Popularity Prediction and Recommendation of Music

By

Raja Gopala Reddy C

Vamsi Mulpuri

Project Goal

- To analyze and choose the right set of minimal features amongst the high dimensionality of the data and to predict the popularity score of a particular song.
- This Prediction helps any content creator/musician to understand what music listeners prefer to hear more nowadays which is key in order to compete in the market.
- This can be achieved using machine learning algorithms like the Linear Regression, Decision trees, Random Forest and other regression algorithms.

Insights about the Data

Our dataset comprises of the following features:

- Input
  - id: Title identifier
  - name: Title of the song
  - duration_ms: The duration of the track
  - explicit: It refers to a particular song if it has curse words or not.
  - artists: Name of the artist
  - id_artists: Artist identifier
  - release_date: The date when the track is released.
  - danceability: 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.
  - energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
  - key: The estimated overall key of the track.
  - loudness: The overall loudness of a track in decibels (dB).
  - mode: Mode indicates the modality (major or minor) of a track.
  - speechiness: It detects the presence of spoken works in a track.
  - acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
  - instrumentalness: Predicts whether a track contains no vocals.
  - liveness: It referes to reverberation time.
  - valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
  - tempo: The overall estimated tempo of a track in beats per minute (BPM).
  - time_signature: Time stamp of the song.
- Output
  - popularity score(0-100): The popularity of the track

Technical Problem Formulation

	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness	tempo	valence	year
49028	0.170000	['Electric Light Orchestra']	0.640	243573	0.878	0	3b5FY5QJnk1TNJeQO8DsvG	0.000232	2	0.0632	-6.481	1	Don't Bring Me Down	39	1973	0.0335	115.700	0.795	1973
16213	0.448000	['Yunginky']	0.669	91193	0.243	0	0xUAbAov8hiYihtJ8kVsMH	0.000000	8	0.0610	-15.027	1	2021	14	2020-12-22	0.1520	158.015	0.523	2020
36329	0.000023	['Rank 1']	0.456	100453	0.901	0	4b70D8Ubo30QKTXtkV2Yak	0.212000	4	0.2680	-12.414	0	Top Gear [ASOT 229] - Original Mix	7	2005-12-29	0.1070	138.535	0.176	2005
46181	0.845000	['Blossom Dearie']	0.637	186693	0.403	0	16UdsCC8mjirUrtjA9HaG5	0.000001	7	0.2510	-12.847	1	Chez Moi	26	1959-01-01	0.1650	93.458	0.525	1959
28434	0.529000	['The Miracles']	0.485	169227	0.661	0	1VMncXCJ34SIAosUKhzjHy	0.000000	7	0.6320	-7.240	1	Shop Around	34	1965-01-01	0.0338	133.858	0.817	1965

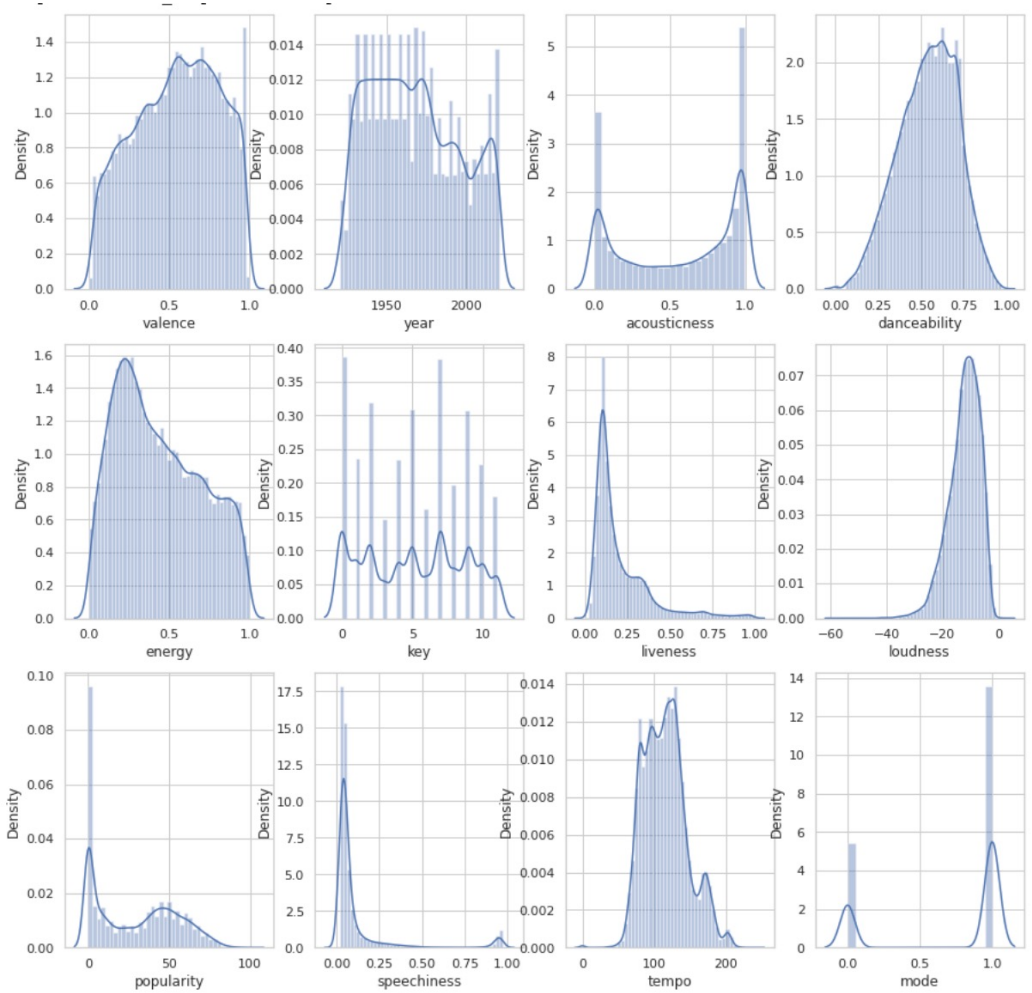
Here, we have removed features that are not required for this prediction of ours.

And as you can see, we have calculated the required statistics of the features of all the features of our dataset.

	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speechiness	tempo	valence	year
count	50000.000000	50000.000000	5.000000e+04	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	0.562788	0.541857	2.208295e+05	0.441963	0.064900	0.191349	5.198560	0.205726	-12.164000	0.713960	29.642300	0.114162	115.944877	0.542638	1967.874100
std	0.377602	0.170482	1.350002e+05	0.264567	0.246352	0.331485	3.488441	0.168107	5.529927	0.451913	24.896397	0.201400	30.393947	0.257811	27.966741
min	0.000000	0.000000	5.991000e+03	0.000000	0.000000	0.000000	0.000000	0.000000	-60.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1920.000000
25%	0.156000	0.423000	1.618670e+05	0.220000	0.000000	0.000000	2.000000	0.101000	-15.375000	0.000000	3.000000	0.035300	92.829000	0.340000	1944.000000
50%	0.662000	0.554000	1.954930e+05	0.402000	0.000000	0.000391	5.000000	0.140000	-11.450500	1.000000	30.000000	0.045850	114.491000	0.559000	1965.000000
75%	0.941000	0.671000	2.516670e+05	0.654000	0.000000	0.212000	8.000000	0.263000	-8.073000	1.000000	51.000000	0.078300	133.690250	0.754000	1990.000000
max	0.996000	0.988000	4.800118e+06	1.000000	1.000000	1.000000	11.000000	0.997000	3.855000	1.000000	100.000000	0.970000	243.372000	1.000000	2021.000000

Exploratory Data Analysis

Histograms for our Input Features



- From the plots, we can see that danceability looks nearly like a Gaussian distribution(bell curve).
- Liveness, Speechiness looks more likely to be right skewed meaning that mean is greater than the median.
- Loudness is left skewed meaning that mean is to the left of the median.
- Acousticness and Mode has two peaks which is bimodal.
  - You could split them into two different variables(high_var1 and low_var1) and do multi-variable regression
  - Ignore this bimodal variable.
- Rest of the features are not uniform

As expected, popularity is highly correlated with the year released.

Energy also seems to influence a song's popularity. Many popular songs are energetic. Because the correlation here is not too high, low energy songs do have some potential to be more popular.

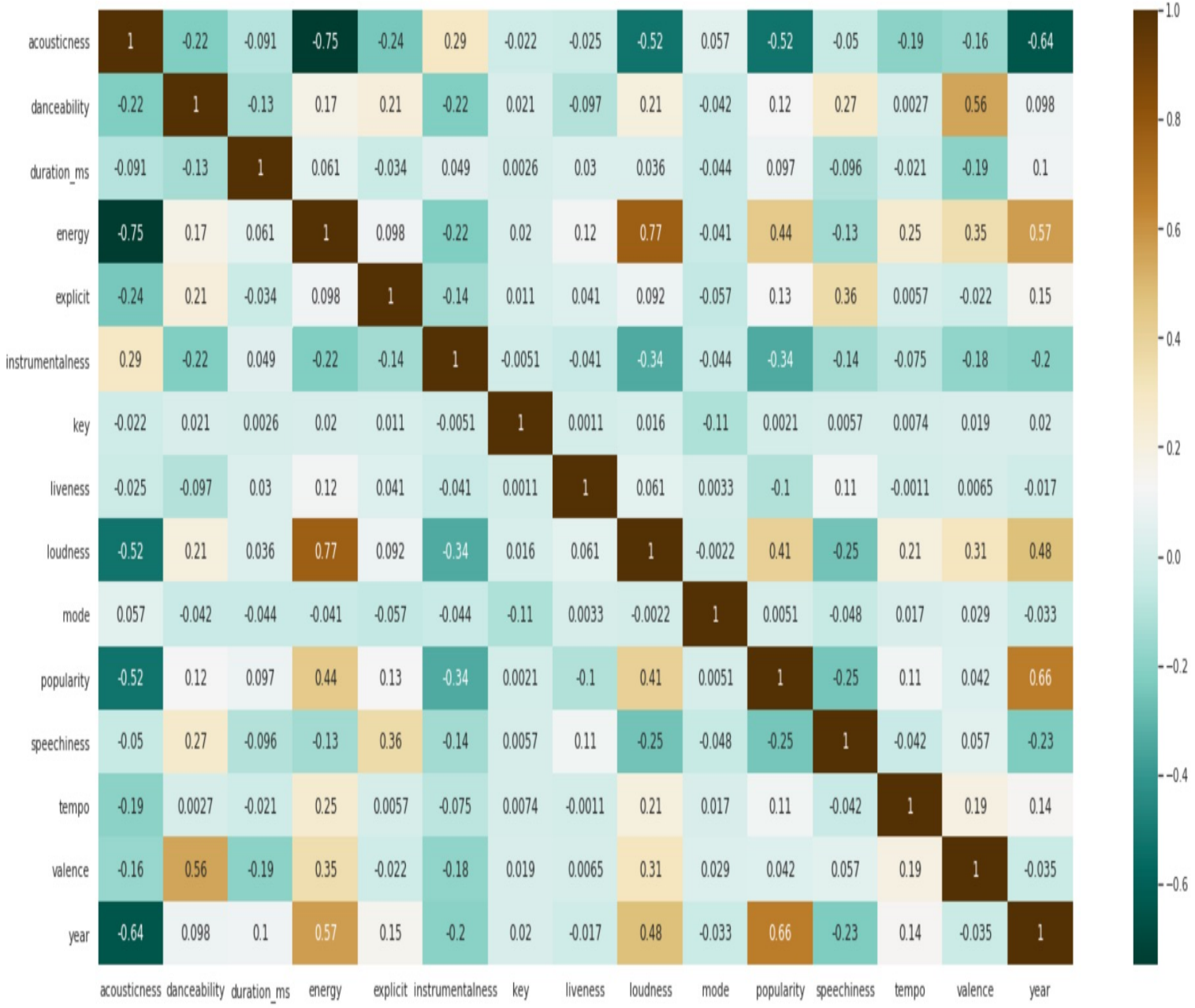
Acousticness seems to be uncorrelated with popularity. Most popular songs today have either electronic or electric instruments in them. It is very rare that a piece of music played by a chamber orchestra or purely acoustic band becomes immensely popular (though, again, not impossible).

Loudness and energy are highly correlated. This makes some sense as energy is influence by the volume the music is being played at.

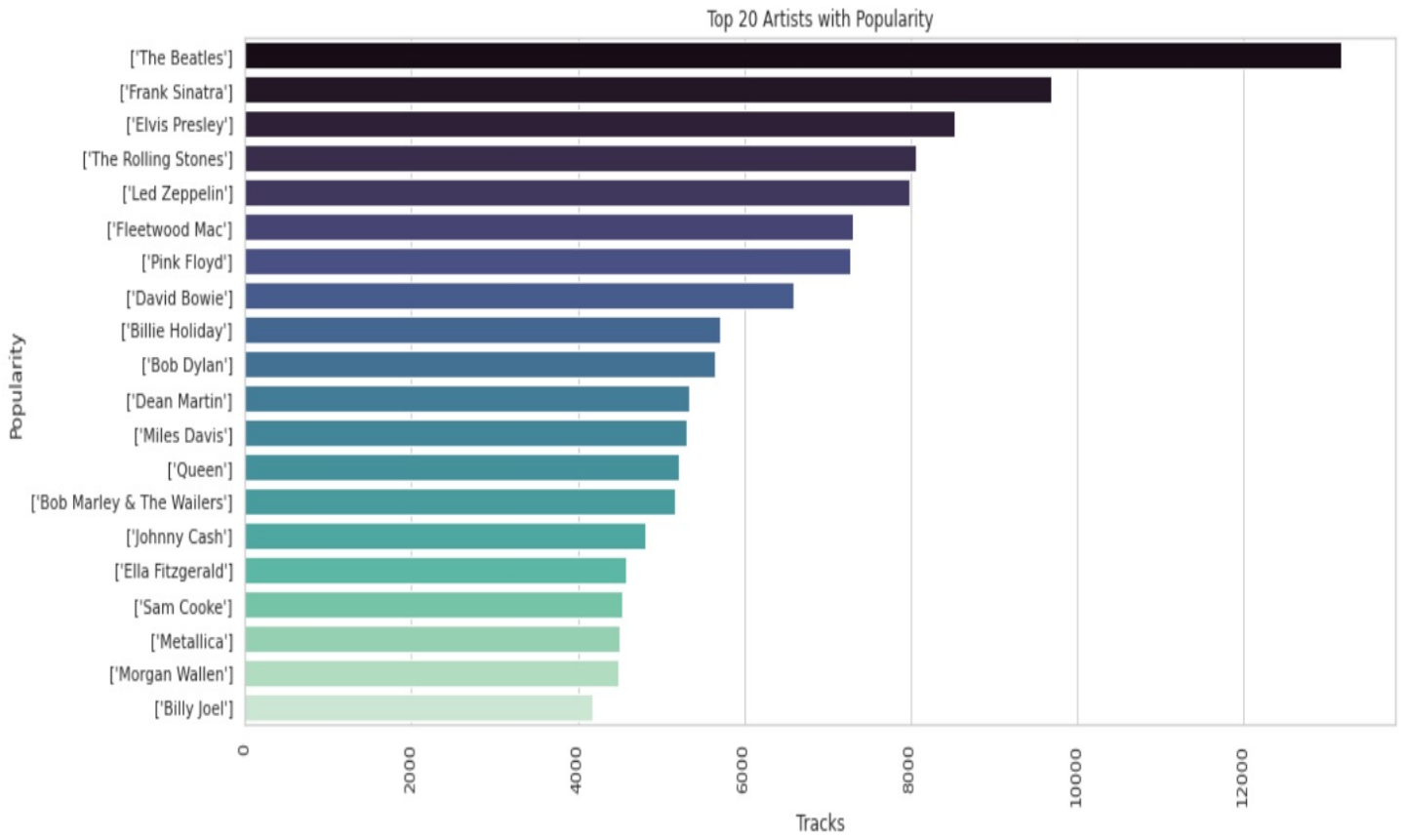
Acousticness is highly negatively correlated with energy, loudness, and year.

Valence and danceability are highly correlated.

Thus, from this data, it would be better for an artist to create a high energy song with either electric instruments or electronic songs to have the best chance at generating the most popularity.



Top 20 Popular Artists



From the graph, we can infer that More the number of tracks an artist has, the more popular the artist is.

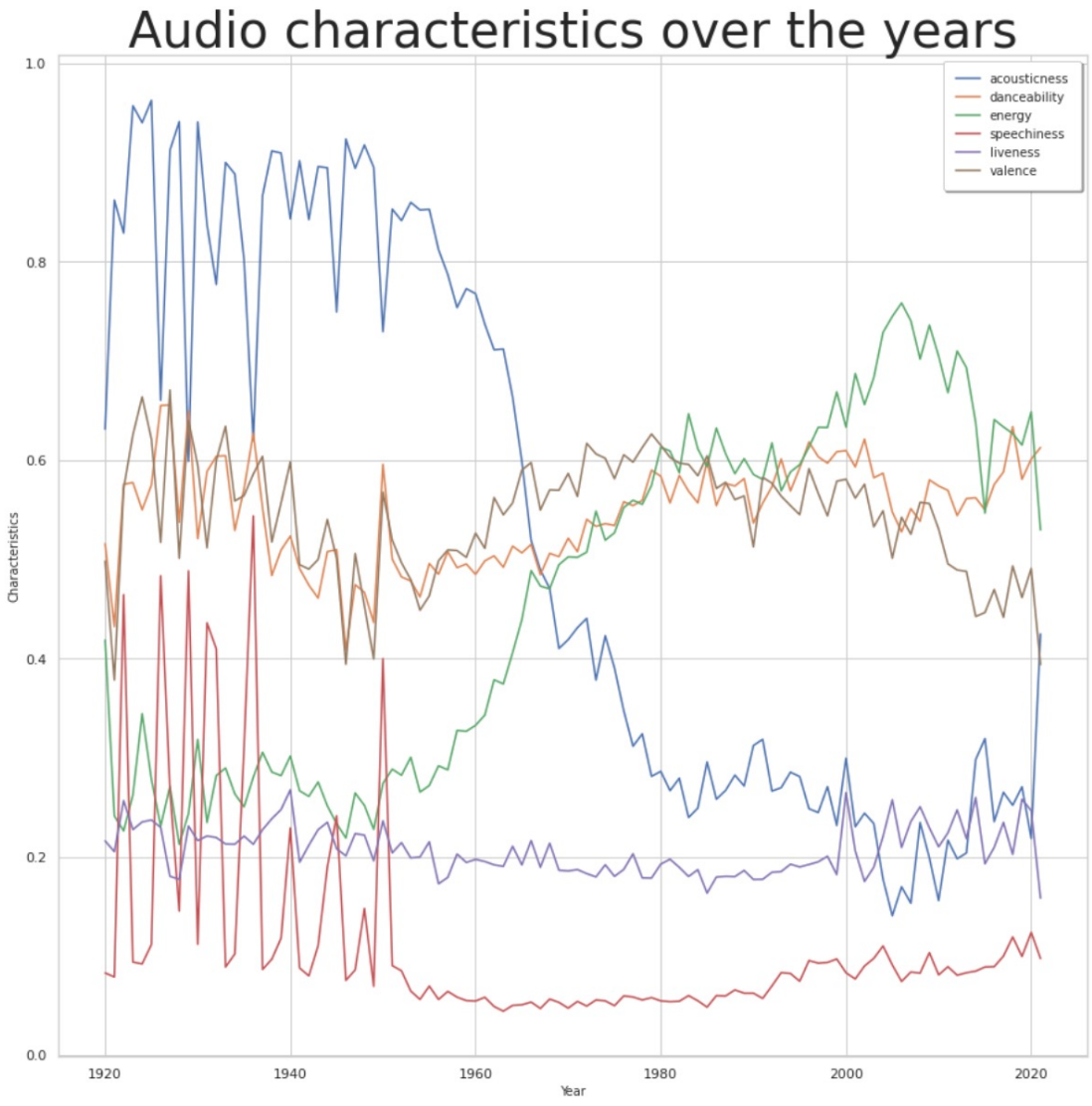
Trends

Acousticness: The characteristic of this feature has been decreasing gradually over the years meaning that acousticness does not play an important role for predicting the popularity.

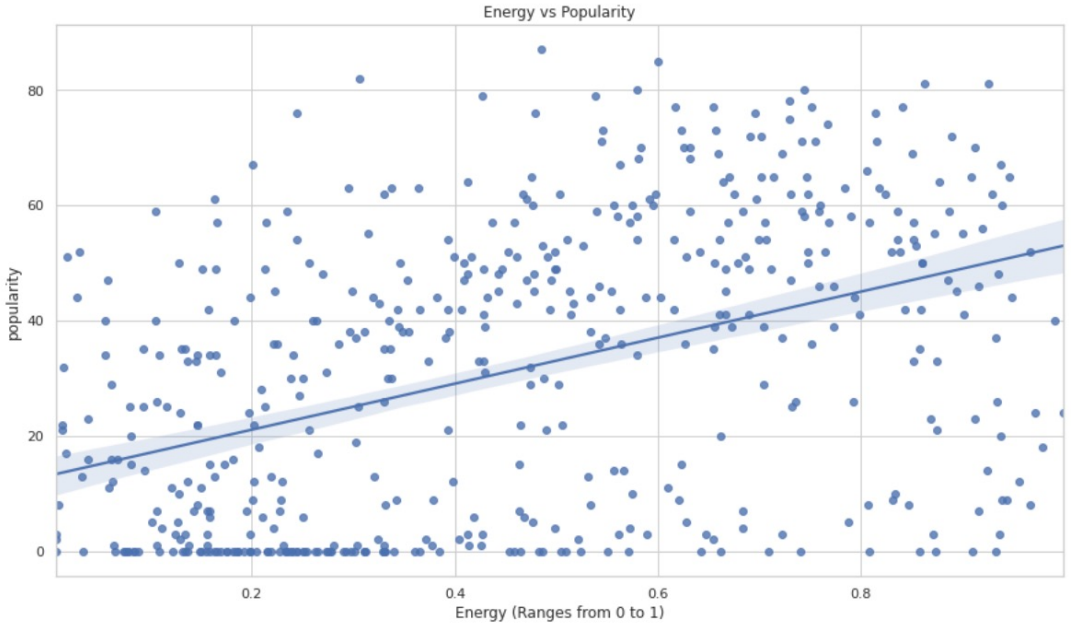
Danceability and Valence: These features are constant over the years meaning that it can play an important role in the popularity of a song.

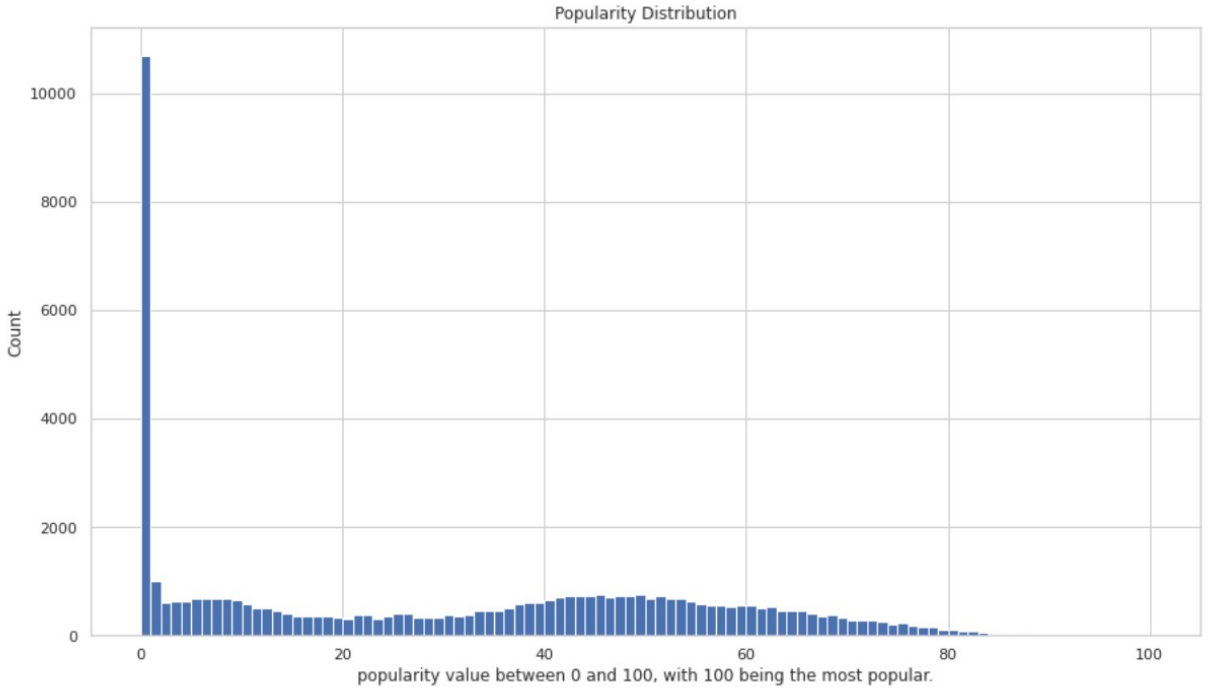
Energy: Over the years, this feature kept on increasing meaning that if a song has higher energy value, more the song will be popular.

Speechiness and Liveness: These features kept on decreasing over the years and seems like they do not play an important role in the popularity of a song.



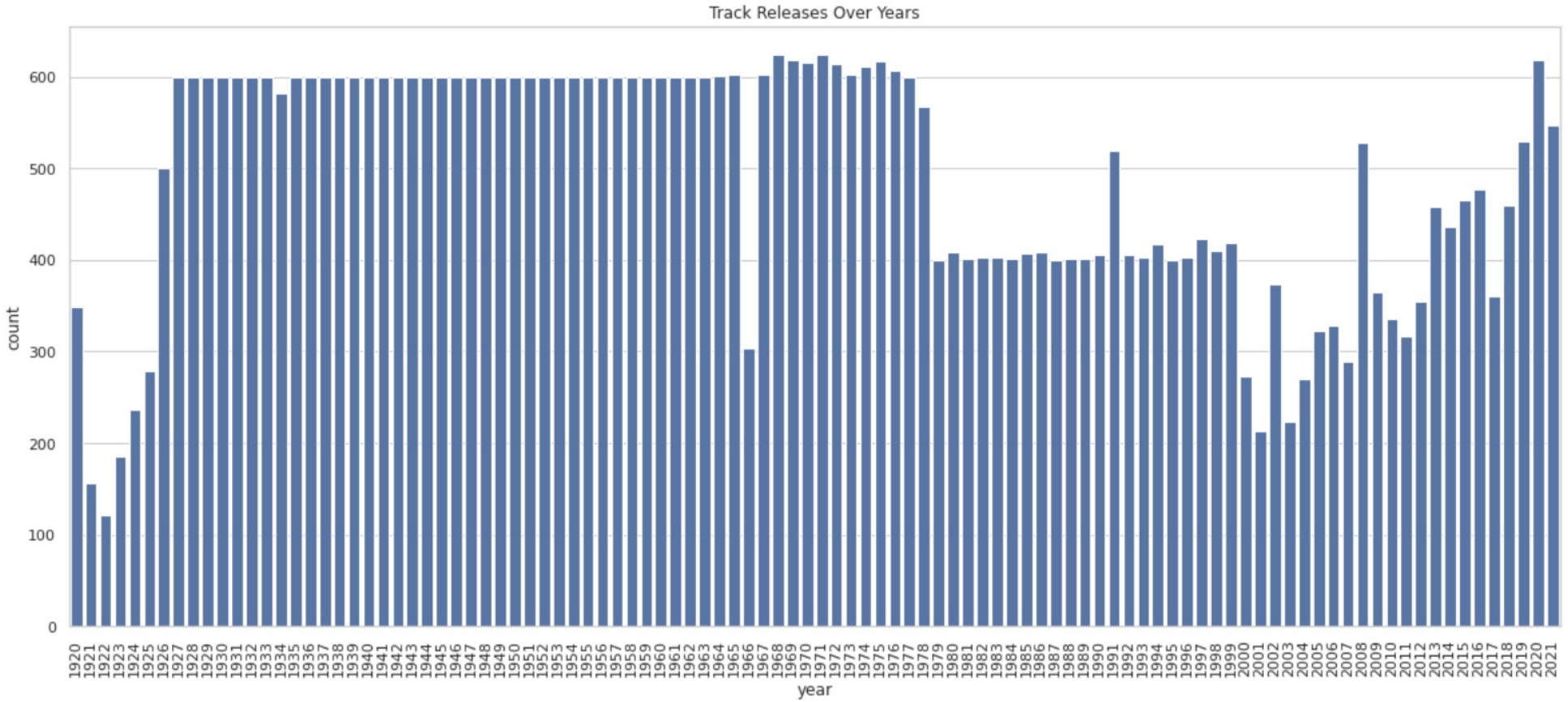
In the Energy vs Popularity graph, we can see that the value of energy kept on increasing with respect to popularity meaning that more energy a song has, the more popular the song would be.





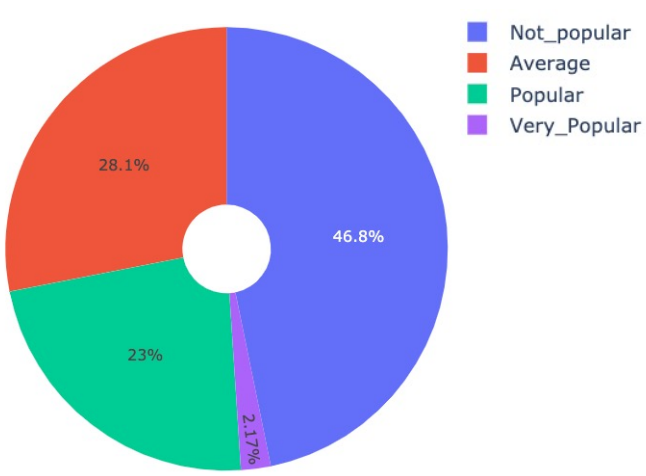
In the Popularity Distribution graph, we can see that a lot of songs has a popularity of 0 and majority of the songs lie in the range between 40 and 60.

We can also see that only a few number of songs has popularity above 80.



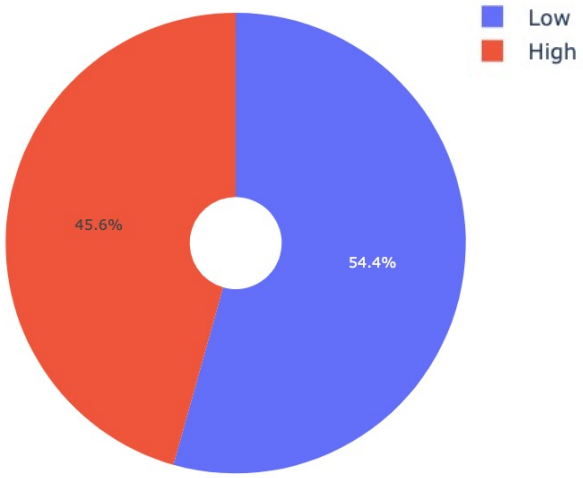
From the above graph, we can see that a greater number of songs were released in the year 2020 and the year with least number of songs is 1992.

Nature of Music



Based on the Popularity feature, the songs are categorized in the following manner:

If a song has range between 75 and 100 - it is Very Popular
If a song has range between 50 and 75 - it is Popular
If a song has range between 25 and 50 - it is Average
If a song has range between 0 and 25- Not Popular



Based on the Energy feature, the songs are categorized in the following manner:

If a song has energy greater than it's mean - it is High
If a song has energy lesser than it's mean - it is low

Model Training

- Feature Scaling
  - Most of the times, our dataset will contain features highly varying in magnitudes, units and range.

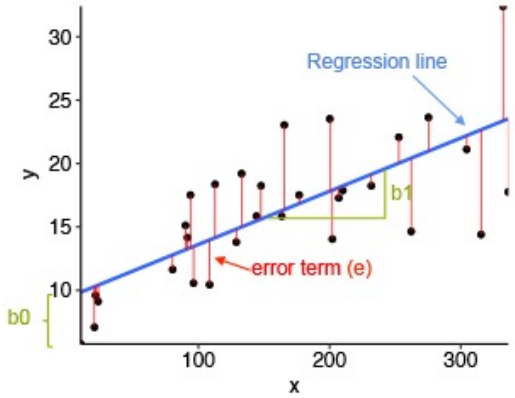
```
 X= data.loc[:,data.columns!='popularity']
y= data.loc[:,data.columns=='popularity']
```

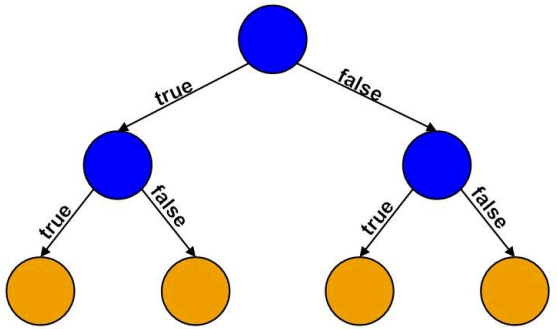
```
from sklearn.preprocessing import StandardScaler
sc_X= StandardScaler()
sc_y= StandardScaler()
X=sc_X.fit_transform(X)
y=sc_y.fit_transform(y)
```

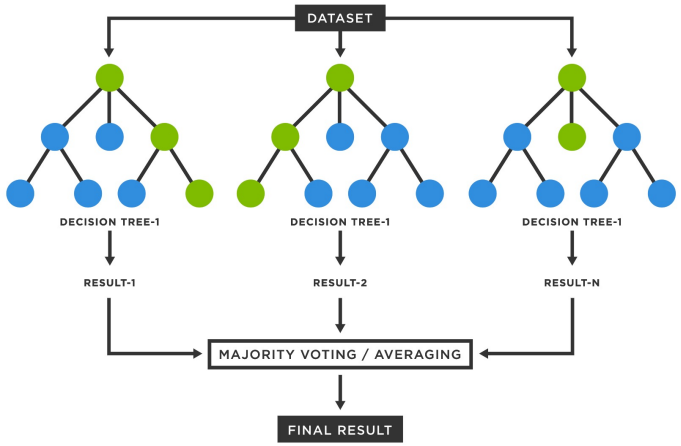
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=0)
```

Algorithms

- Here, we are training our data using the below algorithms and they are as follows:
  - Linear Regression
    - Linear regression attempts to model the relationship between two variables by fitting a linear equation (= a straight line) to the observed data.
  - Decision Tree
    - Decision tree builds classification or regression models in the form of a tree structure. The final result is a tree with decision nodes and leaf nodes. The decision split is made based on Entropy and Information gain.
  - Random Forest
    - A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees







Linear Regression

Here, we are using R2 score and Mean Squared Error as our metrics.

```
from sklearn.linear_model import LinearRegression
regressor_lin=LinearRegression()
regressor_lin.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred_lin = regressor_lin.predict(X_test)
```

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

```
print("R2 Score for training data is: {}\n".format(regressor_lin.score(X_train, y_train)))
print("R2 Score for testing data is: {}\n".format(r2_score(y_test, y_pred_lin)))
print("Mean Squared Error of Linear Regression is: {}\n".format(mean_squared_error(y_test, y_pred_lin)))
```

```
R2 Score for training data is: 0.9311002672593812

R2 Score for testing data is: 0.9307435753046618

Mean Squared Error of Linear Regression is: 0.06956181279263532
```

We can also see that, in our model, the MSE value is closer to 0 meaning that it is a good model
R2 score is nearly closer to 1, meaning that it is a good model.

Random Forest

Here, we are using R2 score and Mean Squared Error as our metrics.

```
from sklearn.ensemble import RandomForestRegressor
regressor_rf=RandomForestRegressor()
regressor_rf.fit(X_train,y_train)

y_pred_rf=regressor_rf.predict(X_test)
```

```
print("R2 score for training data is: {}\n".format(regressor_rf.score(X_train, y_train)))
print("R2 Score for testing data is: {}\n".format(r2_score(y_test, y_pred_rf)))
print("Mean Squared Error of Random Forest Regression is: {}\n".format(mean_squared_error(y_test, y_pred_rf)))
```

```
R2 score for training data is: 0.99450521090747

R2 Score for testing data is: 0.9611032125541107

Mean Squared Error of Random Forest Regression is: 0.039068303893082815
```

We can also see that, in our model, the MSE value is closer to 0 meaning that it is a good model
R2 score is nearly closer to 1, meaning that it is a good model.

Decision Tree

Here, we are using R2 score and Mean Squared Error as our metrics.

```
from sklearn.tree import DecisionTreeRegressor
regressor_dt= DecisionTreeRegressor(random_state=0)
regressor_dt.fit(X_train,y_train)

y_pred_dt= regressor_dt.predict(X_test)
```

```
print("R2 Score for training data is: {}\n".format(regressor_dt.score(X_train, y_train)))
print("R2 Score for testing data is: {}\n".format(r2_score(y_test, y_pred_dt)))
print("Mean Squared Error of Decision Tree Regressor is: {}\n".format(mean_squared_error(y_test, y_pred_dt)))
```

```
R2 Score for training data is: 0.9998301170750055

R2 Score for testing data is: 0.9237829773097948

Mean Squared Error of Decision Tree Regressor is: 0.07655310373457626
```

We can also see that, in our model, the MSE value is closer to 0 meaning that it is a good model
R2 score is nearly closer to 1, meaning that it is a good model.

Hyper Parameter Tuning

```
parameters={"splitter":["best","random"],
            "max_depth" : [1,3,5,7,9],
           "min_samples_leaf":[1,2,3,4,5],
           "min_weight_fraction_leaf":[0.1,0.2,0.3,0.4,0.5],
           "max_features":["auto","log2","sqrt",None],
           "max_leaf_nodes":[None,10,20,30,40,50] }
```

```
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(regressor_dt,param_grid=parameters,scoring='neg_mean_squared_error',cv=3,verbose=3)

grid_cv.fit(X_train,y_train)

print('the parameters are',grid_cv.best_params_)
grid_cv.best_score_
```

```
from sklearn.tree import DecisionTreeRegressor
regressor_dt= DecisionTreeRegressor(max_depth= 3, max_features= 'auto', max_leaf_nodes= None, min_samples_leaf= 1, min_weight_fraction_leaf= 0.1, splitter= 'best')
regressor_dt.fit(X_train,y_train)

y_pred_dt= regressor_dt.predict(X_test)
```

```
print("R2 Score for training data is: {}\n".format(regressor_dt.score(X_train, y_train)))
print("R2 Score for testing data is: {}\n".format(r2_score(y_test, y_pred_dt)))
print("Mean Squared Error of Decision Tree Regressor is: {}\n".format(mean_squared_error(y_test, y_pred_dt)))
```

```
R2 Score for training data is: 0.9011254881624726

R2 Score for testing data is: 0.9021002411361688

Mean Squared Error of Decision Tree Regressor is: 0.09833145052589419
```

Here we can see that test score and train score has gone down when compared to the normal decision tree without hyperparameter. But we can consider as more generalized model as the train and test are almost similar, whereas for other model the train and test score has some variance which may lead to overfitting.

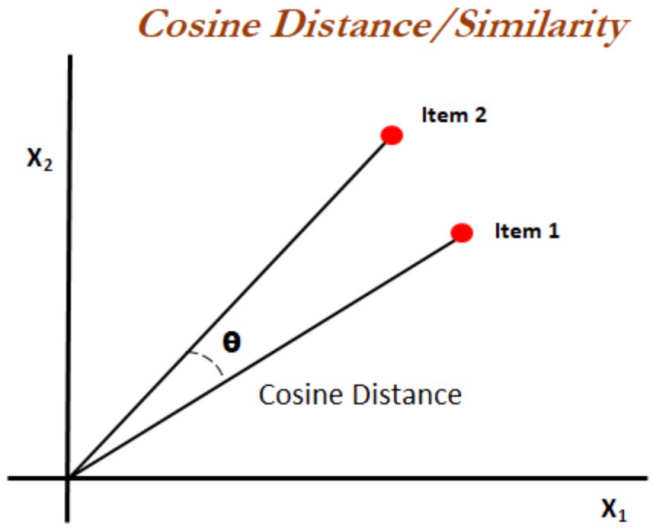
Metrics

Result Parameters	Linear Regression	Decision Tree	Random Forest	Decision Tree with GridSearch CV
R2 Score	0.9307	0.9237	0.9611	0.9021
Mean Squared Error	0.0695	0.0765	0.0390	0.098

Here we can see that Random Forest gave us the best R2 score and least Mean squared error. So we can consider it as our ideal model.

Scope of Recommendation System

- We can build a recommendation system where it recommends similar songs for any given song.
- Here we can use Collaborative Filtering or cosine similarity in order to implement this.
- Here we will focus on Cosine similarity a bit. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. It determines how two items are similar to each other on a scale of 0 to 1.
- The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance, chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.



Thank You