SPOTIFY POPULARITY ANALYSIS

ABOUT

Spotify is a digital music, podcast, and video service that gives you access to millions of songs and other content from creators all over the world. Basic functions such as playing music are totally free, but you can also choose to upgrade to Spotify Premium. Spotify is available across a range of devices, including computers, phones, tablets, speakers, TVs, and cars, and you can easily transition from one to another with Spotify Connect.
In this project we are going to use a variety of different regression models to predict the popularity of songs on Spotify.

DATASET

The dataset is stored in tabular format in csv file.
These are the following columns in the dataset:

1. Number
2. Track ID
3. Artists
4. Album Name
5. Track Name
6. Popularity
7. Duration
8. Explicit
9. Danceability
10. Energy

XGBOOST

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.
In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

IMPLEMENTATION

Steps:

1. Loading necessary packages and the dataset

2. Getting to know the dataset: We can see that we have some duplicates but since they were introduced after we removed some of our columns, they are not actual duplicates and we should keep them in the dataset.

```
number of duplicate rows: 0
number of null values:
Unnamed: 0          0
track_id            0
artists             1
album_name          1
track_name          1
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64
```

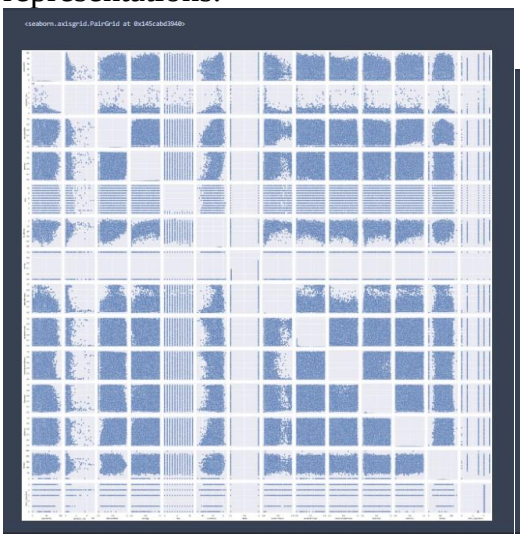
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 16 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   popularity        114000 non-null  int64
 1   duration_ms       114000 non-null  int64
 2   explicit          114000 non-null  bool
 3   danceability      114000 non-null  float64
 4   energy            114000 non-null  float64
 5   key               114000 non-null  int64
 6   loudness          114000 non-null  float64
 7   mode              114000 non-null  int64
 8   speechiness       114000 non-null  float64
 9   acousticness      114000 non-null  float64
 10  instrumentalness  114000 non-null  float64
 11  liveness          114000 non-null  float64
 12  valence           114000 non-null  float64
 13  tempo             114000 non-null  float64
 14  time_signature    114000 non-null  int64
 15  track_genre       114000 non-null  object
dtypes: bool(1), float64(9), int64(5), object(1)
memory usage: 13.2+ MB
```

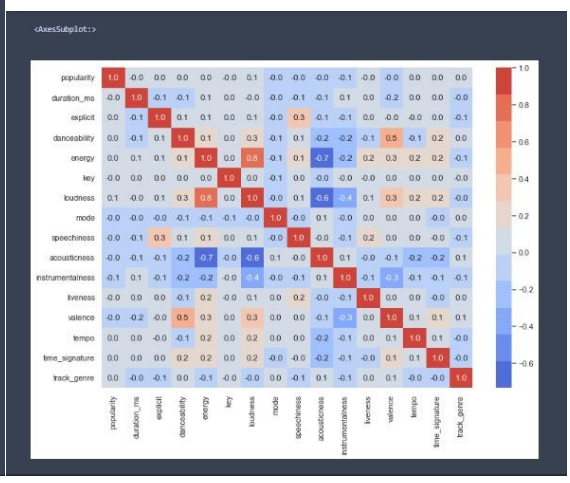
```
number of duplicate rows: 7093
number of null values:
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64
```

3. Pre-processing: First, we take care of the categorical variables in our dataset and turn them into numeric variables.

	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness
0	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430	0.0322	0.000001
1	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763	0.9240	0.000006
2	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557	0.2100	0.000000
3	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363	0.9050	0.000071
4	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526	0.4690	0.000000

4. Exploratory Data Analysis (EDA): It is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

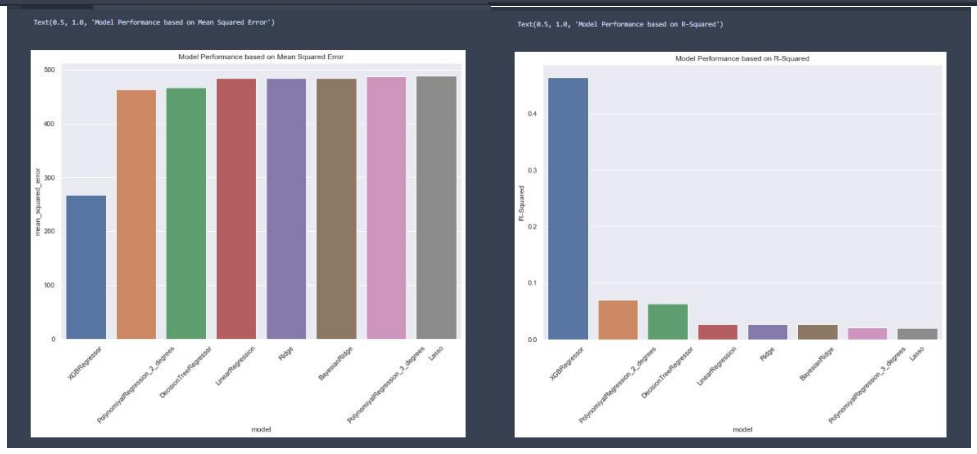




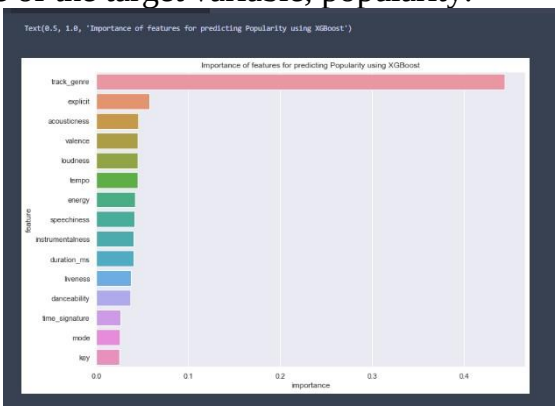
5. Modelling and Model Selection: In the modelling part we should split our dataset into a training and a test dataset.

6. Model Comparison: We can see that the XGBoost model has performed much better than the others. Let us visualize their performance by different metrics.

	model	mean_squared_error	R-Squared	time
0	XGBRegressor	267.25734	0.46335	43
0	PolynomiyalRegression_2_degrees	462.98610	0.07033	1
3	DecisionTreeRegressor	466.44516	0.06339	1
5	LinearRegression	484.55812	0.02702	0
1	Ridge	484.55830	0.02702	0
4	BayesianRidge	484.56537	0.02700	0
1	PolynomiyalRegression_3_degrees	487.34004	0.02143	7
2	Lasso	488.06187	0.01998	0



7. Feature Importance: Here we want to see which features in our dataset explain most of the variance of the target variable, popularity.



CONCLUSION

We could see that even our best model, the XgBoost regression model, has a relatively low R-squared value so there are a couple of things we could try to improve that.

- Using the removed features like artist name, album name etc. (We have to deal with non-numeric features for this)
- Using a Cross-Validation set and try to only select the features that are more important or create new ones based on the performance of our model on the validation set and whether it has high bias(underfit) or high variance(overfit)