Track Popularity Predictor

By: Aanvi Goel

ABSTRACT

The goal of this project was to predict the Track Popularity based on Audio Features defined by Spotify Web API. I used Spotify Web API to build the dataset with track names, audio features and track popularity. After cleaning the data and performing EDA, the column "track_pop" is converted from numerical to categorical target. I built multiple classification models using Logistic Regression and Decision Tree based Ensemble algorithms. After comparing metrics for each model, I finalized on a Random Forest Classifier with Hyperparameter tuning.

DESIGN

Spotify is a Swedish audio streaming company that has taken over globally, with 33 million monthly active users, including 188 million paying subscribers, as of June 2022. The company offers a developer-friendly Web API with easy access to streaming data metrics. This project is focused on the Audio Features APIs available for getting audio metrics such as Danceability, Energy, Speechiness etc for every track.

The popularity of a track has a range between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. A classification model to classify a track as Popular or Not based on audio features would help Spotify as well as record companies analyze whether a track has the potential to be popular before its launch.

DATA

1) **Spotify Audio Features dataset:**
   https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs

The first dataset is in .csv format. The dataset includes 1.2M+ songs along with respective audio features such as tempo, energy etc for each.

2) **Spotify Web API:**
   https://developer.spotify.com/documentation/web-api/reference/#/operations/get-track

Spotify's Web API will be used to fetch the track popularity for each track in the first dataset to train and validate the classification model.

ALGORITHMS

Feature Engineering

- Numerical column track_pop is converted to Categorical target using a 70th percentile cut-off

Models

- Logistic Regression
- Decision Trees
- Random Forest
- XGBoost

Model Optimization

- Logistic Regression is optimized by selecting more optimum value of “C” using GridSearchCV
- Class Imbalance is handled by using oversampling, adjusting class weights and tuning the probability thresholds
- Hyperparameters are tuned for RandomForest Classifier using RandomizedSearchCV
- Hyperparameters are tuned for XGBoost using GridSearchCV

TOOLS

- *Data Acquisition*: Spotipy (Python library for Spotify Web API)
- *Data Cleaning and Analysis*: Pandas, Numpy
- *Data Visualization*: Matplotlib, ConfusionMatrixDisplay
- *Data Modeling: SkLearn, XGBoost, Imblearn*

COMMUNICATION

- A slidedeck and Jupyter notebook code are included along with this write-up as part of the project on Github