SPRINGBOARD DATA SCIENCE COURSE

CAPSTONE PROJECT 2

Building a music recommendation system using machine learning techniques

Abhishek Sukhadia

Sep, 2020

Contents

1. Introduction

In the earlier days, we used to buy products based on recommendations from our friends and family and the products they trust. Before the digital age, that has been the default method of purchase. But now with advent of digital age, this is changing as the circles have expanded to more than just friends and family. Companies like Amazon, Flipkart, Spotify and Netflix have developed recommendation systems that would recommend product to their customers. These systems filters data using different algorithms by capturing past behaviour of a customer and then recommends the most relevant items to the users. This project's focus is to build a similar recommendation system for music industry.

With mobile devices and the internet coming in, offline music CDs, music players have gone extinct. Online streaming has become the new norm for today's generation. This has led to music going online and customers getting access to millions of songs. The number of songs available exceeds the listening capacity of a single individual. People at times feel it is difficult to choose his favourite song from millions of songs. Thus customer convenience, efficient management of songs/playlists and personalized music discovery have become the need of the hour. This has given rise to companies bringing in music recommendation systems to improve customer experience.

Lots of platforms like Spotify, Saavn, Apple Music, etc. are gathering lots of data based on users' listening habits, song info and user info. These huge amounts of data paved the way for machine learning algorithms in building a good recommendation system. Lots of research is also been done on music recommendation system.

2. Data Acquisition and Cleaning

I acquired the data from Kaggle which is freely available. The dataset is part of a competition (WSDM 2018) where the challenge is to build a better music recommendation system using a donated dataset from KKBOX. The data set consists of information of the first observable listening event for each unique user-song pair within a specific time duration. Metadata of each unique user and song pair is also provided. The train and the test data are selected from users' listening history in a given time period and split based on time period. The dataset has over 50lakhs rows and more than 30 features.

Each row is a single encounter of a customer's listening event with details given below.

Customer/Members Characteristics:

- Membership No, City, Gender, Registered via, Age, Registration and expiration date for membership

Song Details:

- Song id, Song length, Genre Ids, Artist Name, Composer, Lyricist, Language

Extra Song Info:

- Song Name, ISRC code

Song Listening Characteristics:

- Source System Tab, Source Screen Name, Source Type

Target:

- Whether there are recurring listening event(s) triggered within a month after the user's very first observable listening event

The above data was downloaded as a CSV and then imported in python. More details of this can be found in this **IPython Notebook**. The training data had 7377418 rows while testing data had 2295971 rows. Post importing, data was pre-processed using the below mentioned steps.

1) Merging and Transformation: The different features like song and user characteristics are present in different tables. Thus there was a need for merging into one table using the columns song_id and membership no (msno) as key ids. Songs, members and songs_extra tables were merged with the training and testing data.
   Registration and Expiration date was imported as object class. So, we transformed these columns to "datetime" type so working with it becomes easy.
2) Missing Value Analysis: Table 1 is the summary of the missing values. Lot of important features have missing values.

	feature_name	null_values	%missing	category_count
0	lyricist	3178798	43.09	33889
1	gender	2961479	40.14	3
2	composer	1675706	22.71	76065
3	isrc	577858	7.83	269761
4	source_screen_name	414804	5.62	21
5	genre_ids	118455	1.61	573
6	source_system_tab	24849	0.34	9
7	source_type	21539	0.29	13
8	name	1457	0.02	234145
9	song_length	114	0.00	60267
10	artist_name	114	0.00	40583
11	language	150	0.00	11

Table 1: Features with missing values

Testing data set had similar missing values but more in volume. (Table 2)

	feature_name	null_values	%missing	category_count
0	lyricist	1224744	47.90	24911
1	gender	1052224	41.15	3
2	composer	619304	24.22	52307
3	isrc	196643	7.69	170988
4	source_screen_name	162883	6.37	23
5	genre_ids	42110	1.65	502
6	source_system_tab	8442	0.33	9
7	source_type	7297	0.29	13
8	name	778	0.03	154717
9	song_length	25	0.00	45658
10	artist_name	25	0.00	27564
11	language	42	0.00	11

Table 2: Missing Values in Testing Data

Missing values were handled differently for each feature.

- **lyricist, gender and composer** have >20% missing values. These columns are important features so dropping these columns is not an option. We cannot also replace the missing values with mode. So, we created separate categories as "missing" for these features to tackle this.
- **isrc, source_screen_name, source_type, genre_ids, source_system_tab, name and artist name** have 0-10% missing values. Most of the columns being important and categorical in nature, missing values were considered as a separate category and rows were not dropped.
- **Language** has some values as -1. This value is assumed to be for missing values or where language is not known. So, missing values of language were replaced by -1.
- Missing values of **song_length** were replaced by median song length

Another interesting thing observed was the the missing values between few variables were correlated. Eg. 90% correlation between missing values of artist name and language were observed. We can infer that artist name was determining the language of the song. Source type and Source_system tab, composer and lyricist were other correlations that were found (Figure 1). Though there was correlation, this was not used to fill the missing values.

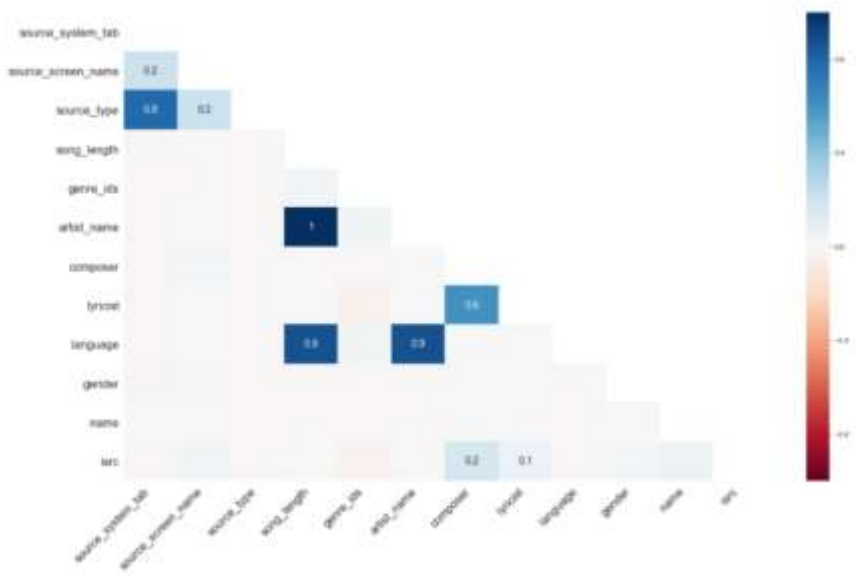


Figure 1: Correlation between missing values of different features

3. Feature Creation and Outlier Analysis

3.1 Feature Creation

We had a limited set of features and most of them are categorical variables like song name, artist name etc. These makes the problem complex and hard for algorithms to learn. So, to provide more information, we manually extracted many features from existing variables. They are listed as below.

Features Based on Songs Features:

Sr. No.	Feature Notation	Feature Description
1	lyricist_count	Number of Lyricist for a song
2	artist_count	Number of Artists for a song
3	composer_count	Number of Composers for a song
4	genre_id	Number of genres for a song
5	count_song_played	Number of times the song was played
6	song_freq	Cumulative count of the song played over time
7	Count_artist_played	Number of times an artist was played
8	artist_freq	Cumulative count of the artist played over time
9	composer_artist_lyricist	Whether composer, artist, lyricist were same or not
10	composer_artist	Whether composer and artist were same or not

Features based on Time:

Sr. No.	Feature Notation	Feature Description
1	duration	Membership duration i.e. difference between registration and expiration date
2	registration_year	Year of membership registration
3	registration_month	Month of membership registration
4	registration_day	Day of membership registration
5	expiration_year	Year of membership expiration
6	expiration_month	Month of membership expiration
7	expiration_day	Day of membership expiration

Other features:

Sr. No.	Feature Notation	Feature Description
1	age_of_song	Created from isrc year. Difference between 2017 and the isrc year. Recency of the song being registered with isrc
2	song_type	Whether the song is long (>6 mins) or short (<3 mins) or medium (3 to 6 mins)

Many more features can be created but this being a huge data set and having limitation on computational power, the exercise was stopped after creating above mentioned features. More Details of the feature creation is available in this **IPython Notebook**.

3.2 Outlier Analysis

The numerical variables in the data set were age, song_length and all the count features that were extracted above. Figure 2 and Figure 3 shows the distribution of all numeric variables in the dataset.

Age shows very abnormal distribution having values more than 1000 and negative values. A peak is also observed at zero value. It is inferred that zero is missing values and other outliers are data entry problems. We then cleaned the age using the following rules.

- Making Negative values as positive values as they are data input error
- (> 1000) values are data input error so subtracting 1000 from such values
- For age between 1 to 10 and >110 considering them as missing values (0). It is assumed that oldest persons living more than 110 are rare and children with less than 10 years are not matured enough to use a music app.
- Apart from that there are lot of values with age as zero (0). These are mostly missing age values and kept as it is.

Composer count also showed some abnormal values more than 100. A song can rarely have so many composers. Upon checking that, it was found to be erroneous data and cleaned. Similar checks was done for all the count variables for the highest values and they were

cleaned. Duration also had some negative values, which were replaced by mean duration. These negative value errors were due to incorrect registration or expiration date.

Figure 4 shows the distribution of variables post outlier analysis. This clean data is now ready for exploratory analysis. **Note:** All the above cleaning replicated for test data set also.

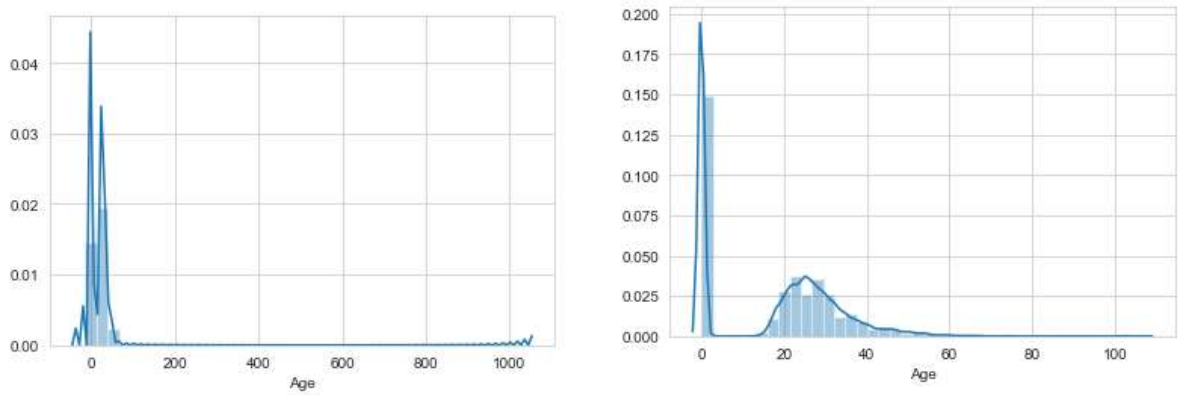


Figure 2: Original and Cleaned Distribution of Age

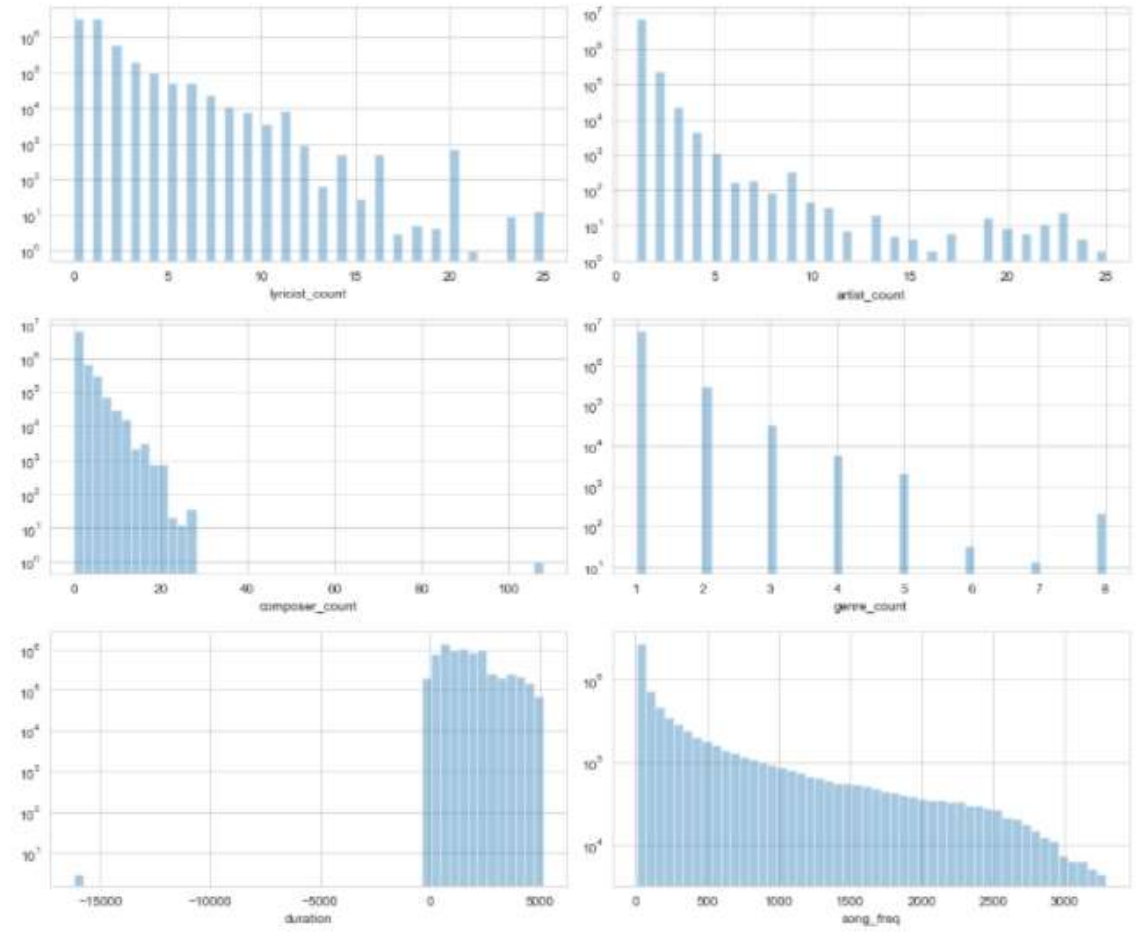


Figure 3: Pre-Outlier Analysis Distribution of Variables

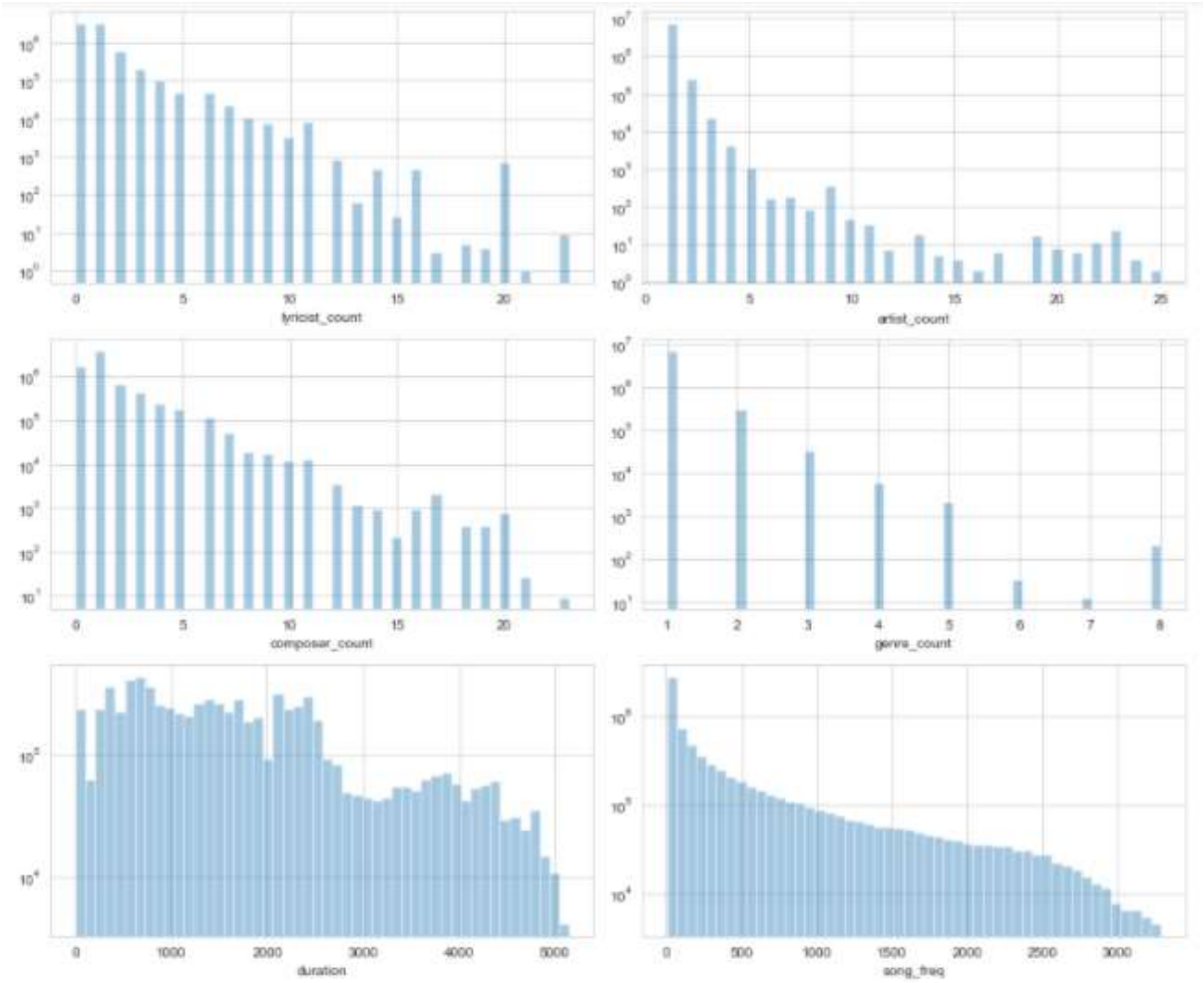


Figure 4: Post-Outlier Analysis Distribution of Variables

4. Exploratory Data Analysis

In the cleaned data, there are now 7377418 records, 36 features/columns and one target variable. We will go through most of the features in the dataset to explore the relation with the target and the listening habits of the users.

The target variable for this project contains two values:

- "0" : the song was not listened again within 30 days of the first observable event
- "1": the song was not listened again within 30 days of the first observable event

The distribution of target variable in **Figure 5** shows that our dataset is fairly balanced. Thus we will be not needed to use techniques to solve imbalances.

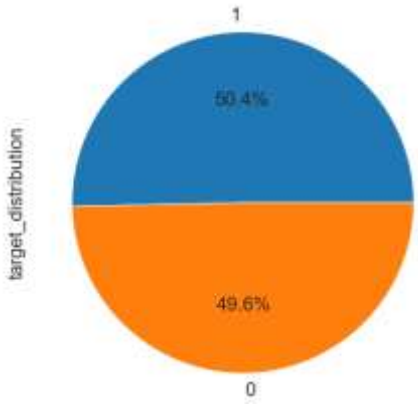
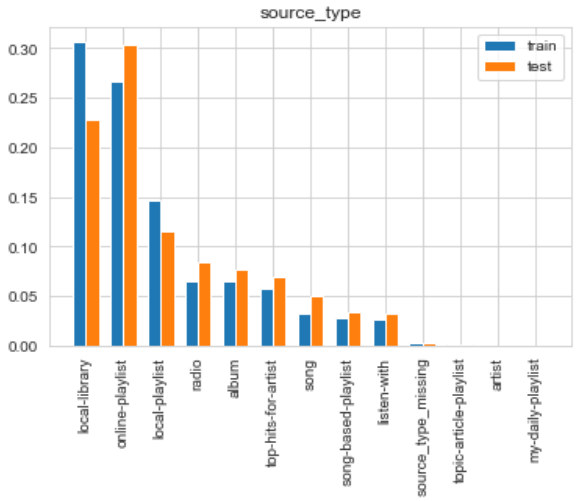


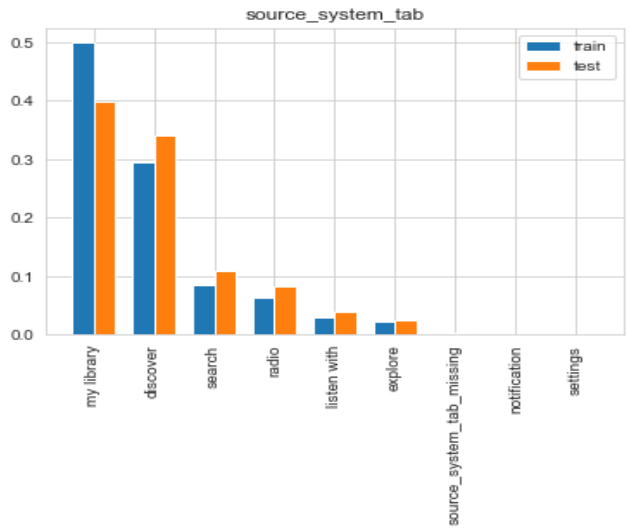
Figure 5: Distribution of Target Variable

4.1. Song Source

A user can access songs on the app/web is through various sources. They are source_type, source screen name and source system tab. Based on intuition, a user journeys through these sources to listen to his song of choice. The category names of these features also hints towards that. We observe in that for this category of features, there is change in distribution between training and testing data. Shift is from listening songs in local library to online playlist and discovering. This tells us that there is a change in listening habits over time. This tells us that our model should be robust enough to handle this changing behaviour.

Distribution of target variable with these source features tells us that users prefers to listen to their favourite songs from local playlists, local library, my library and artist (Fig)





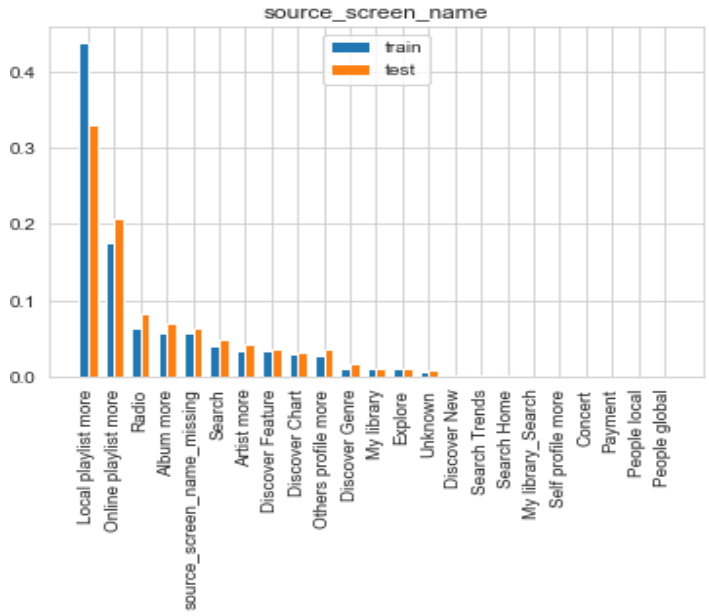


Figure 6: Distribution of song sources between test and train data

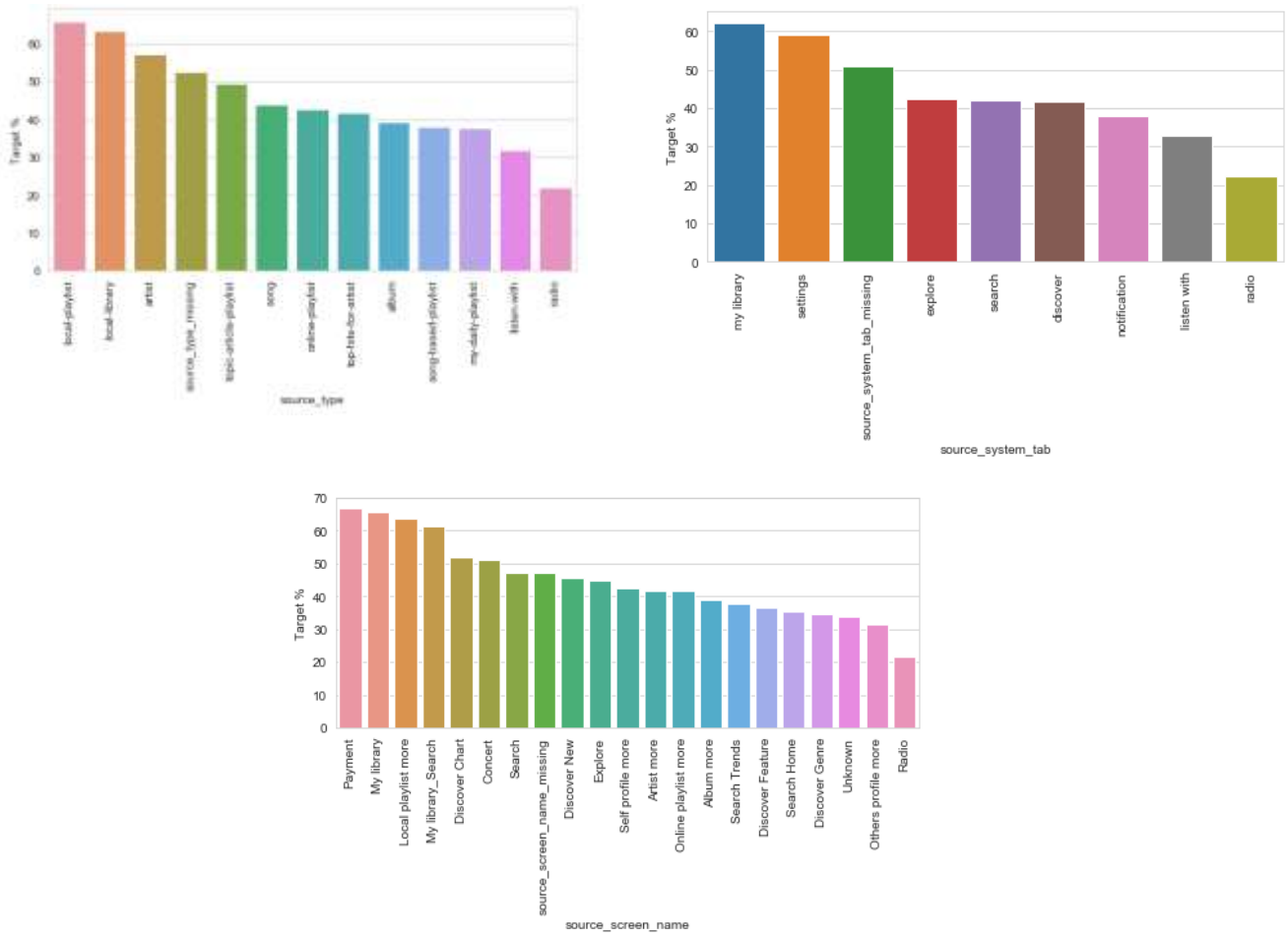


Figure 7: Influence of Song Source on repeat listening of songs

4.2. Gender

There are three categories in gender i.e. male, female and gender_missing. There is fairly equal distribution of male and female with male being 31% and female being 29% in the training set. There was no correlation found between gender and target variable, though users with missing gender prefer repeat listening less than male or female (Figure 8).

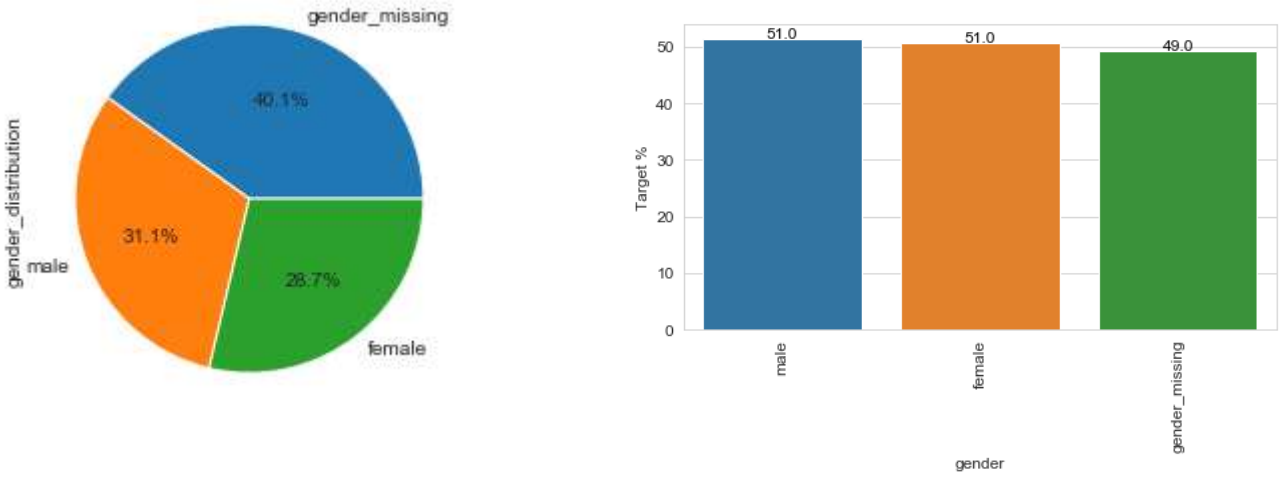


Figure 8: Gender Distribution and its influence on the repeat listening

4.3. Song Language and City

In language, 52 and 3 are the most listened to (~80%). On Comparing with cities and countries, we can infer that these two languages dominate uniformly across all cities and 53 might be English while 3 might be Taiwanese or Chinese. This is as expected because KKBox is a Taiwanese company targeting East and Southeast Asia. More details can be found in **IPython Notebook.** It seems like languages "-1", "17" and "45" lead to a lower chance of repeated listens. Language "38" does that too, but there are only 100 songs in it, which in total haven't received a lot of plays too.

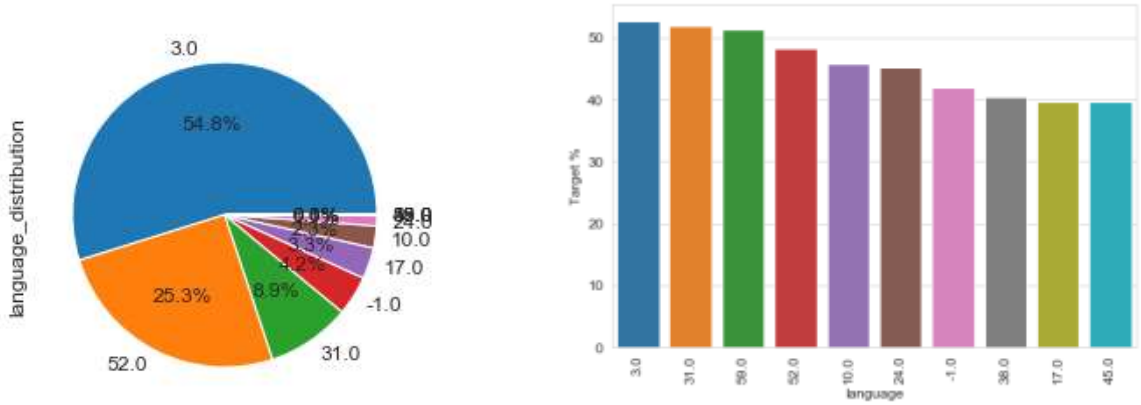


Figure 9: Language Distribution and its influence on the repeat listening

On the other side, city doesn't seems to have much influence on repeated listening. Cities like 20, 16, and 19 do affect the target but their frequency is low (Figure 10).

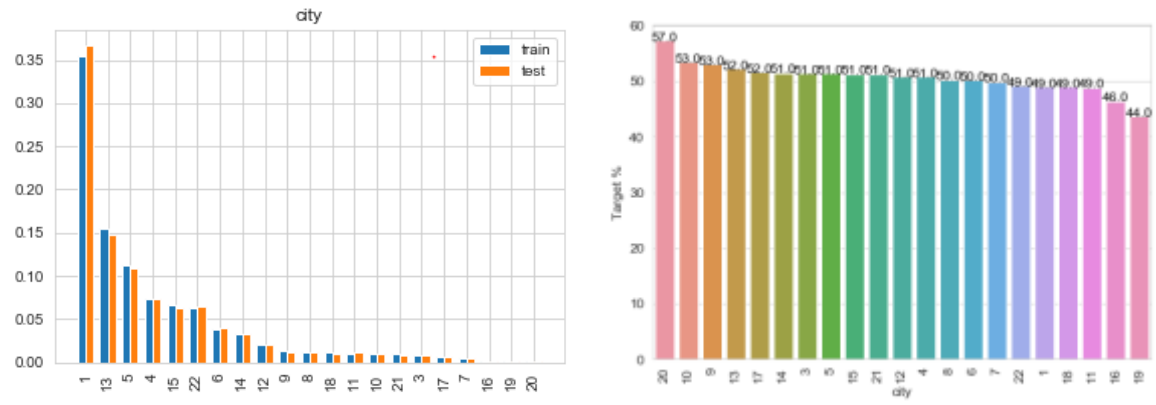


Figure 10: Language Distribution and its influence on the repeat listening

4.4. Count Features

Features analysed here were artist count, lyricist count, composer count and genre count for a song. On exploring the relationship between these variables and repeated listening, it was observed that genre count and artist count will have more influence than others. There were some exemptions for higher count values in composer and lyricist but these can be ignored due to their low frequency.

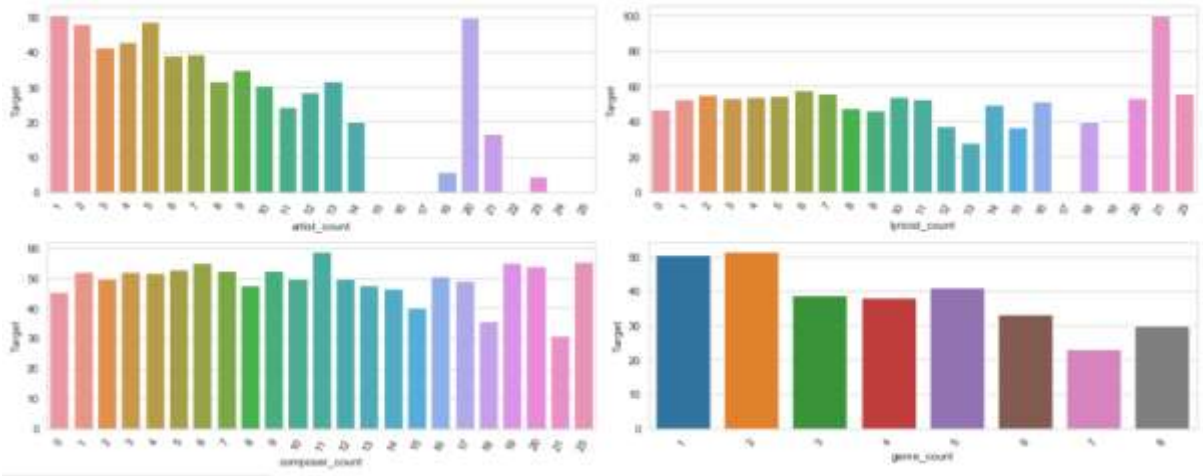


Figure 11: Count Feature and their relationship with repeated listening

We also created additional feature whether artist, composer and lyricist are same or not. It was observed that not many songs (<0.5%) had artist, composer or lyricist same. But there is some influence on the repeated listening.

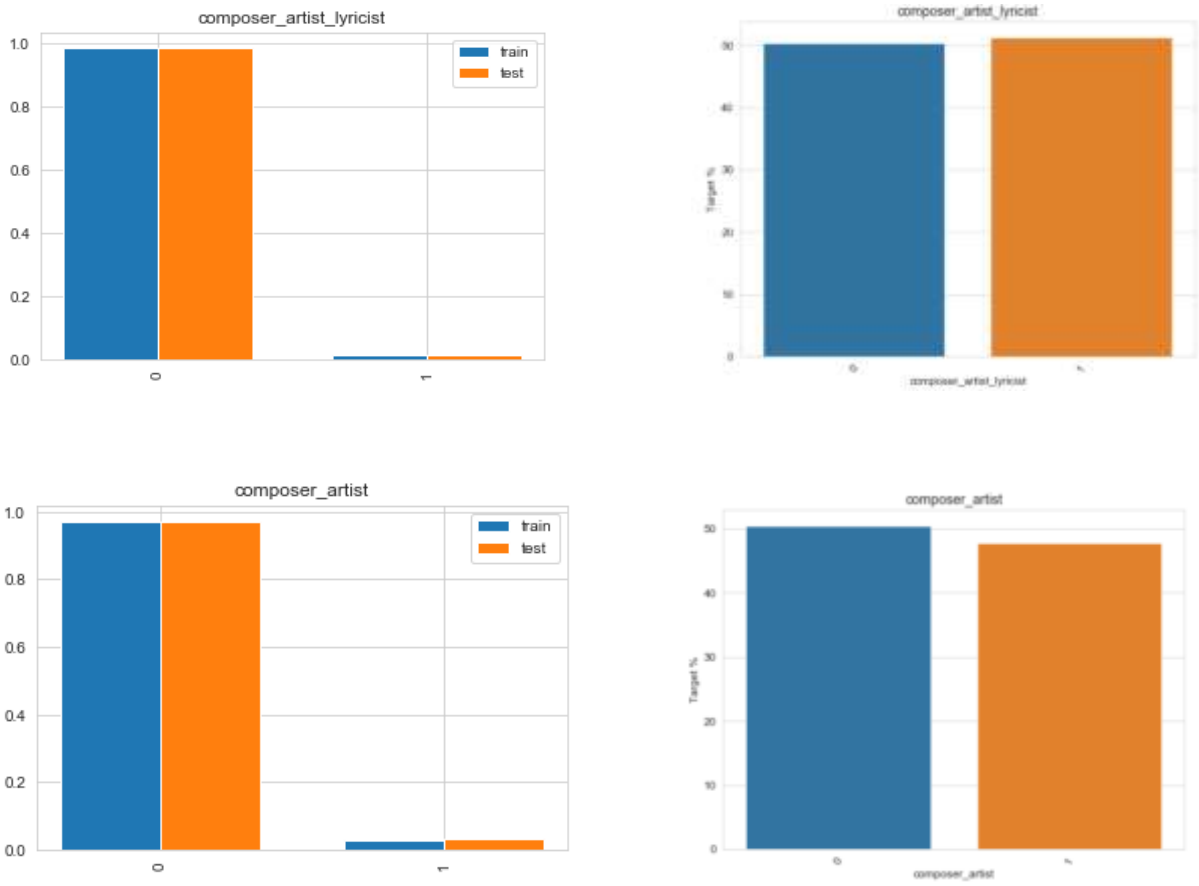


Figure 12: Composer, artist and lyricist similar feature distribution and its influence on repeated listening

4.5. Date and Time Features

We extracted day, month and year from registration and expiration time. The aim was to capture some time series and evolving customer behaviour through these features. Fig shows that the company's customer base expanded very fast from 2009 onwards. Also, almost 99% of the customers' subscription is ending in either 2017 or 2018.

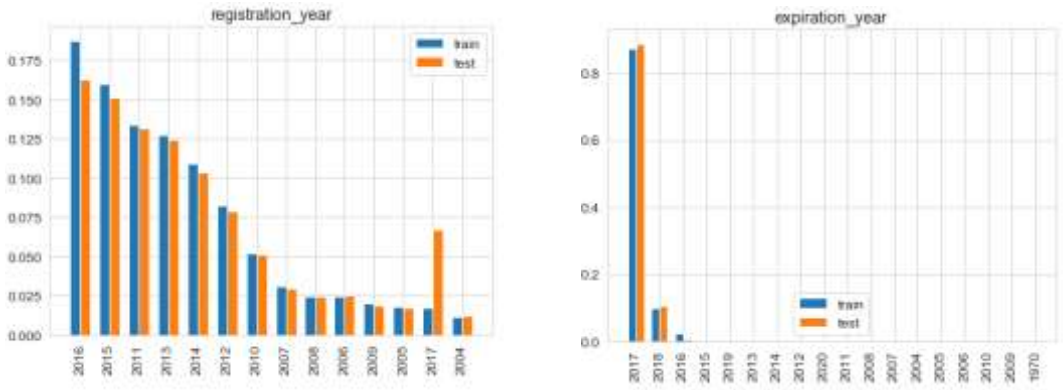


Figure 13: Distribution of date features between test and train data

With respect to repeat listening behaviour, we observe that no concrete trend was observed for features except for expiration year. Expiration year also might not have big influence because the frequency of years other than 2017 and 18 are very less as discussed above.

There were few months, years or days which has lower chance of repeated listening but nothing concrete can be said. We decided to keep the features and let model do its work on extracting information from these columns. A combination of registration and expiration can give more information. So we created a feature of membership duration.

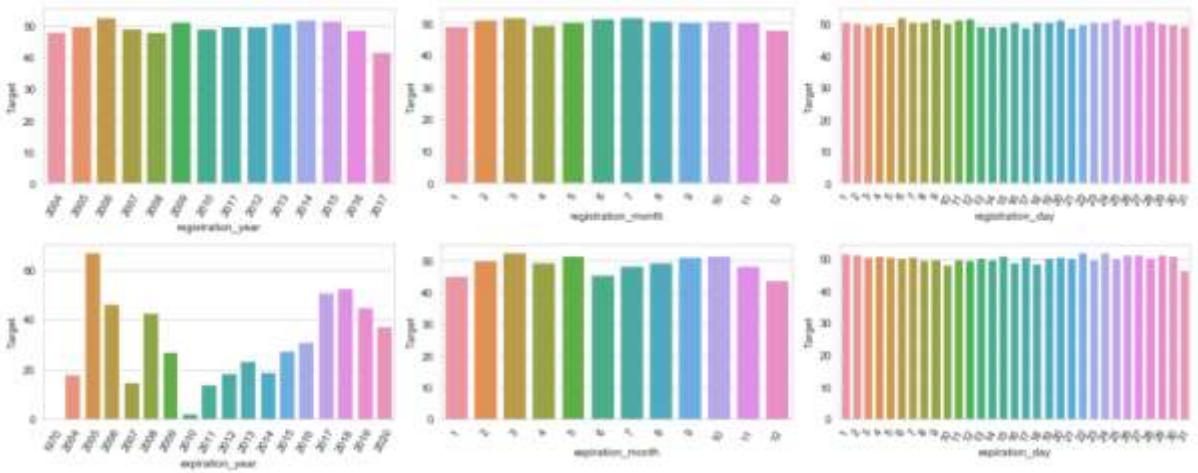


Figure 14: Date Features and their relationship with repeated listening

4.6. Song Type

Song type consist of 3 categories – long, short, medium. These are classified based on song length. As it will be expected, long and short duration songs will be less compared to medium length songs. Length of the song, has influence on the repeated listening. Short and long songs have less chance of getting repeated compared to a medium duration song (Figure 15).

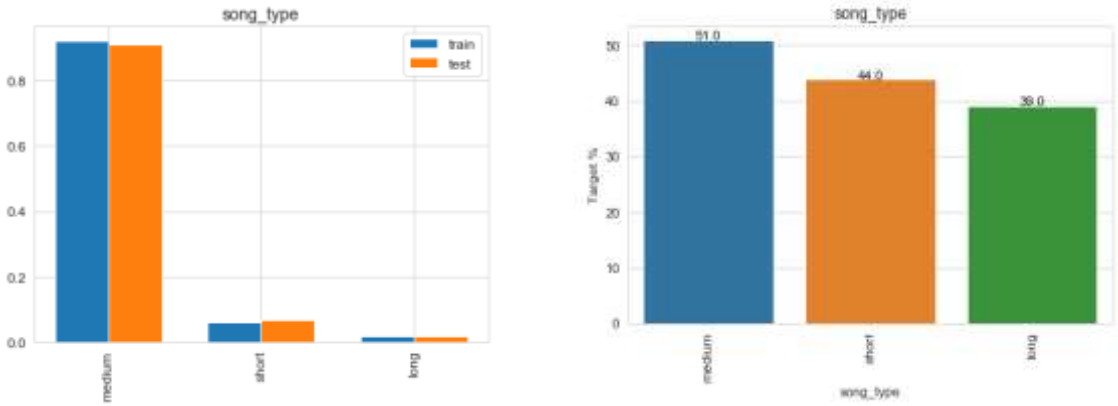


Figure 15: Song Type and its influence on the repeat listening

4.7. Registered Via

This features give information on how did the customer registered for the membership. 9 and 7 are the most frequently used ways to get registered. Fig shows that this feature will not have significant influence over the repeated listening behaviour.

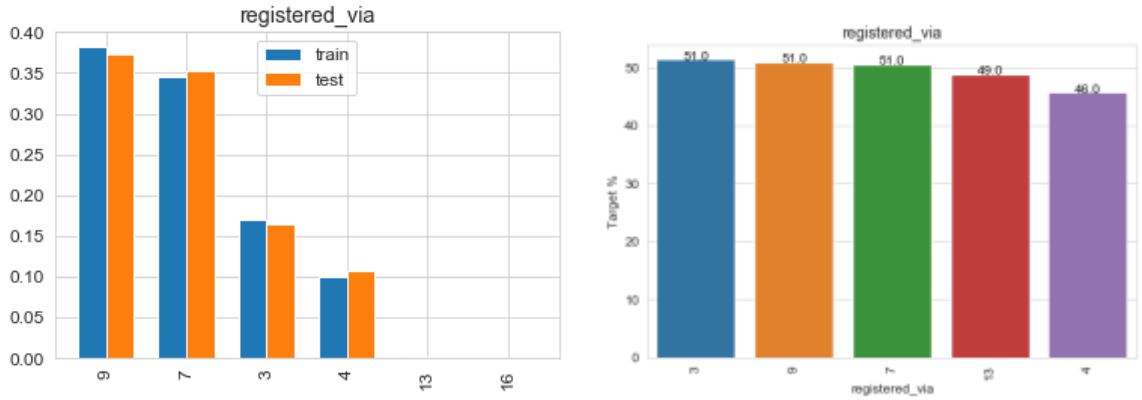


Figure 16: Registered Via and its influence on the repeat listening

4.8. Numerical Features

Following numerical features were explored:

- song_length
- age
- count_song_played
- count_artist_played
- duration
- age_of_song
- isrc_year
- song_freq

Above features are log normally distributed except for duration which shows a bimodal distribution (Figure 17). Song length shows variation in distribution for target variables, supporting the song type hypothesis. Age doesn't show significant influence on the repeated listening behaviour. Variables like count_song_played, count_artist_played, age of song, isrc_year and song _freq shows different distribution between target variables.

Figure 18 is the heat map of correlation between all the numerical variables and target. This quantifies and support some of the findings from above exploratory analysis. There are some collinearity observed for extracted features. The prominent was registration year and duration that shows 0.99 correlation, implying that we can drop one of the feature. So we decided to drop registration year.

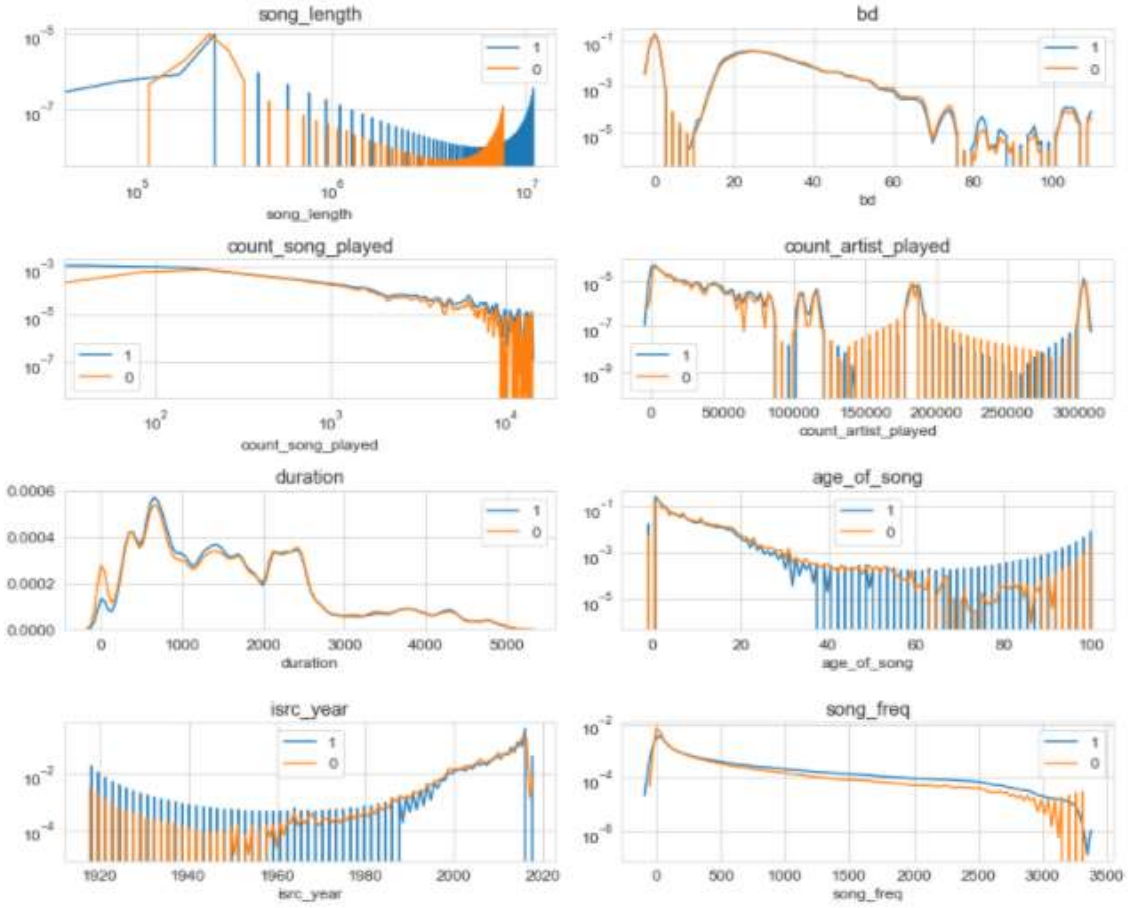


Figure 17: Distribution numerical features between repeated and no-repeated listening

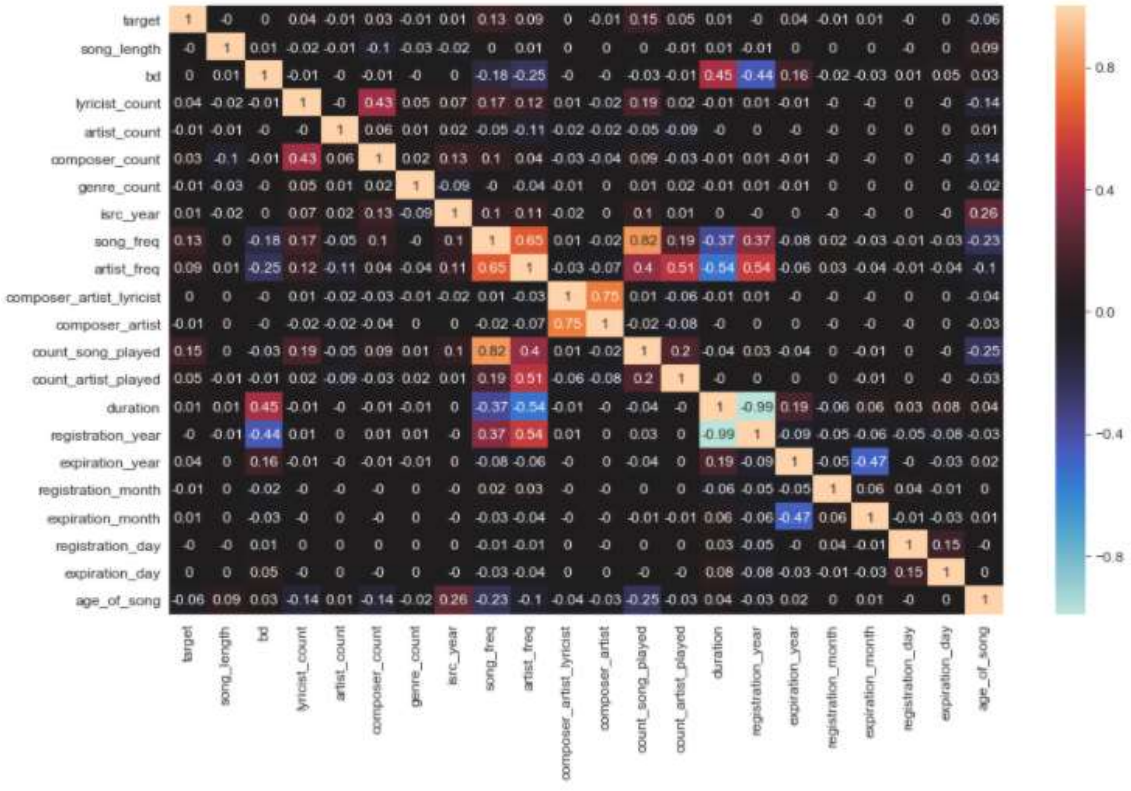


Figure 18: Heat Map of Correlation between numerical features

4.9. Artist and Song Frequency

Artist and song frequency were created to capture time element and the number of times the song or artist was played. The feature measures the cumulative number of times a song was played as the time passed by. Intuition says, the more a song is played higher is the chance that this will be listened frequently by the users (Figure 19 and Figure 20). Also as the song gets old, the frequency might decrease. This is supported by the negative correlation of -0.23 between age of song and song frequency (Figure 18).

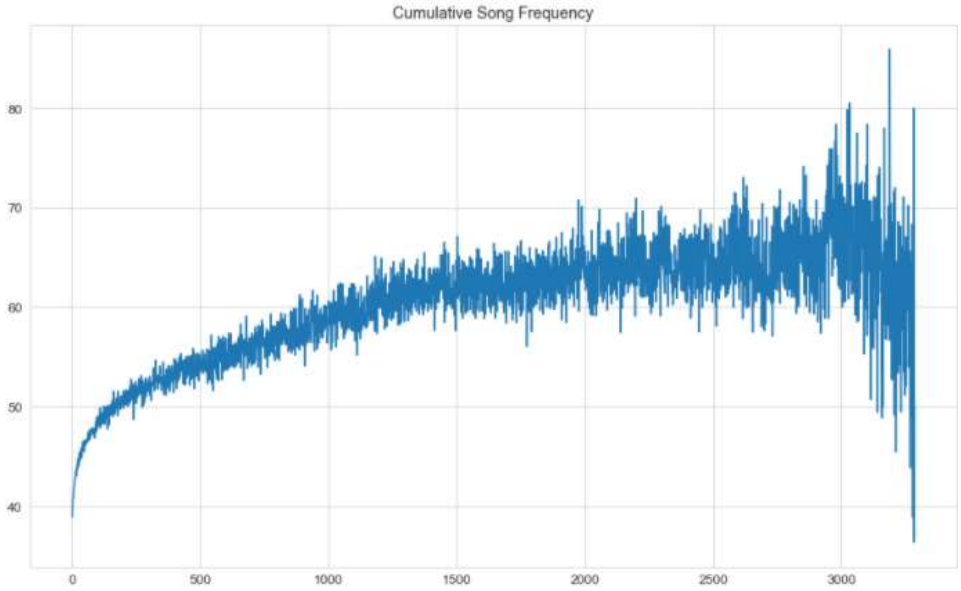


Figure 19: Song Frequency and Target

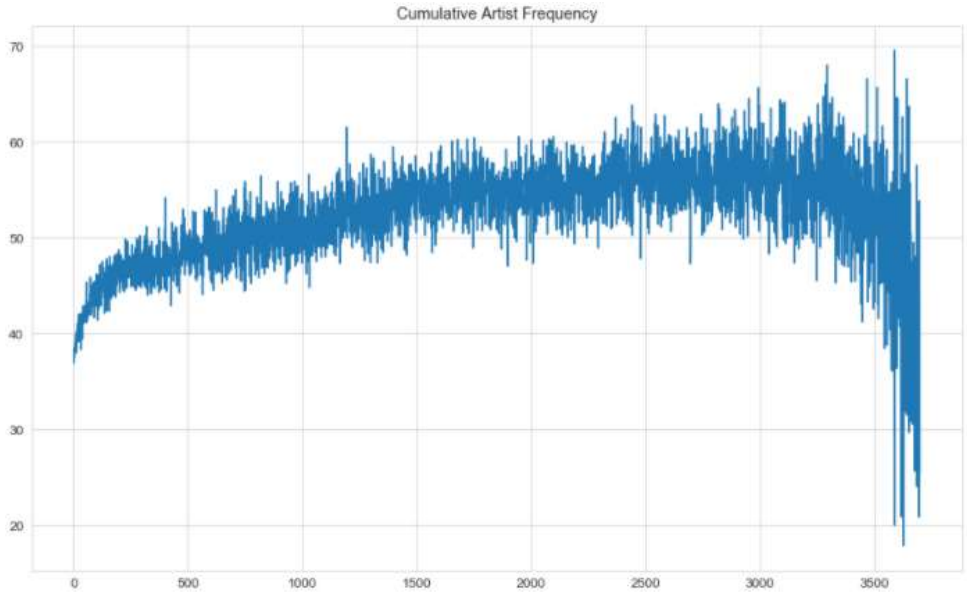


Figure 20: Artist Frequency and Target

4.10. Summary and Key Findings

- Exploring testing data tells us that we will have to deal with cold start problem during modelling to make a robust model.
- This is a balanced data set
- Distribution variations between test and train says there has been slight change in user behaviour over time
- Features that will significantly influence repeated listening are:
    - source_type, source_system_tab, source_screen_name
    - language
    - composer_count, artist_count and genre_count
    - expiration year
    - song_type
    - song_length
    - song and artist frequency
    - count_song_played, count_artist_played
    - membership duration
    - age of song
    - isrc_year
- Features that will not significantly influence repeated listening are:
    - Age
    - Lyricist count
    - Composer_artist_lyricist
    - Composer_artist
    - registration year, registration_day, registration_month
    - expiration_day, expiration_month
    - city
    - gender
- Feature selection will be done during base line evaluation model as discussed in next section.
- Categorical features like lyricist, artist, composer, were not touched in this analysis. They will be used to later in the project.

5. Modelling

This problem can be solved using collaborative/content based filtering approach (CF) and classification models. Collaborative filtering works wells but presents the challenge of data sparsity and cold start problems. Working with limited computational power, I followed approached the problem using a combination of classification and model based CF. We will use 50% of the data for training and 50% of the data as validation set. The split % was decided keeping in mind memory and speed efficiencies. If we used 60% of the data for training then models failed to run and we encountered memory errors. This is approximately a balanced data set.

5.1. Data Pre-processing

Before we feed the data to a machine learning algorithm, we need to perform some pre-processing steps. Below is the outline of all the steps performed during pre-processing. Note that there are some steps which would not be required for some algorithms but for the sake of completeness we are mentioning all the steps in the order they are performed:

There are both numerical and categorical values present in the data set. For numerical and binary class, there is no need for encoding. For categorical values, both target encoding and one hot encoding was done depending on category size.

1) **Target Encoding:** Features with more than 25 sub-categories where one-hot encoding was note feasible, were used for target encoding. Smoothing constant was used to ensure low frequency features are not given much weightage. Genre_ids, composer, artist_name, lyricist and isrc_country were target encoded.
2) **One Hot Encoding:** For < 25 sub-categories, I have used one hot encoding.
3) **Data Splitting:** In this second step, we split our labelled data into 50% training and 50% validation data set. We ensure that fraction of both classes are same in training and test data set.
4) **Data Size Optimizing:** Reducing the data size by down casting variables. This would ensure low memory usage.
5) **Scaling:** To remove the bias of large values, it is require to scale the numeric variables so that they lie within a specified range. We used MinMax scaling technique for scaling.

5.2. Modelling and Evaluation metrics

After pre-processing of the data, we the use the data to train and build our model. Thereafter, model is evaluated on test data. It was ensured that the same pre-processing steps used for training set were applied to test and validation data. Due to computational limitation, hyper-parameter tuning was not performed.

Accuracy and area under the curve (AUC) of receiver operating characteristics (ROC) curve were used as the metrics to evaluate the performance. They are popular metrics in this type of classification and balanced data sets.

As a part of modelling, an initial base line model was evaluated. That base line model was then used to remove features that were not relevant. Post the feature selection, new latent features were extracted using SVD decomposition to further improve the accuracy of the model.

6. Base line Evaluation

The data for the base line evaluation consists of 65 features post data pre-processing. Following algorithms were used during

- Logistic Regression
- SVM
- Decision Tree
- Random Forest
- XGBoost

Features were iteratively removed and added to see the effect on AUC and accuracy score. 6 features were identified which didn't have significant influence on metrics. They were registration_day, expiration_day, registration_month, composer_artist_lyricist, gender and composer_artist. This is very well supported by the results obtained from exploratory analysis. The variation in AUC score obtained was not more than 0.05%

After the removing the features, results obtained are showed in Table 122.

Model	Training AUC	Testing AUC (Validation)
Logistic Regression	0.5903	0.5905
SVM	0.6820	0.6825
Decision Tree	0.6892	0.6896
Random Forest	0.7363	0.7250
Gradient Boosting	0.7114	0.7117
XGBoost		0.7322

During modelling, issues of memory error was observed with XGBoost. So as an alternative, LightGBM and Catboost will be tried during final modelling.

As expected, this being a non-linear problem Logistic Regression performed the worst while random forest performed the best.

During base line evaluation, normalization was not done. With normalization, slight better AUC would have been obtained. As a next step, we will create more features through SVD decomposition using users and song metadata.

7. Truncated SVD Based Embedding Features

Feature engineering is important part of any machine learning project and goes a long way in improving the performance of the model. The current data set doesn't have many features and existing features are very not useful in its current form. Eg. Artist and composer name doesn't gives much information unless we relate it with users and songs. So, we need the help of content based filtering to extract more information. For this project SVD based Embedding features were created.

SVD or "Singular Value Decomposition" is a matrix factorization techniques. The singular value decomposition (SVD) of an nxd matrix A expresses the matrix as the product of the three simple matrices:

$A = U S V^T$

where:

(1) U is an nxn orthogonal matrix.

(2) V is an dxd orthogonal matrix.

(3) S is an nxd diagonal matrix with nonnegative entries, and with the diagonal entries sorted from high to low.

Example: If we create a user – song matrix then U will give us user latent factors, S will be strength of each latent factor and V will be Song latent Factors.

We created latent features using with the following combinations

- User – Song Pairs
- User – Genre_ids
- User – Artists

All of the above will give us sparse matrix, so we used truncated SVD. It can deal with sparse matrix and also produces a factorization where the number of columns can be specified for a number of truncation. To create the embedding features based on truncated SVD, we proceed as follows.

a) First, the sparse matrix A is created
b) Then, the k dimensions of the matrices U or V are extracted as embedding factors. Plot of explained variations with no. of component was used to select the k. Computational power was the secondary condition.

User – Song Pairs: k = 20 and return U as embedding factors

User – Genre_ids: k = 5 and return U as embedding factors

User- Artist: k = 12 and return U as embedding factors

More details can be found in **IPython Notebook.**

```
Total Explained Variation: 0.1702441027400643
```

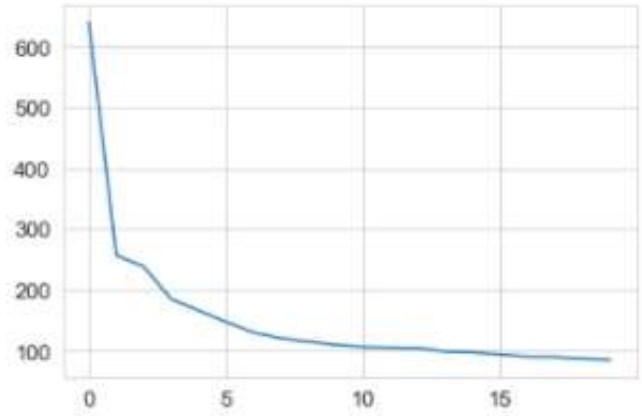


Figure 21: User - Song explained variations of extracted features

```
0.9679396048126867
[<matplotlib.lines.Line2D at 0x27a5e4facf8>]
```

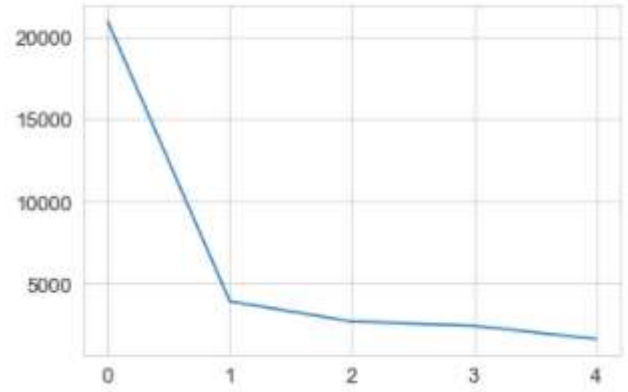


Figure 22: User – Genre_ids explained variations of extracted features

```
0.45846842453414877
[<matplotlib.lines.Line2D at 0x27a5ebbe358>]
```

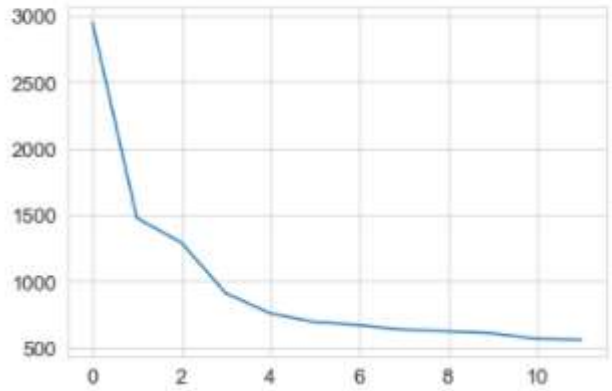


Figure 23: User – Genre_ids explained variations of extracted features

8. Final Modelling

Post feature engineering, the final model was built using logistic regression, SVM, random forest, LightGBM and CatBoost. Hyperparameter tuning was done only on catboost which gave us the best results while for others default values were used.

8.1. Logistic Regression

All the 4 steps of pre-processing were followed. No hyper parameter tuning was done on this model. Both L1 and L2 regression was performed.

Final results for L1 and L2 regression are given in Table 3.

Model Name	Training Time	Metric	Training	Testing
L1 Regression	92 mins	Accuracy	0.6563	0.6569
		ROC AUC	0.7119	**0.7126**
L2 Regression	98.7 min	Accuracy	0.6563	0.6569
		ROC AUC	0.7120	**0.7127**

Table 3: Final Result - Logistic Regression

Above results suggests that our models predicts label "1" more accurately than label 0. Precision and recall for both the labels are approx. similar. Class of interest have 68% recall, and AUC is more than 0.5 (Figure 24 and **Error! Reference source not found.**). This means that our model is better than luck or chance. Compared to base line model, there is a 20% increase in AUC score. This is inferred to be the effect of normalization.

Classification Report - Testing

	precision	recall	f1-score	support
Not Listened	0.66	0.63	0.65	1831842
Listened	0.65	0.68	0.67	1856867
accuracy			0.66	3688709
macro avg	0.66	0.66	0.66	3688709
weighted avg	0.66	0.66	0.66	3688709

Figure 24: Classification Report - Logistic Regression

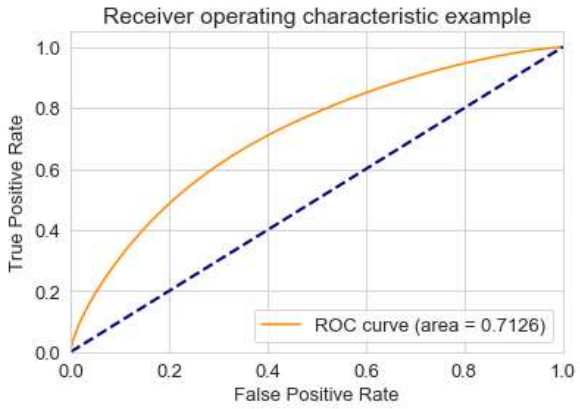


Figure 25: ROC AUC Curve - Logistic Regression - L2

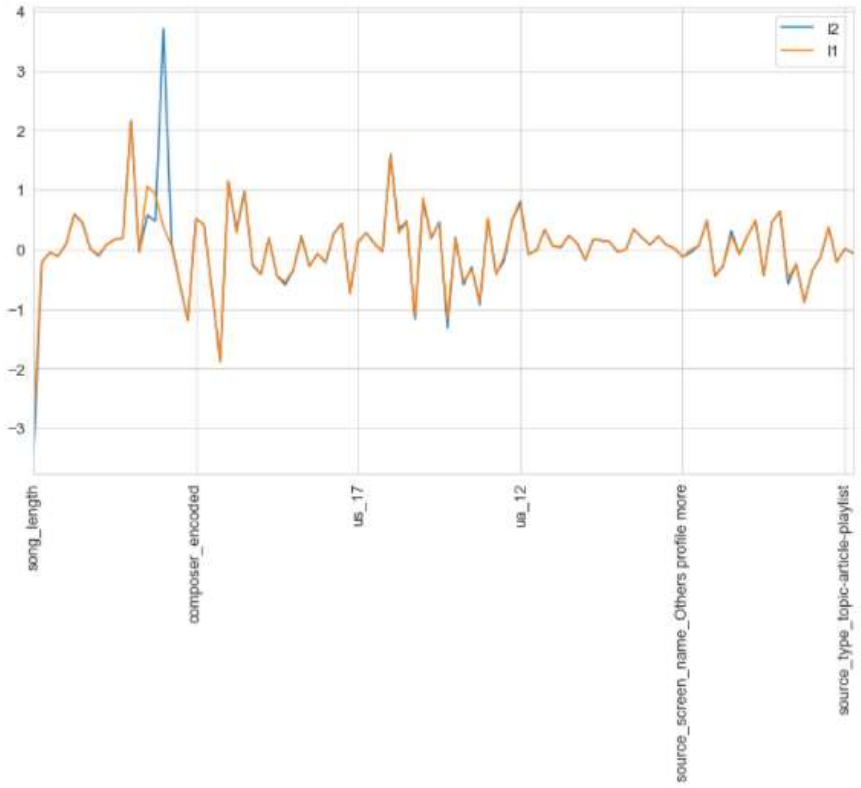


Figure 26: Comparison of coefficients for L1 and L2 Regularization

Through L1 regularization we found only one features with zero coefficient. It was source_screen_name_payment. A comparison of coefficients for features between L1 and L2 are given in Figure 26.

8.2. Support Vector Machines (SVM)

Another algorithm that we tried was support vector machines. It is one of the popularly used off the shelf classifier. Linear kernel was used. It is widely used in text classification, face recognition etc. Similar to logistic regression, all 4 pre-processing steps were followed. Normal Standardization was used for Scaling. The package used for this was LinearSVC from scikit-learn

Final Result post normalization is given in Table 4.

Model Name	Time	Metric	Training	Testing
SVM		Accuracy	0.6560	0.6566
		ROC AUC	0.7117	0.7125

Table 4: Final result - Linear SVC

SVC was faster than Logistic Regression in terms of computational time. In this model, there was improvement in precision for the “0” label. Overall accuracy and ROC AUC was slightly less than Logistic Regression.

```
Classification Report - Testing
              precision    recall  f1-score   support

Not Listened       0.66      0.63      0.65   1831842
    Listened       0.65      0.68      0.67   1856867

    accuracy                           0.66   3688709
   macro avg       0.66      0.66      0.66   3688709
weighted avg       0.66      0.66      0.66   3688709
```

Figure 27: Classification Report - SVM

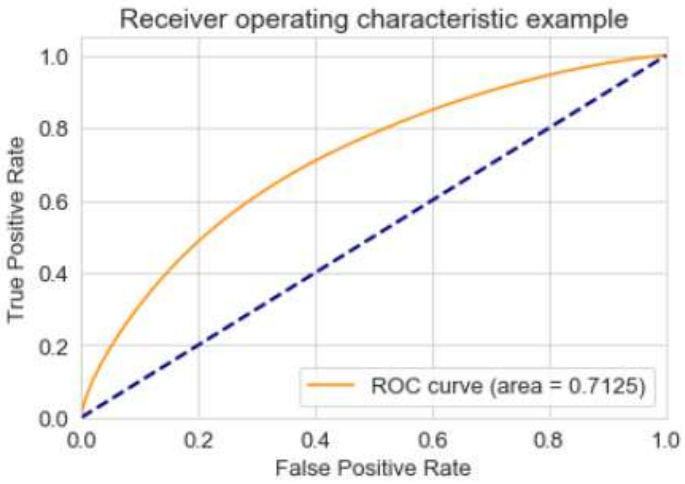


Figure 28: ROC AUC Curve - SVM

Top 10 features with highest absolute coefficient values are given below. The latent features created through svd are showing high importance compared to other features. This is reflected in higher ROC values than base line model.

```
Top 5:
                    Coefficient
us_3                   0.426403
us_1                   0.491930
ug_1                   0.682871
count_song_played      0.892494
expiration_year        2.258697

Bottom 5:
                    Coefficient
song_length           -1.196208
isrc_country_encoded  -0.822923
ua_3                  -0.560514
ug_4                  -0.522763
genre_ids_encoded     -0.503024
ua_7                  -0.408180
```

8.3. Random Forest

The current problem is non-linear and so ensemble models like Random forest would help us give better results. Normalization was performed in this model though no major effect is expected.

Final Result post normalization is given in Table 5.

Model Name	Training Time	Metric	Training	Testing
Random Forest	23.3 mins	Accuracy	0.7249	0.7099
		ROC AUC	0.8050	**0.7856**

Table 5: Final result - Random Forest

This model shows good improvement over SVM and Logistic Regression. The AUC curve and classification results are shown in Figure 29 and Figure 30. Source and Latent Features dominate the top 10 feature important list as shown in Figure 31.

```
Classification Report - Testing
              precision    recall  f1-score   support

Not Listened       0.70      0.72      0.71   1831842
    Listened       0.72      0.70      0.71   1856867

    accuracy                           0.71   3688709
   macro avg       0.71      0.71      0.71   3688709
weighted avg       0.71      0.71      0.71   3688709
```

Figure 29: Classification Report - Random Forest

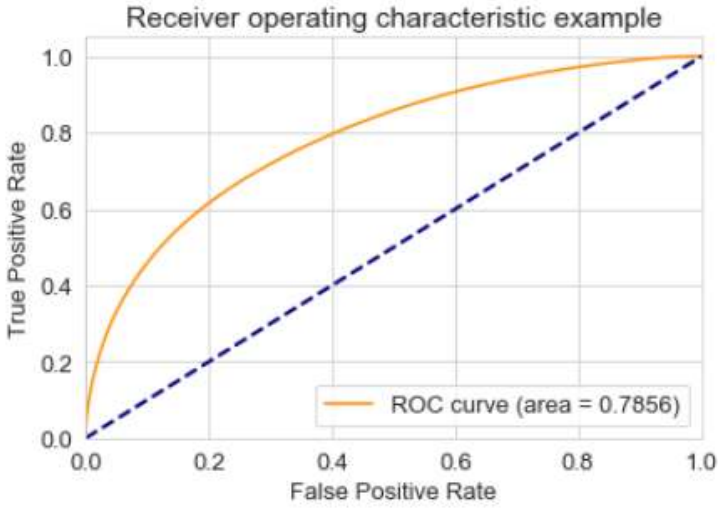


Figure 30: ROC AUC Curve – Random Forest

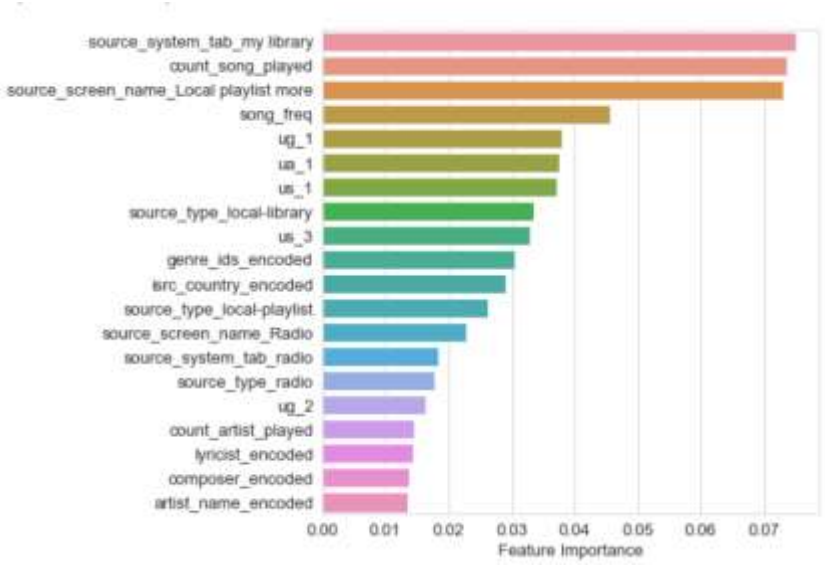


Figure 31: Feature Importance - Random Forest

8.4. LightGBM

LightGBM is high-performance gradient boosting framework based on decision tree algorithm. It has faster training speed and higher efficiency. At times, it gives better accuracy than other boosting algorithm. For this model, similar to random forest normalization was done though it was not needed.

Final Result post normalization is given in Table 5.

Model Name	Training Time	Metric	Training	Testing
LightGBM	5.3 mins	Accuracy	0.6962	0.6959
		ROC AUC	0.7679	**0.7675**

Table 6: Final result - Random Forest

LightGBM performed better than SVM and logistic regression but poorer than random forest. The poorer performance can be because of no proper tuning. But this was not investigated further. But this model gave significant speed improvements. The AUC curve and classification results are shown in Figure 32 and Figure 33. Contrary to Random forest, in this model Latent Features dominate the top 10 feature important list as shown in Figure 34.

```
Classification Report - Testing
              precision    recall  f1-score   support

Not Listened       0.69      0.70      0.70   1831842
    Listened       0.70      0.69      0.70   1856867

    accuracy                           0.70   3688709
   macro avg       0.70      0.70      0.70   3688709
weighted avg       0.70      0.70      0.70   3688709
```

Figure 32: Classification Report – LightGBM

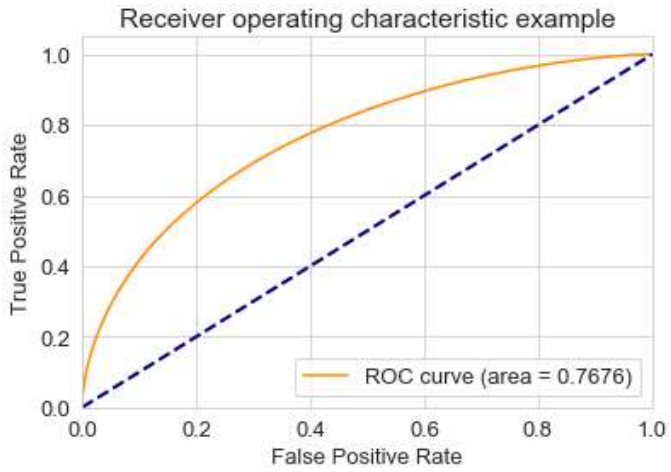


Figure 33: ROC AUC Curve – LightGBM

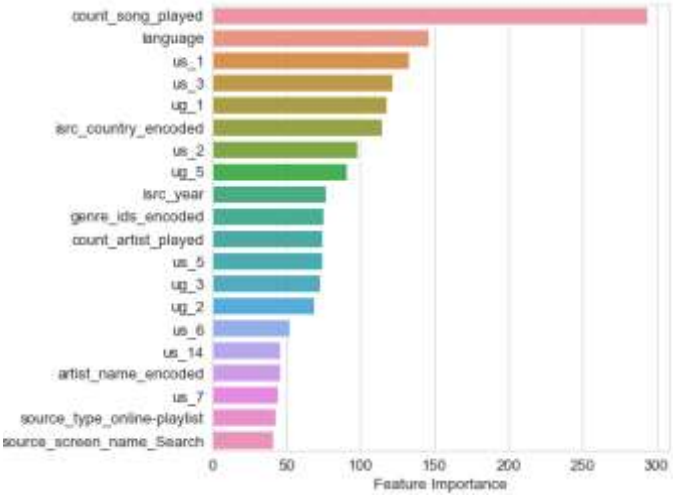


Figure 34: Feature Importance – LightGBM

8.5. CatBoost

My Dataset has lot of categorical variables compared to numeric variables. CatBoost is newly developed algorithm to handle categorical data. It is based on gradient boosting of decision trees. In this model there is no need of encoding categorical data. Without any pre-processing it converts categories into numbers using innovative algorithm. Thus for CatBoost only step 1 of target encoding was not done. Rest all steps were performed.

This model gave results better than all the above models.

Model Name	Training Time	Metric	Training	Testing
CatBoost	61 mins	Accuracy	0.7596	0.7366
		ROC AUC	0.8412	**0.8149**

Table 7: Final Result post Hyper parameter Tuning - CatBoost

The AUC curve and classification results are shown in Figure 35 and Figure 36. Source and Latent Features dominate the top 10 feature important list as shown in.

Classification Report - Testing

	precision	recall	f1-score	support
Not Learning	0.73	0.74	0.73	1831842
Learning	0.74	0.74	0.74	1856867
accuracy			0.74	3688709
macro avg	0.74	0.74	0.74	3688709
weighted avg	0.74	0.74	0.74	3688709

Figure 35: Classification Report – CatBoost

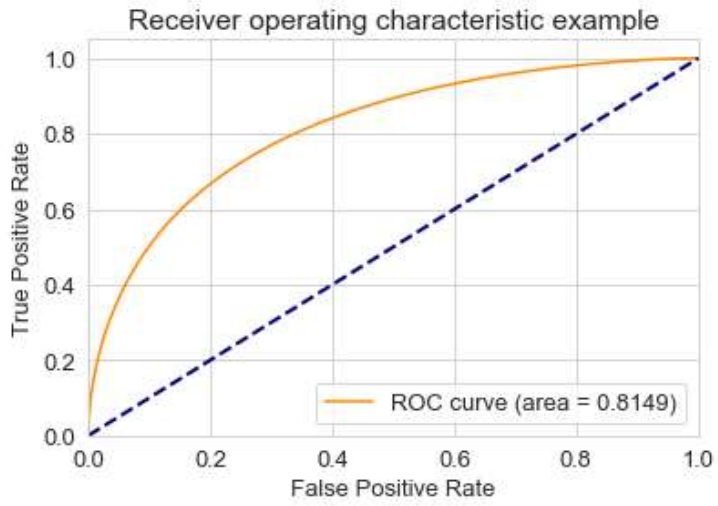


Figure 36: Feature Importance – CatBoost

It has similar set of feature importance as with LightGBM.

Top 5:

Feature	Importance
count_song_played	11.121768
ug_1	7.720955
us_1	5.369432
source_system_tab_my library	3.421270
language	3.222950
artist_name	3.071420
source_screen_name_Local playlist more	2.884552
count_artist_played	2.853216
us_3	2.766137
ua_1	2.372643
source_type_local-library	2.038195

8.6. Model Comparisons

We have used Logistic Regression, Linear SVC, Random Forest, LightGBM and Catboost classifiers to build a model a music recommendation. Based on testing the models on test data, there were differences in performance of all the models. The results are shown in Table 8.

Model Name	Accuracy	ROC AUC Score	Computational time
Logistic Regression	0.6569	0.7127	~98.7 mins
SVM	0.6566	0.7125	~10 mins
Random Forest	0.7099	0.7856	~23.3 mins
LightGBM	0.6959	0.7675	~5.3 mins
CatBoost	0.7366	0.8149	~61 mins

Table 8: Comparison of different models. Red for the worst and Green for the best

Other metrics like Log Loss, Brier Loss, PR AUC score can also be used for comparison. They were not calculated in this project. Table shows that linear models like logistic regression and SVM performed the worst while tree based models performed better. There is also variation in computation time. LightGBM was the fastest while Logistic regression took almost 1.5 hours.

Across all the models, latent features, count_song_played and source features were found in top 10 in feature importance list.

9. Using Model and Recommendations

To now use the best model, cleaning, pre-processing steps of Splitting Data and Scaling would be needed. Feature Engineering would be an additional step to improve model performance. Some features were removed due to their non-correlation with the target values.

Also it is recommended to use probability values from the model to make top 10 recommended songs for the users. This model can also be used to personalize song recommendations based on the specific source where the user listens the most. It can also be used to create curated playlists for the users.

10. Assumptions and Limitations

- Computational Power was a big limitation in improving this model through hyper-parameter tuning.
- This model is limited to predicting music songs only
- Cold Start was a big challenge in this project and global average values were used to deal with it. Cold start can be tackled better through using values based on similarity, song metadata or user metadata.

11. Future Work

- Including more latent features combining user and song metadata through word embedding.
- Trying deep learning methods to improve the performance using their capacity to generate features.
- Capturing more data like, no. of times a song was listened by a user, time of listening of the songs (morning, evening, etc.) can help is better and more personalized song recommendations.

12. Conclusion

The data set for this project was sourced from a Kaggle Competition conducted by WSDM. The first step in this project was to explore data for any missing values. Missing values for each of the column were handled differently. Then we did some data wrangling to make data more analysable. Post this, we further explored the data through various visualization to understand the distribution of the features. EDA was an important step in feature selection. Feature engineering was done using truncated SVD to extract latent features. We understood that importance of feature engineering and it is an irreplaceable ingredient for any ML/AI project. It is extremely helpful to improve performances in most cases.

Post the feature engineering analysis, we build 5 different models consisting of both linear and ensemble methods. Hyper parameter was done due to computational limitations. Based on accuracy and ROC AUC score, CatBoost was selected as the final model for implementation. It gave the highest ROC AUC score of 0.8149 and accuracy of 73.7%. There was lot of variations in computational time between models.

Obviously, when datasets are large, time costs may become an issue, since in our approach, we generate features to then evaluate and possibly retain them. It's imperative to learn and evaluate how to balance costs versus performance increase.

The thing that was left to do was working on cold start issues and generating more time dependent features to get better performances. Using neural networks methods would have surely helped because of their capacity to generate features. In this project due to limitation of time, above things were not done.