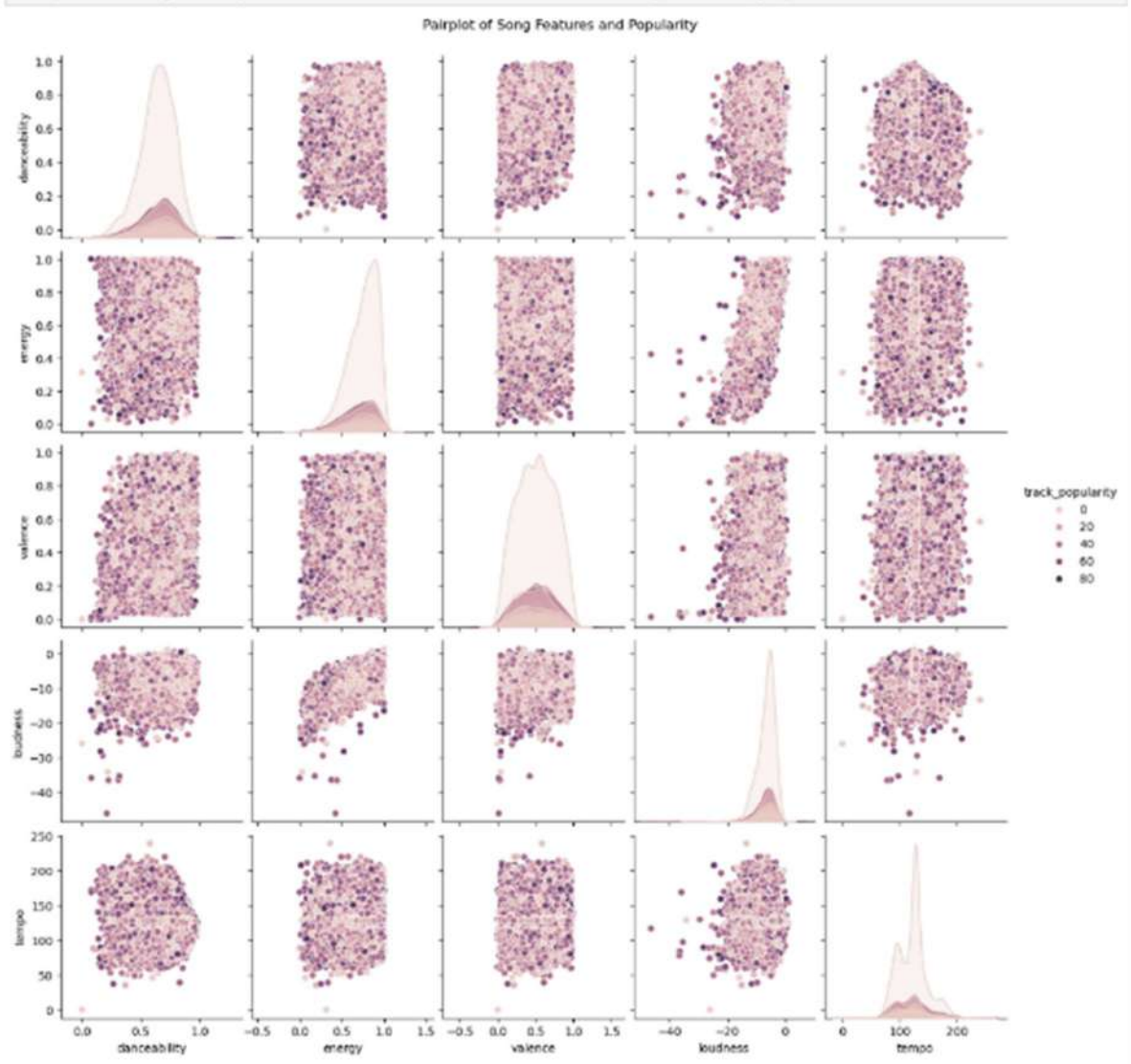
```
Artist with the most songs: Queen (111 songs)
Artist with the least songs: Kevcody (1 song)
```



An interesting observation was obtained after plotting the trend of song releases between the years 1960 - 2020. We see that there are not more than 1000 song releases per year for over 50 years starting from the 1960s up until approximately 2012. This reflects the lack of technology or other music industry prospectives. The music industry was very slow in the beginning when things were developing in that field. But we see a steady increase after 2010 and this is due to the rise of internet related services like YouTube or the rise of music devices being introduced and most people being able to afford these devices and services. But there is a sharp decline during 2020 because of the COVID-19 pandemic and the entire world being quarantined and shut down leading to a decline of artists making music and trying to make a life during the time.

To detect multicollinearity between features, we use a scatter plot (pair plot). This was not easy to interpret so we use our classic method of correlation matrix and heatmaps. Variables like energy and loudness are highly positively correlated. This is justifiable because songs with high energy tend to be louder depicting the energy of the song and vice-versa. Features like acousticness and energy are highly negatively correlated. This is also intelligible because songs with high acoustics make use of more instruments and instruments vary in the kind of sounds they make, especially songs that have low energy are usually just played with instruments than words.


Correlation Matrix Heatmap for Scaled Data
danceability 1.00 -0.07 0.01 0.01 0.17 -0.03 -0.00 -0.13 0.34 -0.17
energy -0.07 1.00 0.01 0.69 -0.03 -0.55 0.02 0.17 0.15 0.15
key 0.01 0.01 1.00 -0.00 0.02 0.01 0.01 0.00 0.02 -0.01
loudness 0.01 0.69 -0.00 1.00 0.00 -0.39 -0.17 0.09 0.07 0.10
speechiness 0.17 -0.03 0.02 0.00 1.00 0.03 -0.11 0.06 0.08 0.04
acousticness -0.03 -0.55 0.01 -0.39 0.03 1.00 -0.00 -0.08 -0.01 -0.11
instrumentalness -0.00 0.02 0.01 -0.17 -0.11 -0.00 1.00 -0.01 -0.17 0.02
liveness -0.13 0.17 0.00 0.09 0.06 -0.08 -0.01 1.00 -0.02 0.02
valence 0.34 0.15 0.02 0.07 0.08 -0.01 -0.17 -0.02 1.00 -0.02
tempo -0.17 0.15 -0.01 0.10 0.04 -0.11 0.02 0.02 -0.02 1.00
danceability
energy
key
loudness
speechiness
acousticness
instrumentalness
liveness
valence
tempo
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Pairplot of Song Features and Popularity
danceability
energy
valence
loudness
tempo
track_popularity
0
20
40
60
80

Because of these high correlations there is a high chance of multicollinearity and we rectify this situation by looking at Variance Inflation Factor (VIF). If there is correlation between features then there is high variance and this is determined by VIF. If VIF ≥ 10 there is presence of multicollinearity and this will result in contradictory results when deciding which feature is insignificant enough that it can be dropped. The remedy for this is to center and scale the features. Once we do the standardizing we obtain all VIFs ≤ 10 which indicates that there is no correlation between any of the feature variables and we will have accurate analysis results. In this project, we used the scaled data from now on.

	Feature	VIF
0	danceability	18.627477
1	energy	19.482459
2	key	3.171256
3	loudness	7.561613
4	speechiness	2.248755
5	acousticness	2.104711
6	instrumentalness	1.291220
7	liveness	2.615568
8	valence	6.861232
9	tempo	18.228049

Before standardizing

	Feature	VIF
0	danceability	1.286881
1	energy	2.720484
2	key	1.001452
3	loudness	2.114826
4	speechiness	1.061658
5	acousticness	1.488041
6	instrumentalness	1.143523
7	liveness	1.055531
8	valence	1.258983
9	tempo	1.061393

After standardizing

Model Fitting

(a): Multiple Linear Regression and linear regression with polynomial features

Since we are trying to predict a quantitative response, we start with the most simplest and obvious choice – Linear regression. To select the most significant features for this model, we use the method of forward stepwise selection. In this procedure, "best" subset of the feature variables are sequentially added at each step and tested if they contribute to the model significantly. Using this method, we obtained all features are significant except feature variable 'key' and hence we continue with this best model to obtain **a test $R^2 = 0.056$ and a training $R^2 = 0.05$,** which is very low and hence the fit is not good because using linear regression for both training and test data we can only explain **about 5%** of the proportion of variance. For a good model, we are aware that R^2 must be higher. Therefore, we add polynomial features to see if this issue persists or improves.

Multiple linear regression model:

track_popularity = 39.757 + 0.812 * danceability – 4.594 * energy + 4.08 * loudness – 0.498 * speechiness + 1.274 * acousticness – 2.257 * instrumentalness – 0.663 * liveness + 0.666 * valence + 0.767 * tempo

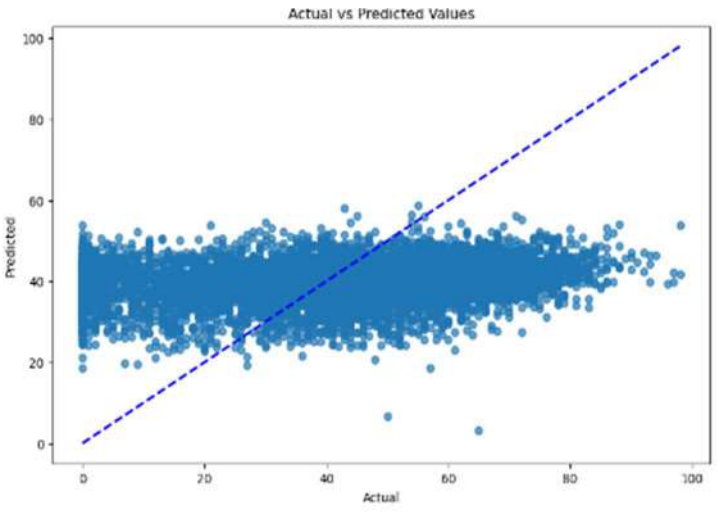
Polynomial linear regression model (providing an equation is inefficient because of the increased number of coefficients):

```
Intercept (beta_0): 39.63815146880444
              Feature  Coefficient
0        danceability     1.213027
1              energy    -4.315998
2                 key    -0.100787
3            loudness     4.330121
4         speechiness    -0.856083
..                ...          ...
60  liveness valence     0.055730
61    liveness tempo    -0.117265
62         valence^2    -0.693602
63     valence tempo     0.000106
64           tempo^2     0.462841
```

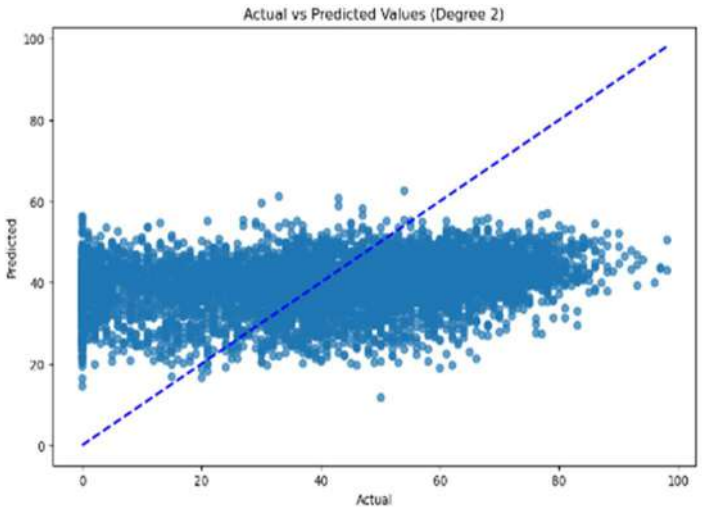
Notice that many coefficient estimates for degree 1 features are very close to that of multiple linear regression estimates above

With polynomial features of degree more than 2, the system crashed and couldn't handle the exponential growth. Hence, we stick to a polynomial of degree 2 and obtain a **test R^2 = 0.068 and training R^2 = 0.071,** which is better than multiple linear regression but still very low.

Finally, we conclude that a regression model does not provide a good fit and we must try other models for model performance and making predictions. The figure below depicts how incorrect our predictions would be if we considered a linear regression model.



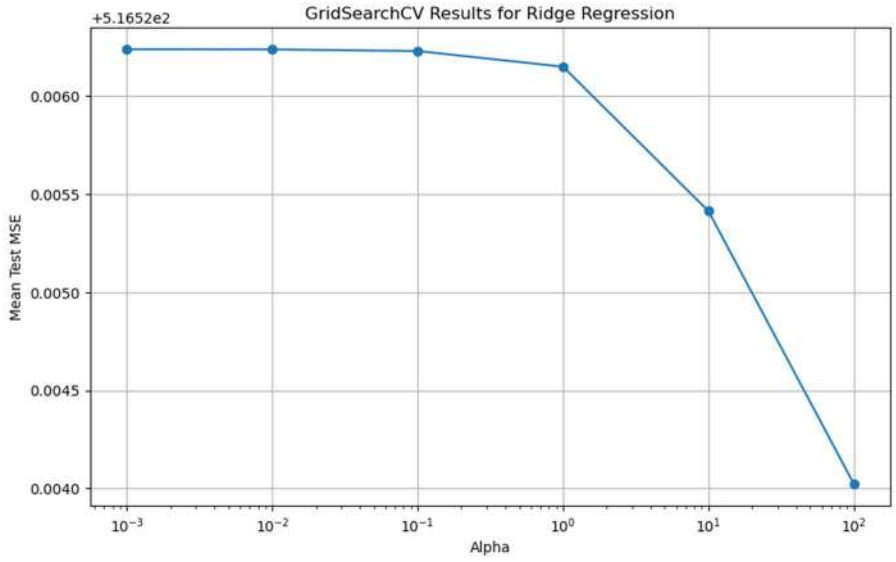
Multiple linear regression



Linear regression with polynomial features

(b): Ridge Regression

We use a ridge regression model to predict track popularity next. This is because of the presence of multicollinearity we detected in our analysis previously. Ridge regression is one of the remedies for this issue just like VIF. We use hyperparameter tuning for this model to reduce high variances. This model is less flexible and increase in bias is definitely less than the decrease in variance. As tuning parameter λ increases, training error also increases while test error initially decreases but increases gradually. To counter this issue we have tuned hyperparameter $\alpha = \{0.001, 0.01, 0.1, 1, 10, 100\}$. Using 5-fold cross validation and 'GridSearchCV' we determine our best tuning parameter to be 100. This can be seen from the elbow in the plot.




```
Training MSE: 515.6700
Training RMSE: 22.7084
Test MSE: 498.4962
Test RMSE: 22.3270
Training R^2: 0.5019
Testing R^2: 0.0559
```

	Feature	Coefficient
0	danceability	0.819008
1	energy	-4.497691
2	key	-0.053079
3	loudness	3.997916
4	speechiness	-0.491948
5	acousticness	1.287285
6	instrumentalness	-2.261747
7	liveness	-0.667259
8	valence	0.651961
9	tempo	0.759389

Model evaluation metrics: **Test R^2 = 0.055 and Training R^2 = 0.50.** The test R^2 is very low whereas the training R^2 is quite high. This indicates that we may be overfitting slightly. Note that training RMSE and test RMSE are 22.70 and 22.32 respectively. This means that this model performs quite well on both the test and the training data despite the low R^2 value for test data.

Hence, **we can conclude that ridge regression is better than multiple linear regression (with and without polynomial features)** but we can do better to avoid overfitting.

(c): Random Forest

We trained a random forest model without hyperparameter tuning first with the following hyperparameters:

- Number of trees = 150
- Maximum depth of each tree in the forest = 20
- Minimum number of samples required to split an internal node = 5
- Minimum number of leaf nodes of each subtree = 2 (binary tree)

For this model we also include interaction terms between 'danceability' and 'energy' to improve model performance on test data since the previous models performed very poorly on it. We also transform feature variable 'liveliness' which was heavily right-skewed (obtained in one of the histograms in EDA for numerical feature variables). After all these we fit the model, make predictions appropriately and achieve a **high training $R^2 = 0.75$ and a very low test $R^2 = 0.076$** which is a classic depiction of overfitting. This makes sense because random forests tend to overfit data.

This model considered the following feature variables as important ones in order of increasing feature importance measures:

```
Training R^2: 0.7507
Testing R^2: 0.0766

Top Feature Importances:
            feature  importance
8             tempo    0.119266
2          loudness    0.112008
4      acousticness    0.108233
3       speechiness    0.102962
5  instrumentalness    0.099835
```

To overcome this overfitting problem, we decided to tune our hyperparameters hoping that model performance will definitely improve if tuning was done. Unfortunately, the program kept crashing after a run time of approximately 2 hours. So, to avoid Jupyter Notebook from crashing, this code chunk was deleted from the notebook and hence not included in analysis and report and hopefully will be worked on in the future after some more analysis and work on feature engineering to avoid the large runtime by reducing feature dimensions; but maybe done on another statistical software like 'R' which gives very interpretable results and is easier to work with for data analysis.

(d): Gradient Boosting

Since we have observed that including interaction terms and transformations improve training R^2, we continue to do so for gradient boosting as well. We use the same interactions as in random forest but include a couple more based on the feature variables that are strongly correlated with each other after taking a closer look at the heatmap generated in EDA. These include acousticness * valence, dance * speechiness, energy * liveliness. The transformations are also done not just for liveliness but also for tempo (log), energy (squared because left skewed) and, loudness (squared because left skewed). And then we proceed just like we did for random forest.

First, we fit the gradient boosting model **without tuning the hyperparameters** to see the difference between models that are not tuned and models that have tuned hyperparameters. This model has a **test R^2 = 0.0793 and a training R^2 = 0.296**. These values are very low and are similar to the results of multiple linear regression we did at the beginning. Since the model performance is low for both, we train and fit a gradient boosting model with hyperparameter tuning.

```
Model Performance:
Training R^2: 0.2960
Testing R^2: 0.0793

Cross-Validation R² Scores: [0.05690778 0.06335279 0.07472757 0.0762595  0.06749369]
Mean CV R²: 0.06774826481097129

Top 10 Feature Importances:
                  feature  importance
5       instrumentalness    0.115309
4           acousticness    0.074214
3            speechiness    0.068984
0           danceability    0.068545
7                valence    0.067326
11             log_tempo    0.065644
8                  tempo    0.064650
6               liveness    0.062464
14     dance_speechiness    0.059036
10  acousticness_valence    0.057984
```

The model with hyperparameter tuning contains the following parameter grid:

- Number of estimators in the boosting step = {100, 150, 200}
- Learning rate = {0.01, 0.1}
- Maximum depth of each tree = {3, 5}
- Minimum number of splits required to split an internal node = {5, 10}
- minimum number of subsamples required to fit each tree = {0.8, 1}

This model fortunately gave very good results. Test **R^2 = 0.8959 and training R^2 = 0.8742.** This model helps explain more than 80% of the proportion of variance for both test and training data. Note that we have included less hyperparameters and performed 3-fold CV to decrease run time from hours to minutes. The output below also has all information about the best hyperparameters that resulted in the aforementioned model performance metrics.

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
Best Hyperparameters: {'subsample': 0.8, 'n_estimators': 100, 'min_samples_split': 5, 'max_depth': 3, 'learning_rate': 0.1}
Training R^2: 0.8742
Testing R^2: 0.8959

Top 10 Feature Importances:
             feature  importance
5   instrumentalness    0.204180
11         log_tempo    0.086177
4       acousticness    0.079133
13  loudness_squared    0.075239
0       danceability    0.069565
8              tempo    0.068974
2           loudness    0.067609
3        speechiness    0.050851
7            valence    0.045449
6           liveness    0.044150
```

Conclusion

Looking at all the models above we can build a summary table to make our conclusions based on the inferences done in the above sections:

Model	Training data			Test Data		
	R^2	MSE	RMSE	R^2	MSE	RMSE
Multiple linear regression	0.0502	515.66	22.708	0.056	498.47	22.326
Multiple linear regression with Polynomial features of degree 2	0.0715	504.12	22.452	0.0683	498.95	22.337
Ridge regression	0.5019	515.67	22.708	0.0559	498.49	22.327
Random forest	0.7507	-	-	0.0766	-	-
Gradient Boosting	0.296	-	-	0.0793	-	-
Gradient Boosting with hyperparameter tuning	0.8742	-	-	0.8959	-	-

Based on above summary, ridge regression and random forest overfit our data and hence should not be considered in prediction of track popularity. **Instead, we can use gradient boosting models with hyperparameter tuning to do so since the model $R^2 \geq 0.80 = 80\%$ of the proportion of variance is explained by both test and training data.**

Future

For this project, I initially wanted to do genre classification but data pre-processing for this was very time consuming and I never seemed to be satisfied with it and often came to a dead end when I started building models. Especially with logistic regression, misclassification rates were always ≥ 80% and could not be rectified unless some advanced feature engineering was done; for which I did not have sufficient amount of time. But it performed better after the outliers were removed; I was not willing to remove outliers and influential points completely from my analysis because they were informative.

In the future, I would like to improve model performance by creating more interaction terms for Random Forest and try implementing AdaBoostRegressor to see if model performance for test data improves. Another goal with this dataset would be to build a recommendation system which is very popular in these times and can help with producing a personal playlist for every Spotify subscriber.