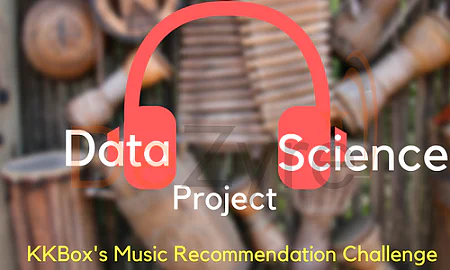
Music Recommendation System

Introduction:

*“****Music is the pleasure the human mind experiences from counting without being aware that it is counting****”*

Music helps anybody to connect with what you are doing. It elevates mood and rejuvenates the waves of thoughts. Different people have different flavors of music. Music has served its users with various platforms like waves of Victrola, a culture of Cassette, Walkman era, i-pods, FM-Radios, and now latest musical apps like Spotify, Amazon Prime Music, Deezer, SoundCloud, Gaana, etc.

The Internet made life easy in terms of selecting music of users’ choice, but still, algorithms are needed to recommend favorite music to users without selecting it manually.



Data Source:

<https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data>

Tech Stack (Technical Tool):

*Language:* Python

*Libraries:* Sklearn, Xgboost, Pandas, NumPy

Overview of Music Recommendation System Project using Machine Learning:

The 11th ACM International Conference on Web Search and Data Mining (WSDM 2018) challenged us to build a better music recommendation system using a donated dataset from KKBOX. WSDM (pronounced “wisdom”) is one of the premier conferences on web-inspired research involving search and data mining.

The input contains text data only, and no audio features.

This Music Recommendation System will walk us through some Machine Learning techniques that one can apply to recommend songs to users based on their listening patterns. To predict the chance of a user listening to a piece of music repetitively after the first observable listening event within a particular time.

Dataset:

The dataset is from KKBOX, Asia’s leading music streaming service, holding the world’s most comprehensive Asia-Pop music library with over 30 million tracks. There are three datasets available.

*train.csv:* It contains data for different users with attributes such as msno, song\_id, source\_system\_tab, etc. There are about 10.4 million entries available.

*songs.csv: I*t contains the data related to songs with attributes such as song\_id, song\_length, genre\_ids, artist\_name, etc.

*members.csv:* The data is related to users’ information over 34403 different users.

Approach:

Exploratory Data Analysis (EDA):

The music recommender system project dataset has about 3 million rows, and such large-scale data can be easily analyzed using Pandas dataframes in Python. The analysis involves app-user behaviours and, more precisely, what makes a user listen to songs again and again. We will achieve this by plotting insightful plots using Python libraries, matplotlib, and seaborn. Here, we will be using the first 10k rows only.

1. Data Visualization
2. Inference in features
3. Feature engineering

**Questions:**

* How many unique user\_id (msno) are available in the dataset?
* How many unique songs id are available in the dataset?
* From where did the Music source has occurred?
* Which arrival point do users use most in a mobile application? (Likewise local playlists/online playlists/albums/exploring or my libraries?)
* How many unique source types are available for the music?
* How many times do users listen to the same songs in a month(target=1), and which songs users don’t listen to again from the song\_id? (This quest is based on the “target” column)
* From which city users are highly using the musical application in the dataset?
* Which age group is more interested to listen to music?
* Which gender is used to like listing the song more?
* Which registration method has been used most by Users?
* In which year users have registered more for the music application?
* Who is the most favorable artist in the dataset?
* Which composer composes most of the songs in the dataset?
* Which language is more heard by users in music?
* What is the minimum/maximum song length in the dataset?
* How many unique genre\_ids are available in the dataset?
* Who is the leading lyricist in the dataset?

Data Cleaning (Outlier/missing values/categorical):

* Outlier detection and treatment
* Imputing missing values

1. Replacing by mode
2. Removing null values
3. Making a new label as missing

* Converting labeled or string values by numerical values

Model Building on training data:

1. Logistic regression
2. Decision Tree
3. Random Forest
4. XGBoost

Model Validation

* Roc\_Auc

Feature importance and conclusions

**Solution**

Conclusion of Exploratory Data Analysis (EDA):

Count- plot for “source\_system\_tab”:

Chart, bar chart

Description automatically generated

* Users listen to more songs from my library which are stored in their library.
* Songstoredes in the library is the one that users tend to listen to again in a given time frame, other than that all other sources are less likely to be used.

Count- plot for “source\_screen\_name”:

Chart, bar chart

Description automatically generated

* Local playlist is most visited screen for users to listen to music.

Count- plot for “target”:

Chart, bar chart

Description automatically generated

* Both targets have almost the same size in the dataset.

Chart, bar chart

Description automatically generatedCount- plot for “city”:

* People belonging from City-1 are the ones who use the app most.

Count- plot for “gender”:

* The number of males are more in the dataset and they are more likely to listen to the same song again than females.

Chart, bar chart, treemap chart

Description automatically generatedDistribution plot for “Registration Date”:

Chart, histogram

Description automatically generated

* more registration has been done from 2012 to 2016.

plot for “age”:

here, we have used a distribution plot. And after that, we were checking with outliers through Z-Score.

Chart, histogram

Description automatically generated

Mean of Age: 17.539271192170485

Mode of Age: 0

Standard Deviation of Age: 21.554468765781245

Total Outlier in the dataset are: 6953

Maximum Age Outlier: 1051

Minimum Age Outlier: 83

* According to Z-Score, there are total 6953 outliers in the dataset.
* Age between 83 to 1051 are the outliers and can be removed.

Data Cleaning (Outlier/missing values/categorical):

Here, we have used three different methods to handle the missing values such as, ‘replacing by Mode’, ‘removed Null Values’, and ‘making new missing label’.

Machine Learning Model Implementation:

1. **Logistic Regression.**
   1. Replacing by Mode:
      1. Accuracy: 0.5535,
      2. Precision: 0.5532,
      3. Recall: 0.7430
   2. removed Null Values:
      1. Accuracy: 0.52955,
      2. Precision: 0.6049,
      3. Recall: 0.3830
   3. making new missing label:
      1. Accuracy: 0.5601,
      2. Precision: 0.5647,
      3. Recall: 0.6781
2. **Decision Tree Classifier.**
   1. Replacing by Mode:
      1. Accuracy: 0.5411,
      2. Precision: 0.5635,
      3. Recall: 0.5275
   2. removed Null Values:
      1. Accuracy: 0.5225,
      2. Precision: 0.5793,
      3. Recall: 0.4379
   3. making new missing label:
      1. Accuracy: 0.5741,
      2. Precision: 0.5907,
      3. Recall: 0.5939
3. **Random Forest Classifier.**
   1. Replacing by Mode:
      1. Accuracy: 0.6452,
      2. Precision: 0.6639,
      3. Recall: 0.6458
   2. removed Null Values:
      1. Accuracy: 0.6056,
      2. Precision: 0.6976,
      3. Recall: 0.4821
   3. making new missing label:
      1. Accuracy: 0.6541,
      2. Precision: 0.6558,
      3. Recall: 0.7072
4. **XGBoost (n\_estimators = 2).**
   1. Replacing by Mode:
      1. Accuracy: 0.6380,
      2. Precision: 0.6464,
      3. Recall: 0.6733
   2. Removed Null Values:
      1. Accuracy: 0.6087,
      2. Precision: 0.6554,
      3. Recall: 0.5878
   3. Making new Missing label:
      1. Accuracy: 0.639468,
      2. Precision: 0.6443,
      3. Recall: 0.6872

**XGBoost Roc-curve with Confusion-Metrics:**

**Replacing by Mode:**

**Chart, treemap chart

Description automatically generated**

**Removed Null Values:**

Chart

Description automatically generated

**Making new Missing label:**

Chart

Description automatically generated

Closer model Implementation from my point of view:

XGBoost is a scalable end-to-end tree boosting system, its sparsity-aware split finding algorithm helps a lot with the sparse dataset we have. The main idea is to only collect statistics of non-missing entries and classifies missing value into default direction. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

Chart, bar chart, histogram

Description automatically generatedFeature importance and conclusions:

* Looking at the feature importance the user who is listning the song has the highest weightage.
* It's possible that on Registration day and Service Expiration day people use the app more and thus listen to songs on repeat.
* Age(bd) is always a factor as the dataset has a mean age of 17 year and youngsters do listen to more songs.
* Composer and Artist are also important features.

Future Work & Scope of Improvement:

I have used 10k rows for songs.csv due to limitation of resources, for members.csv and train.csv dataset I have used “amazonaws” as resources are available for me. Using all the data will surely lead to better performance of models. More aggressive data pre-processing & feature engineering along with better selection might result in better results. Although I have done hyperparameter tuning to a decent extent, we can surely dive into it much deeper & hunt for better results.