**Hit or Miss:  
Predicting if the song will be a Hit using Spotify audio features and machine learning**

**Author: Adesh Gadge**

**Introduction**

**Data:   
(1)** **Training Data:** It was downloaded from [Kaggle](https://www.kaggle.com/tomigelo/spotify-audio-features). [Spotify for Developers](https://developer.spotify.com/) offers a wide range of possibilities to utilize the extensive catalog of Spotify data. One of them are the audio features calculated for each song and made available via the [official Spotify Web API](https://developer.spotify.com/documentation/web-api/). Each song (row) has values for artist name, track name, track id and the audio features itself (for more information about the audio features check out [this doc from Spotify](https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/)). Additionally, there is also a popularity feature included in this dataset. The size of dataset is around 130K songs and their audio features.

**(2) Data to check generalization performance:** It was downloaded using Spotify API for python ([Spotipy](https://spotipy.readthedocs.io/en/latest/)). Spotify API was used for two purposes: (1) To extract 10,000 popular songs on the given day (2) To extract Spotify audio features for those extracted 10,000 songs.

**Data Description:**The Audio features include:

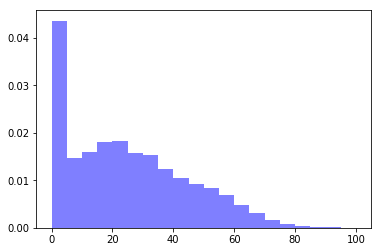
|  |  |  |
| --- | --- | --- |
| Feature | Data Type | Description |
| Danceability | float | Describes how suitable a track is for dancing. This value is based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is the least danceable and a value of 1.0 is the most danceable |
| Energy | float | A value representing a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud and noisy. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. Energy is described as a value between 0.0 and 1.0. |
| Key | int | The key the track is in. Integers map to pitches using standard Pitch Class notation. (https://en.wikipedia.org/wiki/Pitch\_class) |
| Loudness | float | The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). This value usually ranges between -60 and 0 dB. |
| Mode | int | Describes the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by the value 1 and minor is represented by the value 0. |
| Speechiness | float | Detects the presence of spoken words in a track. The more exclusively speech like the recording is, the closer the value is to 1.0. If the speechiness ranges between 0.66 and 1.0, the track is probably made entirely of spoken words (such as audio books, poetry etc.). Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered. Values below 0.33 most likely represent music and other non-speech-like tracks. |
| Acousticness | float | A confidence measure between 0.0 and 1.0 of how acoustic a track is. |
| Instrumentation | float | Predicts whether a track contains no vocals. Sounds like ”Ooh” and ”Aah” are treated as instrumental in this context, while rap or spoken words are clearly ”vocal”. The attribute ranges between 0.0 and 1.0 and the closer to 1.0 the value is; the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks. |
| Liveness | float | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track was performed live. |
| Valence | float | Describes the musical positiveness conveyed by a track. The value ranges between 0.0 and 1.0. Tracks with high valence sound more positive (happy, cheerful etc.), while tracks with low valence sound more negative (sad, depressed etc.) |
| Tempo | float | The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration. |
| Duration | int | The duration of a track in milliseconds |
| Time signature | int | An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). |

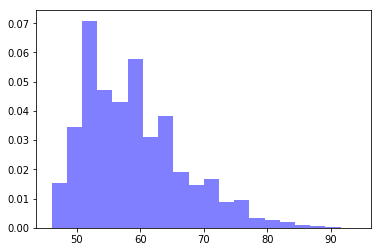
**Target Variable:**

Popularity variable ranges from 0 to 100, target variable was derived from popularity, greater than or equal to 70 as Ispopular =1 and for less than 70 Ispopular= 0

Hence the problem solved in this project is a binary classification problem.

1. Training data popularity distribution:



1. Generalization Performance data popularity distribution:   
   

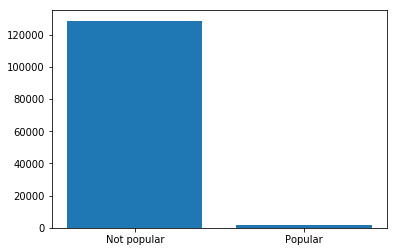
**Data Pre-processing:  
(1) Categorical Variable:** The data had 2 categorical variables: Key, Time Signature.   
**OneHotEncoding** was performed on these two variables.

**(2) Numerical Variable: Standardisation** was performed so that every variable is scaled appropriately.

**(3) Training data was split into train and test to assess the model performance.**

**(4) Training data was highly imbalanced:**

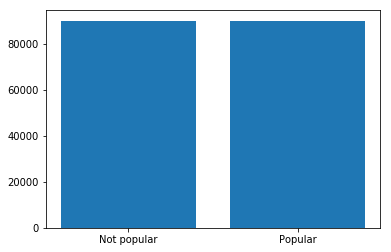
Distribution before SMOTE (entire data):

****

**Not Popular= 128643 Popular =2020**

Up-sampled the popular class using the [SMOTE algorithm](https://arxiv.org/pdf/1106.1813.pdf)(Synthetic Minority oversampling Technique), At a high level, SMOTE:

1. Works by creating synthetic samples from the minor class (no-subscription) instead of creating copies.
2. Randomly choosing one of the k-nearest-neighbours and using it to create a similar, but randomly tweaked, new observations.

Distribution after SMOTE (after train/test split):  


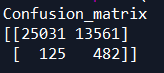
**Not Popular= 90051 Popular =90051**

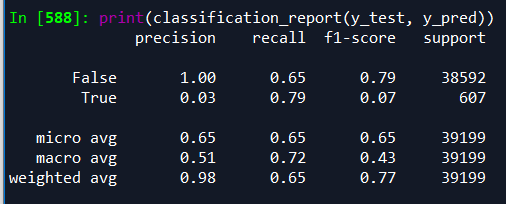
**Model Building:**

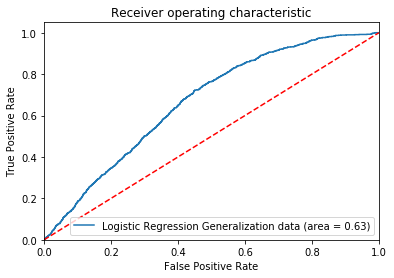
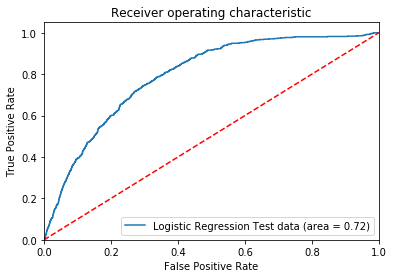
1. **Base Model: Logistic Regression**

Model with no parameter tuning gave various P-values, eliminated variable with p-value >0.05, Key\_6 dummy variable was eliminated in this process and again another logistic regression model was built without key\_6

**Training Accuracy: 0.76, Testing Accuracy: 0.65, Generalization Accuracy: 0.49**

**Confusion Matrix for testing: **

**Classification report: **

**ROC Curve: **

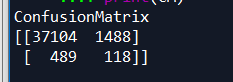
**For testing data For Generalization Data**

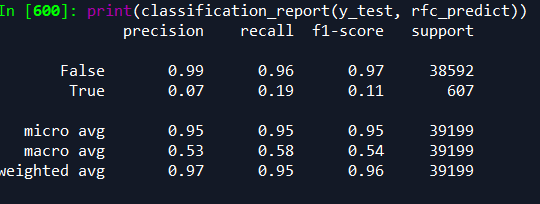
**(AUC: 0.72) (AUC:0.63)**

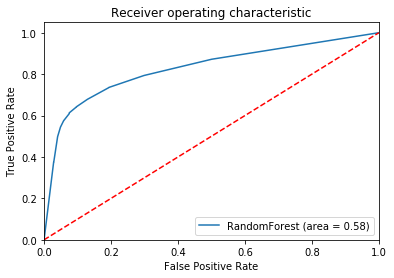
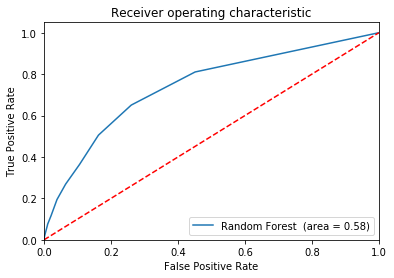
1. **Random Forest Model:**

For model with default parameters,

**Training Accuracy: 1.00, Testing Accuracy: 0.95, Generalization Accuracy: 0.89**

**Confusion Matrix for testing:**

**Classification report: **

**ROC Curve: **

**For testing data For Generalization Data**

**(AUC: 0.58) (AUC:0.58)**

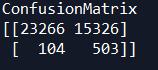
1. **Random Forest with parameter tuning with GridSearch 5-step Cross validation:**

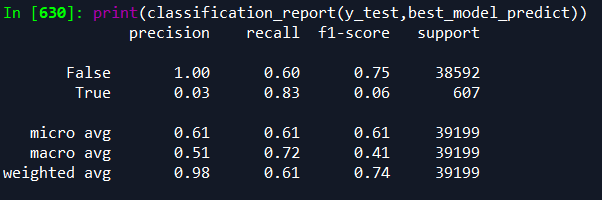
Parameters were tuned using GridSearchCV for random forest, parameters such as n\_estimators, Criterion, bootstrap, min\_sample\_split

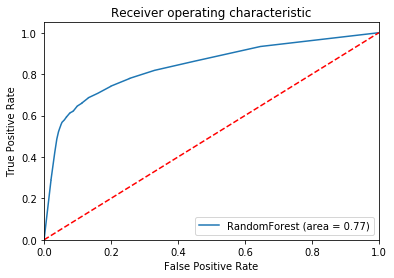
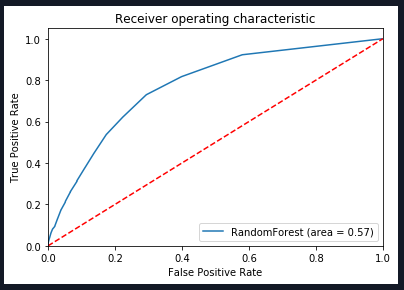
**Best parameters:** n\_estimators=10, Criterion= gini, bootstrap=True , min\_sample\_split=2

The model saw improved ROC in generalization error, because of improved recall

**Training Accuracy:1.00, testing Accuracy:0.95, Generalization accuracy: 0.89**

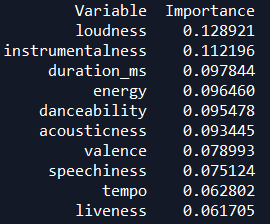
**Confusion Matrix: **

**Classification report: **

**ROC: **

**For testing data For Generalization Data**

**(AUC: 0.57) (AUC:0.77)**

**Feature Importance:** 

**Conclusion:**

The tuned random forest model saw higher accuracy and higher AUC value when compared to the baseline logistic regression model. We saw according to feature importance, loudness, instrumentalness, duration\_ms, energy were the important features according to random forest classifier.

**References**

1. <https://spotifycharts.com/regional>

1. <https://spotipy.readthedocs.io/en/latest/>
2. <https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-audio-features/>
3. <https://www.kaggle.com/tomigelo/spotify-audio-features>
4. <https://github.com/plamere/spotipy>
5. <https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8>