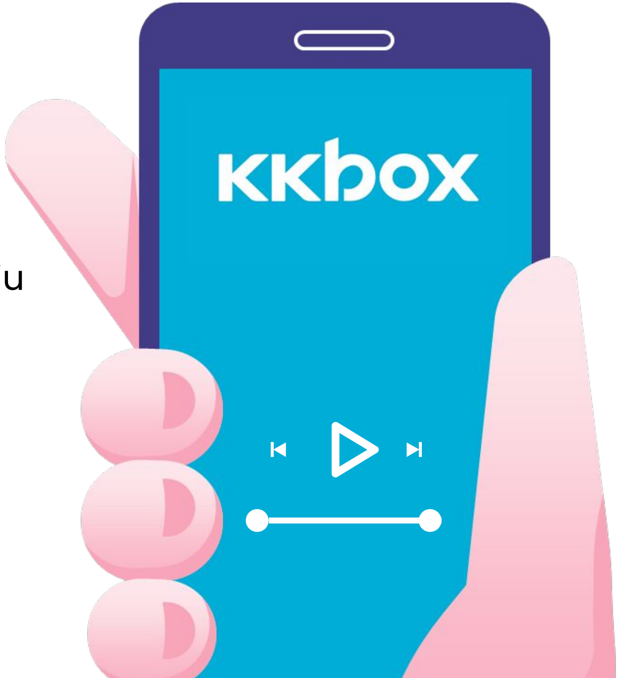
KKBox's Music Recommendation

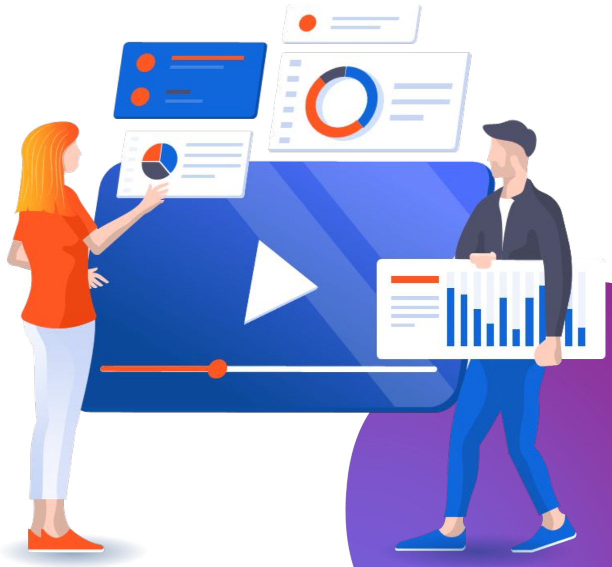
Group7 - Chia-Ling Ni, Yi-Hsuan Tseng, Qian Gao, Ziqi Liu



Agenda

- Project idea and motivation
- Data Description
- Exploratory Data Analysis
- Model Assessments and Results
- Conclusion and Insights
- Future Improvement

Project Idea & Motivation



Project motivation

KKBOX is an Asian leading music streaming service developed in 2005. The service mainly targets the music markets of East and Southeast Asia, holding the world's most comprehensive Asian-Pop music library with over 90 million tracks.

- Provide better music recommendation system based on user behaviors and song features for the app users
- To predict if an user will repetitively listen to a song within a month

Data Description

Data Description

The dataset is from Kaggle, which contains five files with more than **9.9** millions rows:

The **Members file** contains user id, city, user age, user gender, registration method, registration date and subscription expiring date.

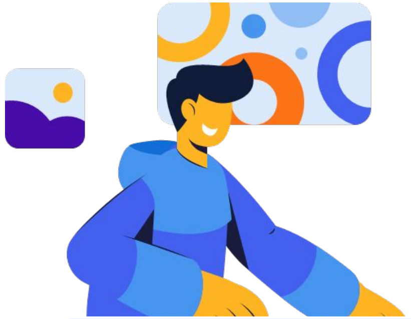
The **Songs file** contains song id, length of song, genre id, artist name, composer, lyricist and language.

The **Song_extrainfo file** contains the name of the singer and isrc.

The **Train file** contains the user id, song id, source system tab, source screen name, source type, and target which represents whether the user re-listens the recommended song again within a month.

Since the dataset is from competition challenge, the **Test file** doesn't have the target variable, we only use train file to build the models.

Data Preprocessing



Data Merge

- Train
- Members
- Songs
- Song_extrainfo

Missing Values Handle

- Some features contain nearly 30% -40% missing values

Features Observe and handle

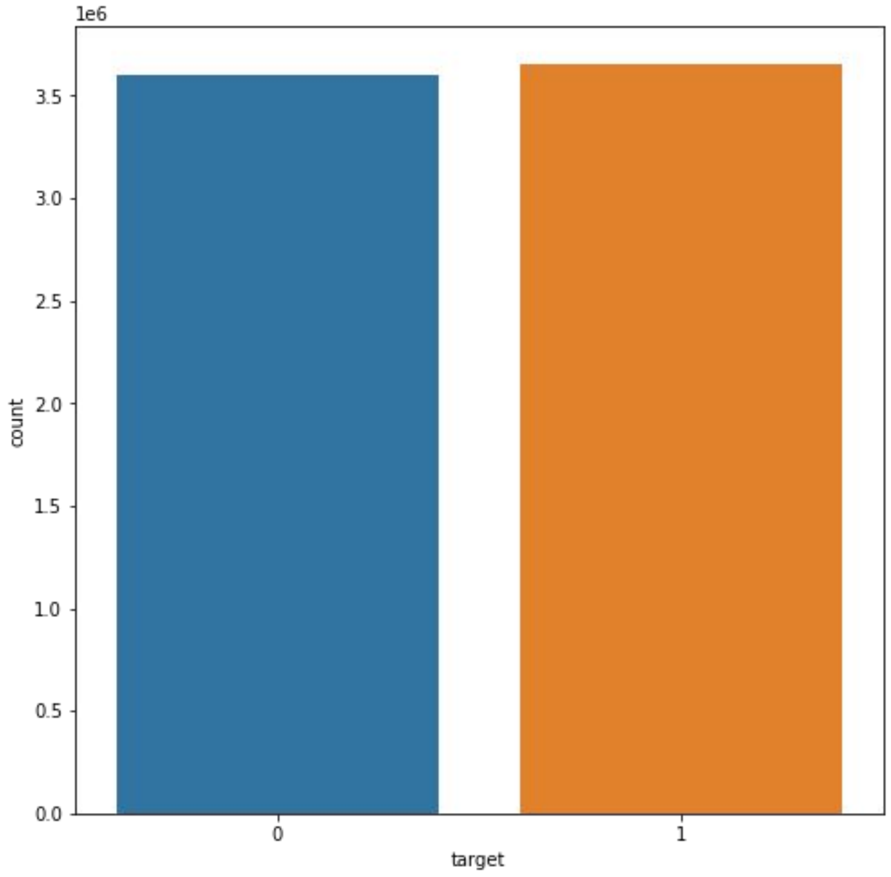
- Pull columns
- correlations

Data Encode and Scale

- Categorical: **LabelEncoder**
- Numerical: **StandardScaler**

Exploratory Data Analysis

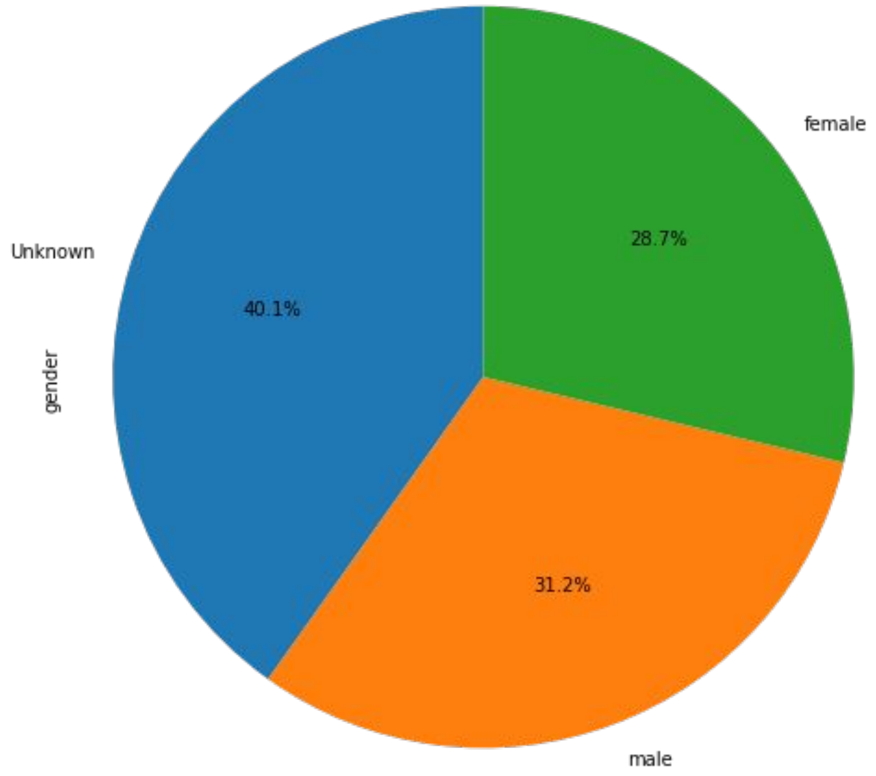
Exploratory Data Analysis – Target



In machine learning classification problems, models will not work as well and be incomplete without performing data balancing on train data.

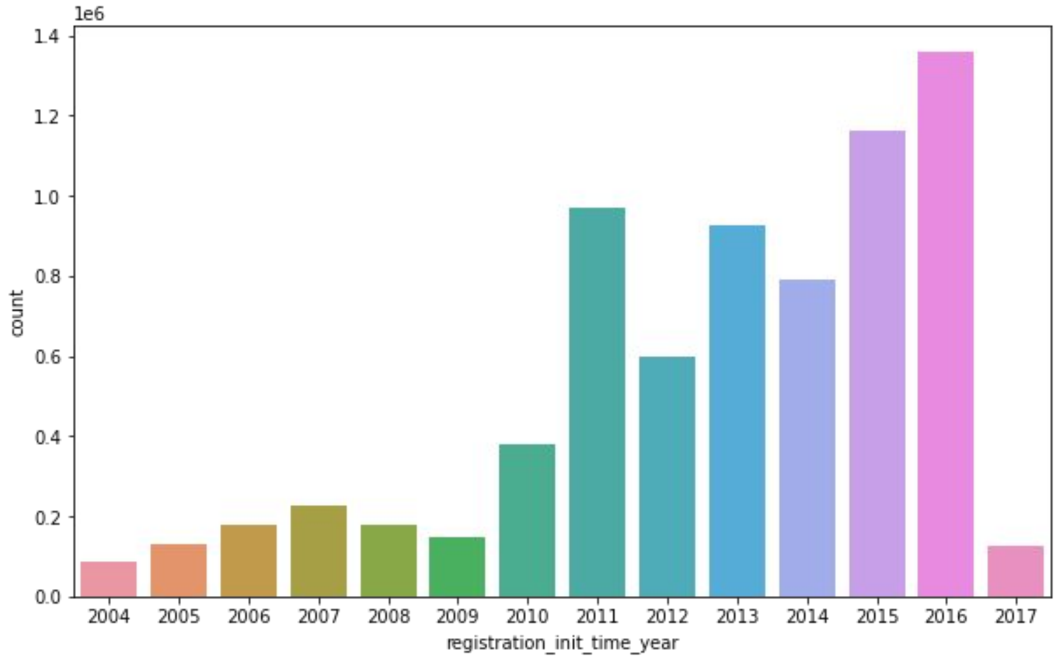
- The ‘target’ variable in our train data is around **balanced**.

Exploratory Data Analysis - Gender



- The portion of female and male users are approximately the same.
- However, 40% of the data is unknown. This result indicates that there might be some missing values or some users refusing to share their gender.

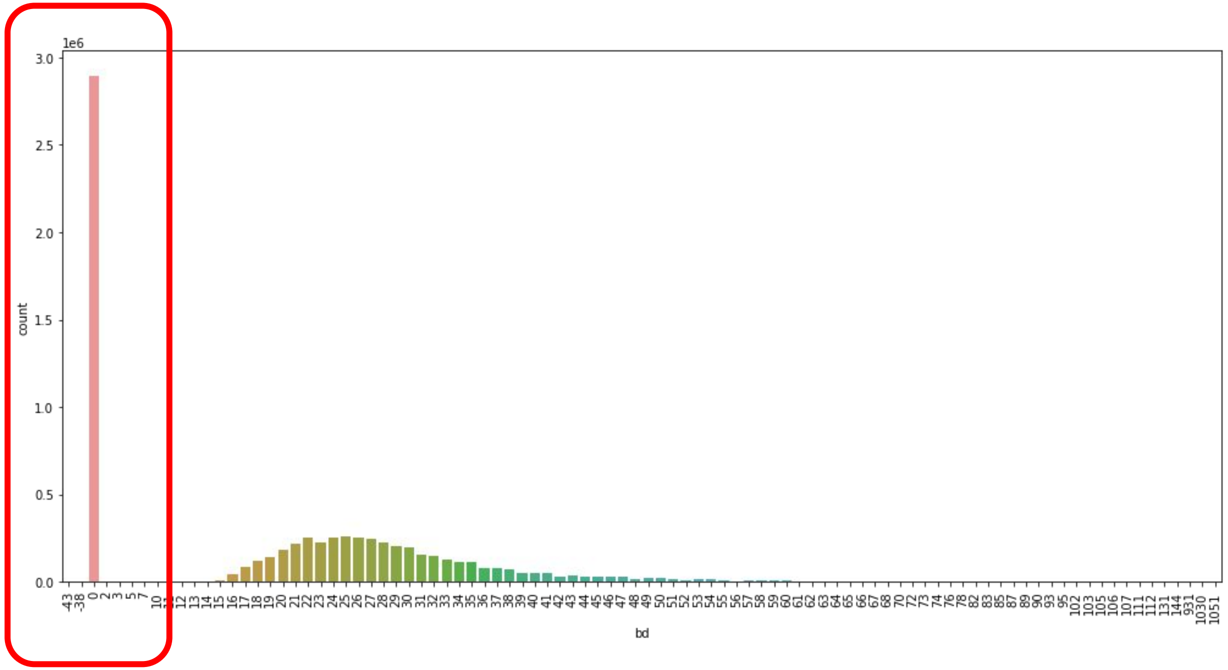
Exploratory Data Analysis - Registration Rate



User Registration Rate:

- Increased a lot since in 2010
- Reached its peak point at 2016.
- Sharply decreased at 2017.

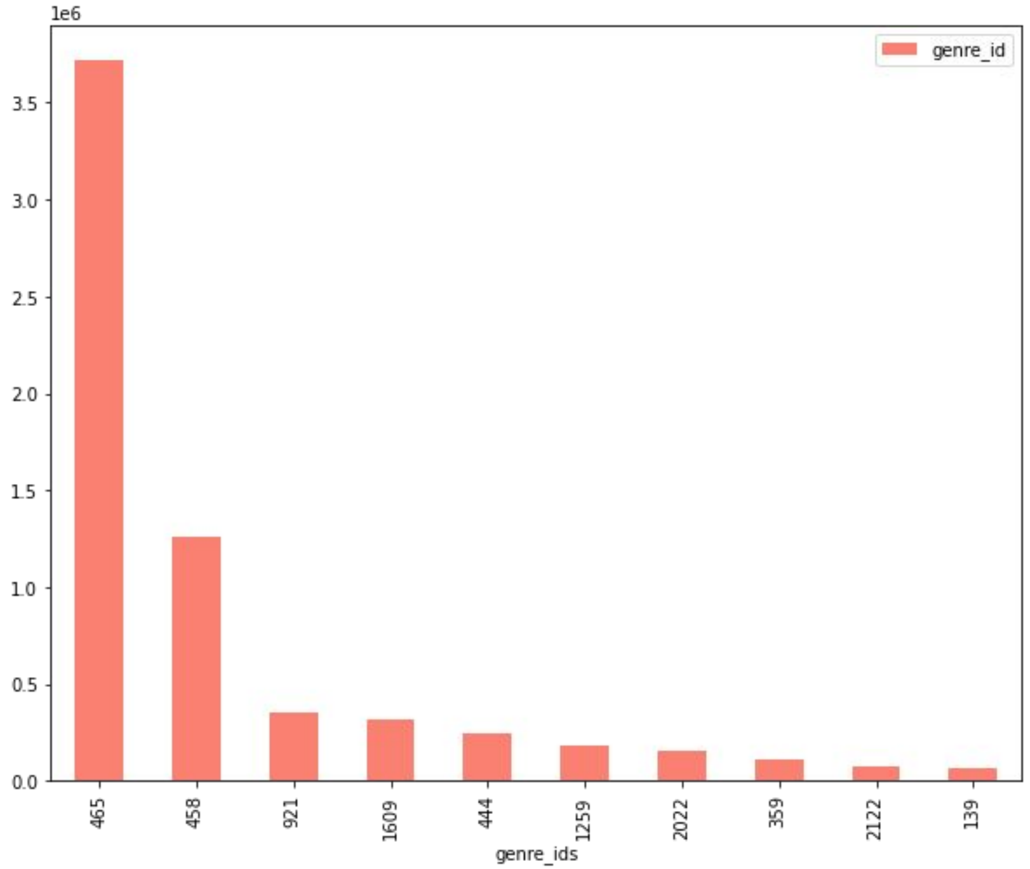
Exploratory Data Analysis - Across Age



- "bd" feature(Age) contains lots of zero values and some negative values like -43.

User cannot have age zero or negative.

Exploratory Data Analysis - Music Genre

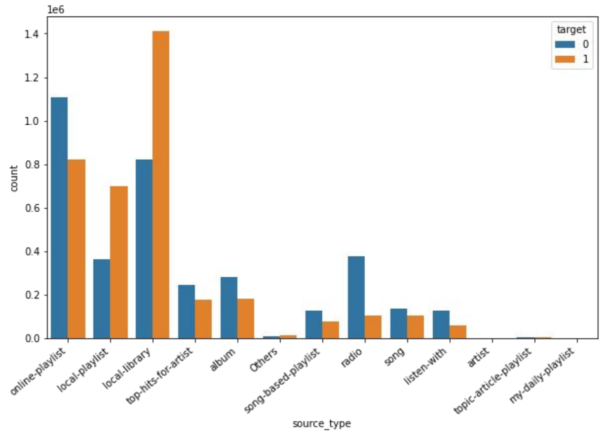


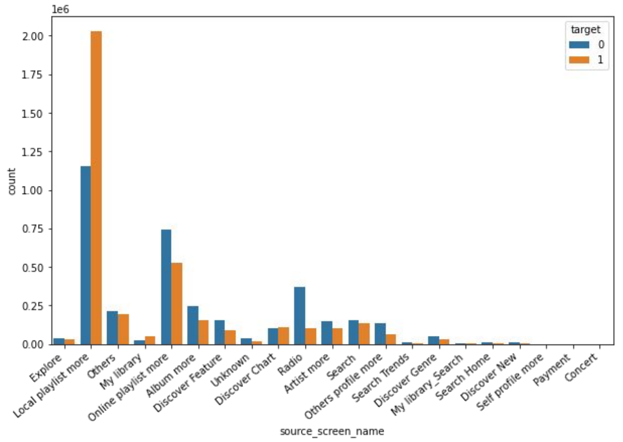
	genre_ids
genre_ids	
465	3717210
458	1261169
921	349958
1609	314913
444	243588
1259	181696
2022	152532
359	109482
2122	71806
139	65816

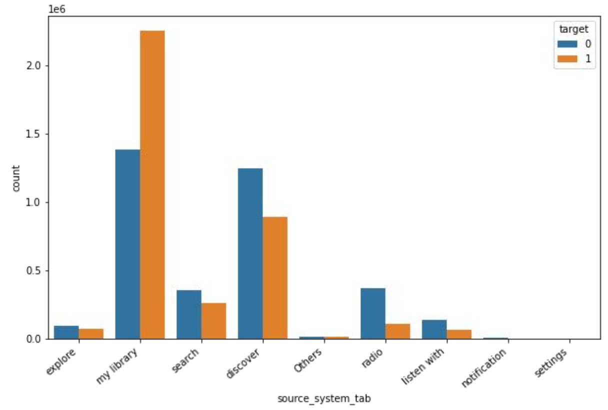
- Genre_id 465 has the highest count: 3717210.

This genre could be the mainstream music in the market so far.

Exploratory Data Analysis - Sources of entry





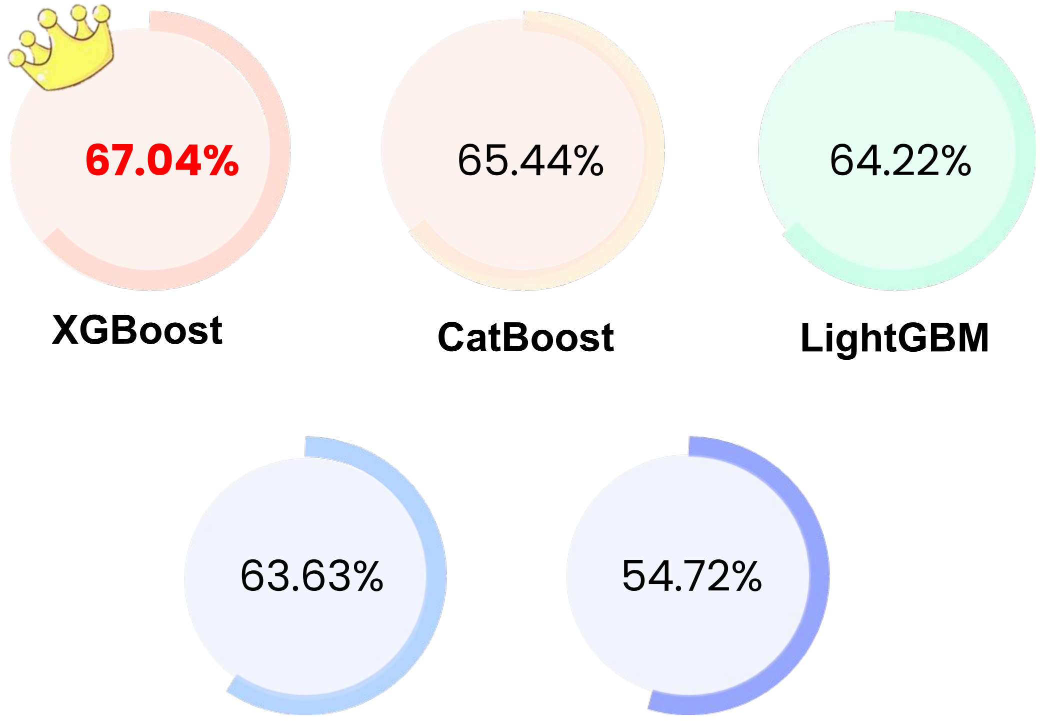


- Local library > online playlist > local playlist
- There are more people using local library as an entry point to first play music on mobile apps.

Users listening from “my library” have the highest chance to listen to the same song within a month. On the contrary, songs played by the radio have the lowest chance to be listened again by the same listener within a month.

Model Methods & Results

Model Methods

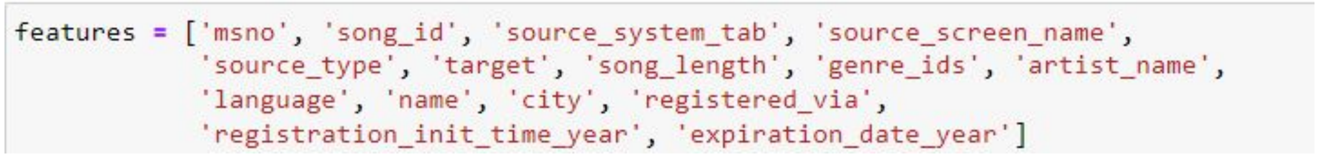


XGBoost CatBoost LightGBM

Random Forest KNN

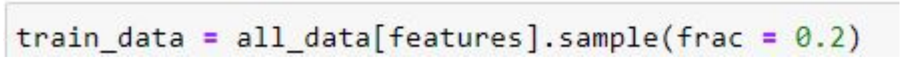
Model Methods and Preprocessing

- Features: 15 features



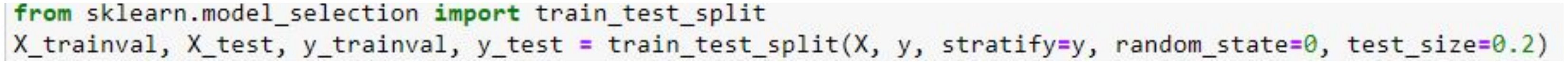
```
features = ['msno', 'song_id', 'source_system_tab', 'source_screen_name',
            'source_type', 'target', 'song_length', 'genre_ids', 'artist_name',
            'language', 'name', 'city', 'registered_via',
            'registration_init_time_year', 'expiration_date_year']
```

- Sample Size: 20% from all data



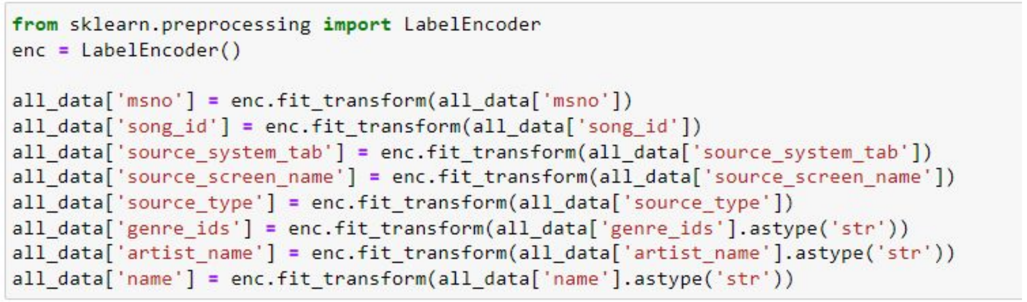
```
train_data = all_data[features].sample(frac = 0.2)
```

- Train-Test: 80-20

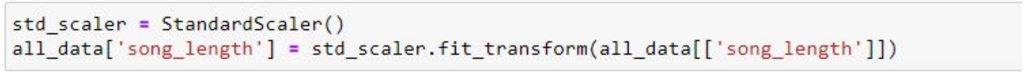


```
from sklearn.model_selection import train_test_split
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, stratify=y, random_state=0, test_size=0.2)
```

- LabelEncoder and StandardScaler

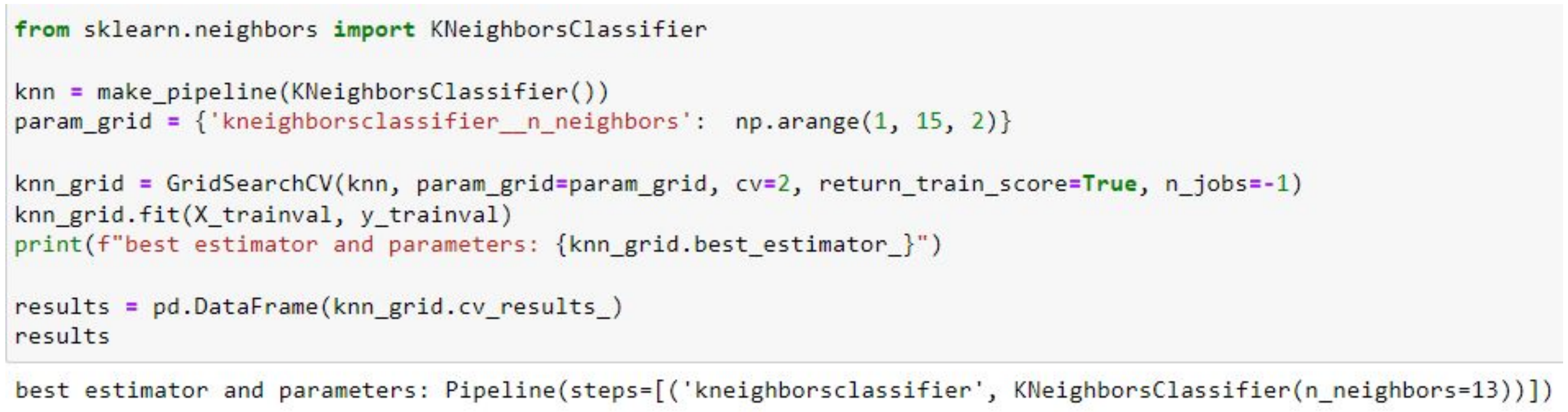


```
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()

all_data['msno'] = enc.fit_transform(all_data['msno'])
all_data['song_id'] = enc.fit_transform(all_data['song_id'])
all_data['source_system_tab'] = enc.fit_transform(all_data['source_system_tab'])
all_data['source_screen_name'] = enc.fit_transform(all_data['source_screen_name'])
all_data['source_type'] = enc.fit_transform(all_data['source_type'])
all_data['genre_ids'] = enc.fit_transform(all_data['genre_ids'].astype('str'))
all_data['artist_name'] = enc.fit_transform(all_data['artist_name'].astype('str'))
all_data['name'] = enc.fit_transform(all_data['name'].astype('str'))
```



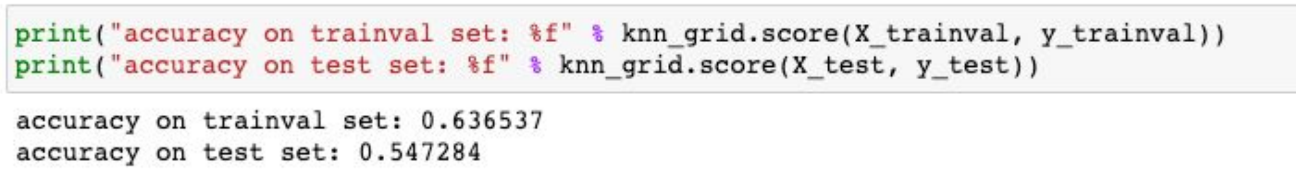
```
std_scaler = StandardScaler()
all_data['song_length'] = std_scaler.fit_transform(all_data[['song_length']])
```

KNeighborsClassifier Model



```
from sklearn.neighbors import KNeighborsClassifier

knn = make_pipeline(KNeighborsClassifier())
param_grid = {'kneighborsclassifier__n_neighbors':  np.arange(1, 15, 2)}

knn_grid = GridSearchCV(knn, param_grid=param_grid, cv=2, return_train_score=True, n_jobs=-1)
knn_grid.fit(X_trainval, y_trainval)
print(f"best estimator and parameters: {knn_grid.best_estimator_}")

results = pd.DataFrame(knn_grid.cv_results_)
results
```

```
best estimator and parameters: Pipeline(steps=[('kneighborsclassifier', KNeighborsClassifier(n_neighbors=13))])
```

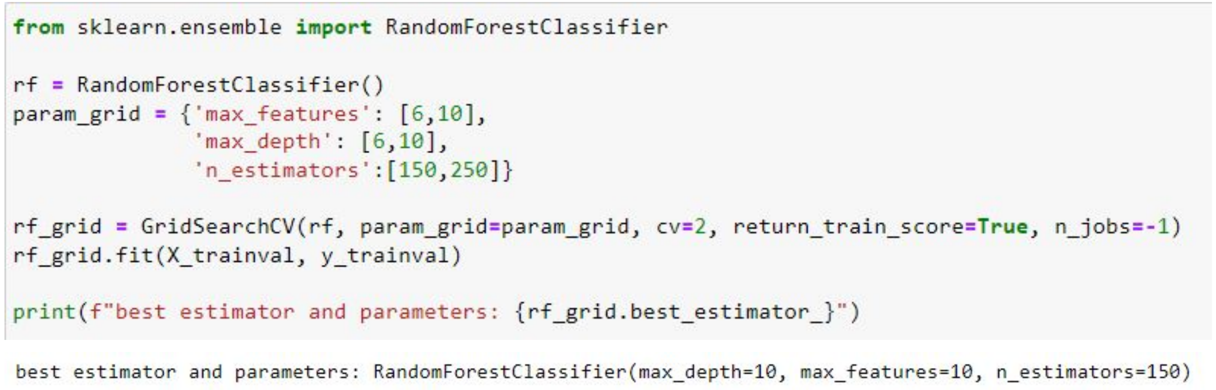


```
print("accuracy on trainval set: %f" % knn_grid.score(X_trainval, y_trainval))
print("accuracy on test set: %f" % knn_grid.score(X_test, y_test))
```

```
accuracy on trainval set: 0.636537
accuracy on test set: 0.547284
```

- KNN model with n=13
  the accuracy rate on test set is only 54%, not so good and has over-fitting problem

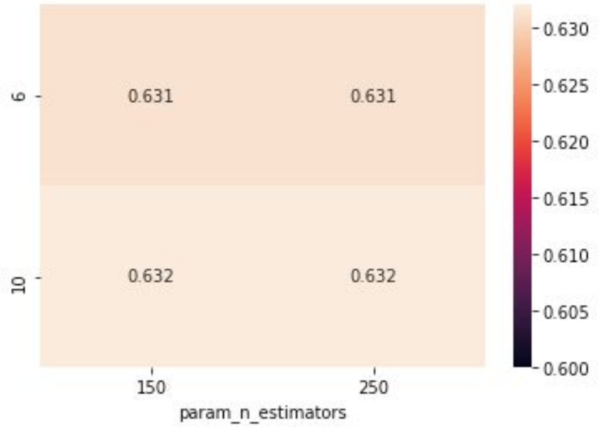
Random Forest Model

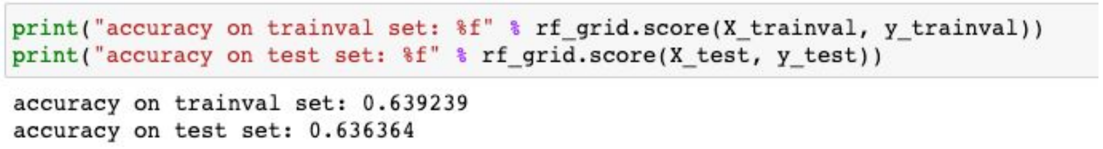


```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
param_grid = {'max_features': [6,10],
              'max_depth': [6,10],
              'n_estimators':[150,250]}

rf_grid = GridSearchCV(rf, param_grid=param_grid, cv=2, return_train_score=True, n_jobs=-1)
rf_grid.fit(X_trainval, y_trainval)

print(f"best estimator and parameters: {rf_grid.best_estimator_}")
```

```
best estimator and parameters: RandomForestClassifier(max_depth=10, max_features=10, n_estimators=150)
```

Mean test score from GridSearchCV



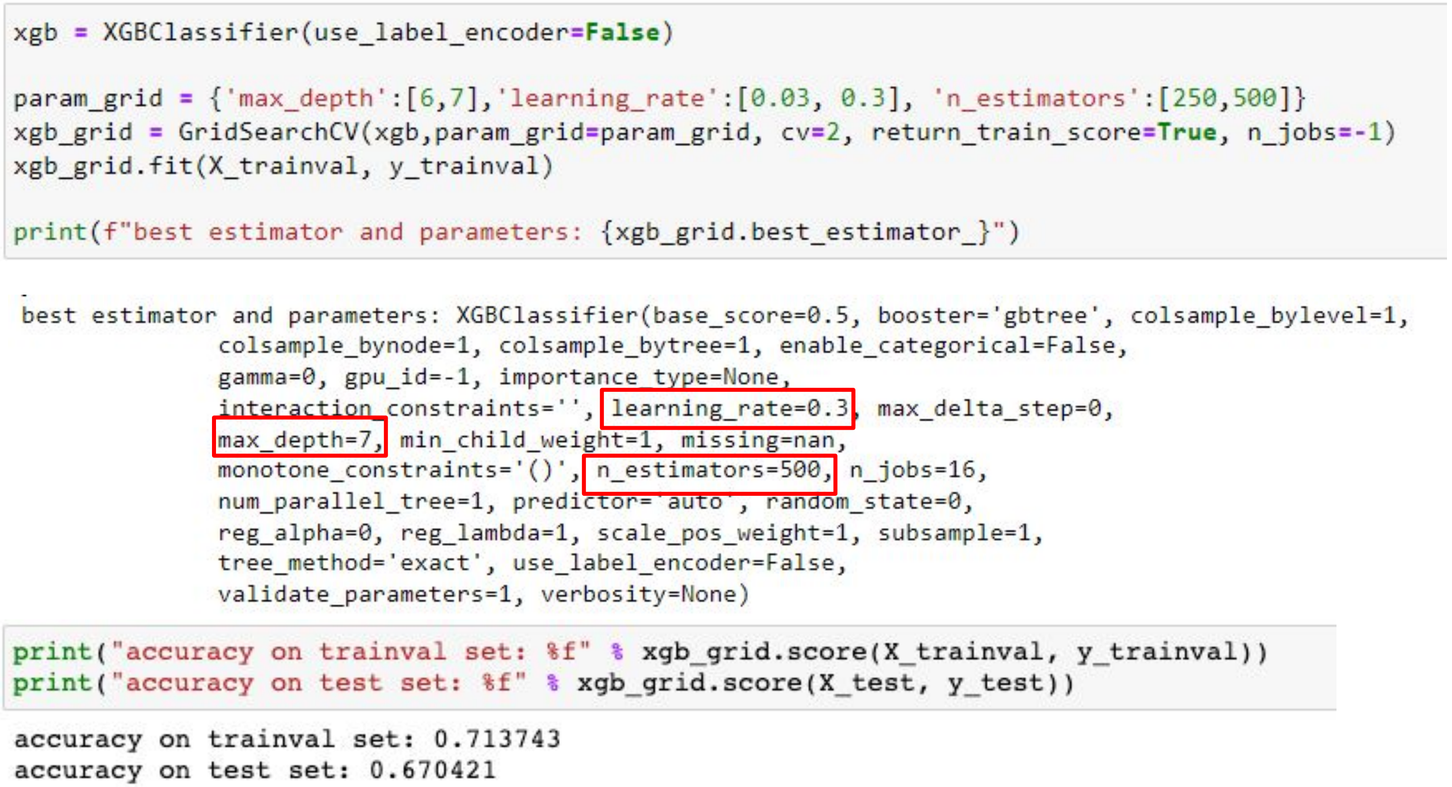


```
print("accuracy on trainval set: %f" % rf_grid.score(X_trainval, y_trainval))
print("accuracy on test set: %f" % rf_grid.score(X_test, y_test))
```

```
accuracy on trainval set: 0.639239
accuracy on test set: 0.636364
```

- Random Forest with max_depth=10, max_features=10, n_estimators=150: 64% accuracy rate on test set, comparing to KNN model.

XGBClassifier Model



```
xgb = XGBClassifier(use_label_encoder=False)

param_grid = {'max_depth':[6,7],'learning_rate':[0.03, 0.3], 'n_estimators':[250,500]}
xgb_grid = GridSearchCV(xgb,param_grid=param_grid, cv=2, return_train_score=True, n_jobs=-1)
xgb_grid.fit(X_trainval, y_trainval)

print(f"best estimator and parameters: {xgb_grid.best_estimator_}")
```

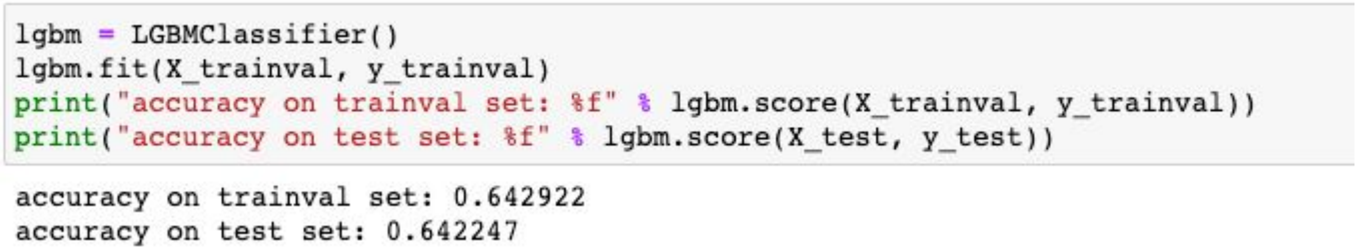
```
best estimator and parameters: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=7, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=500, n_jobs=16,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', use_label_encoder=False,
              validate_parameters=1, verbosity=None)
```

```
print("accuracy on trainval set: %f" % xgb_grid.score(X_trainval, y_trainval))
print("accuracy on test set: %f" % xgb_grid.score(X_test, y_test))
```

```
accuracy on trainval set: 0.713743
accuracy on test set: 0.670421
```

- XGBClassifier max_depth=10, max_features=10, n_estimators=150: 67% accuracy rate on test set

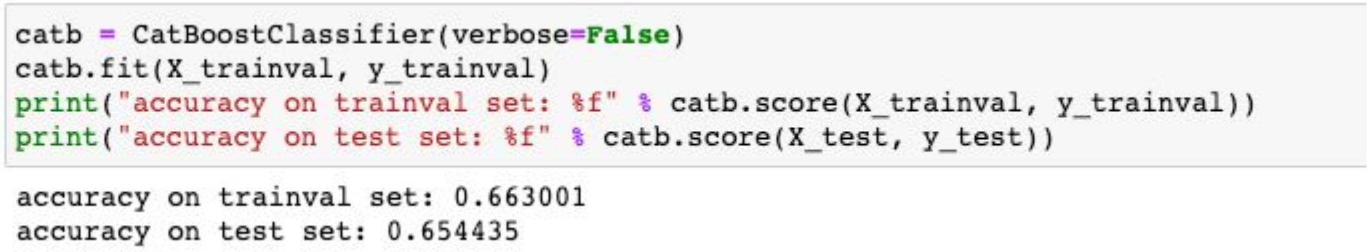
LGBMClassifier Model



```
lgbm = LGBMClassifier()
lgbm.fit(X_trainval, y_trainval)
print("accuracy on trainval set: %f" % lgbm.score(X_trainval, y_trainval))
print("accuracy on test set: %f" % lgbm.score(X_test, y_test))
```

```
accuracy on trainval set: 0.642922
accuracy on test set: 0.642247
```

CatBoostClassifier Model



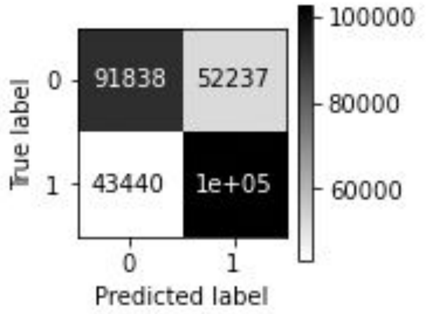
```
catb = CatBoostClassifier(verbose=False)
catb.fit(X_trainval, y_trainval)
print("accuracy on trainval set: %f" % catb.score(X_trainval, y_trainval))
print("accuracy on test set: %f" % catb.score(X_test, y_test))
```

```
accuracy on trainval set: 0.663001
accuracy on test set: 0.654435
```

Best Performance Metrics & Feature Importance

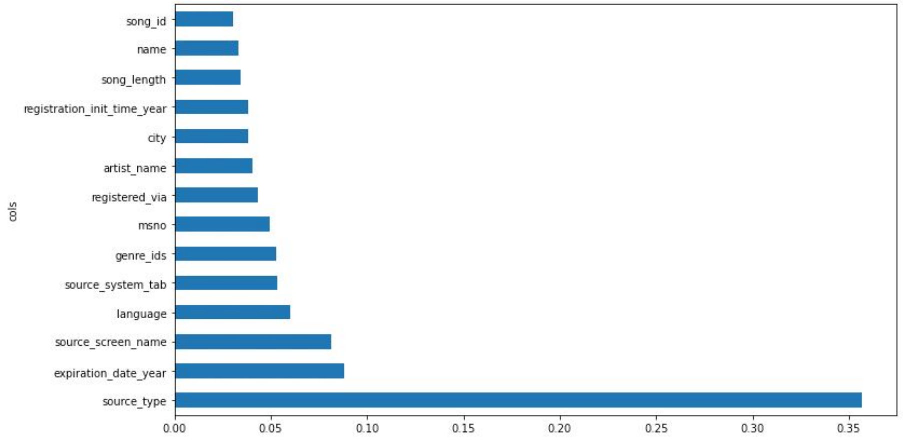
XGBoost Classification

```
XGBoost Classification
 [[ 91838  52237]
 [ 43440 102786]]
0.670421390212228
```



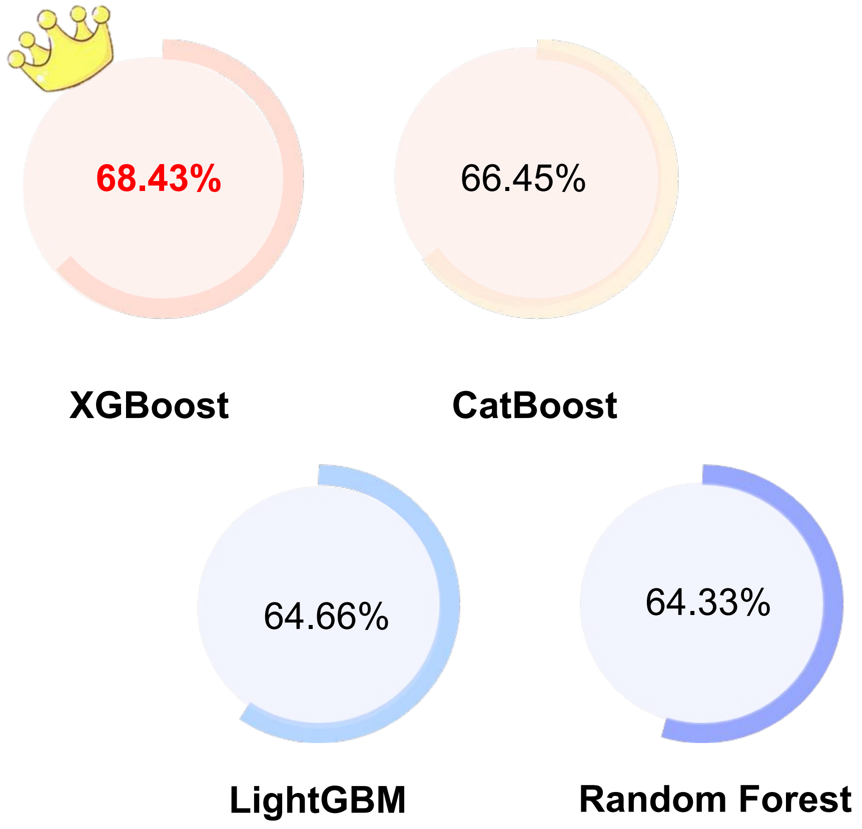
```
Accuracy score of the XGB1 is 0.670
Precision score of the XGB1 is 0.663
Recall score of the XGB1 is 0.703
```

```
                    cols    imp
4            source_type  0.357
13  expiration_date_year  0.088
3     source_screen_name  0.081
8               language  0.060
2      source_system_tab  0.053
6              genre_ids  0.053
0                   msno  0.050
11        registered_via  0.043
7            artist_name  0.040
10                  city  0.038
```



Accuracy Improvement

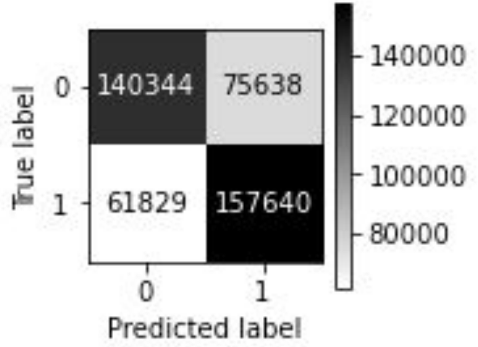
- Adjust Features
  - Remove song name
  - Remove song id
  - Membership_duration
  - Count MSNO(User Id)
- Model Selection
  - Remove KNN
- More Sample Size: 30%

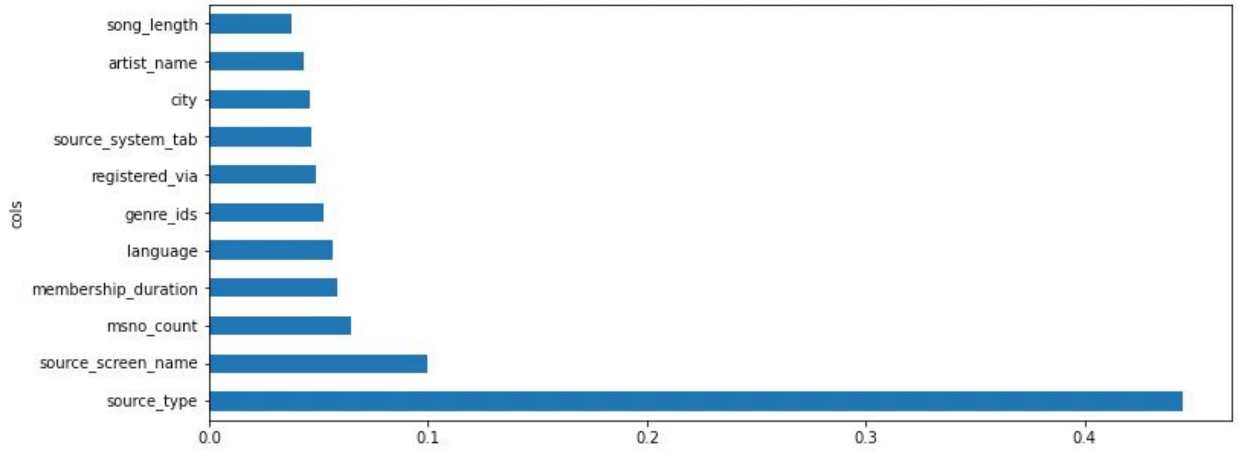


Best Performance Metrics & Feature Importance
XGBoost Classification with Different Feature Set

```
         cols                 imp
3        source_type          0.444
2        source_screen_name   0.100
0        msno_count           0.065
10       membership_duration  0.059
7        language             0.057
5        genre_ids            0.052
9        registered_via       0.049
1        source_system_tab    0.047
8        city                 0.046
6        artist_name          0.043
```

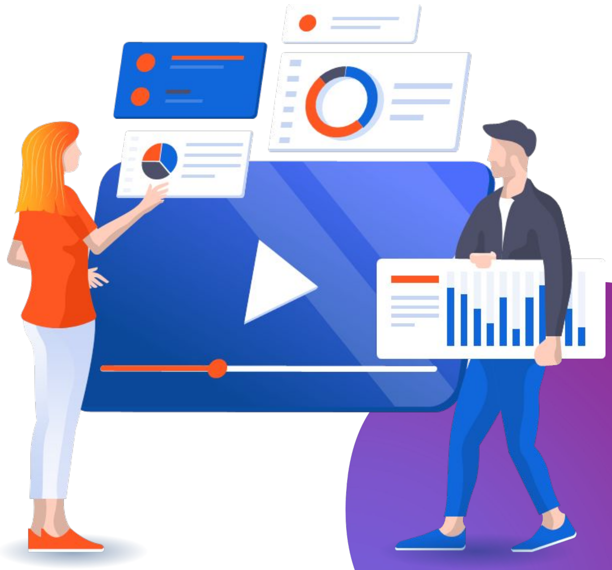
```
XGBoost Classification
 [[140344  75638]
 [ 61829 157640]]
0.6843112083793584
```





```
Accuracy score of the XGB2 is 0.684
Precision score of the XGB2 is 0.676
Recall score of the XGB2 is 0.718
```

Conclusion & Future Work



Conclusions

XGBoost - best model

- Overall, boosting models are producing better results for predicting music learning behavior. For instance, XGBoost model got **68.43%**, CatBoostClassifier got **66.45%**, and LightGBM for **64.66%.** Although all the models are with a slightly higher accuracy individually, KNN clustering model still lacking behind on performance.

Feature Engineering

- With new calculated features we are able to increase our accuracy by a huge margin. We achieved 1% accuracy improvement with XGBoost simply by converting the starting and expiring feature into one time eclipse feature and added the user count feature. More feature engineering could further help with our model performance.

Business Insights

- Recommended song list from KKbox leans heavily from same genres, artists and languages. Alternative algorithm could be utilized, for example they can launch “Discover Weekly”, the recommendation will be more diversified.

- Based on the accuracy rate and feature importance from the models, KKBOX could get deeper insights on which features they can look into more, such as source types.

- User behaviors are important features for music recommendation, but some users information, like genders and ages, are not correct or missing from the data. KKBOX could also include other kinds of user behavior features, such as if the user will listen to the song completely or not, for better music recommendation.

- The input contains text data only, and no any audio features. KKBOX currently uses a collaborative filtering based algorithm with matrix factorization and word embedding in their recommendation system. We believe new techniques could increase the accuracy on recommending music to users and lead to better results.

Future Improvement Plan

- Differ feature sets
  - Using polynomial features
  - Non essential features
  - NA values
- Differ sample rate:
  - Over 7,300,000 data
  - Due to limited memory, only portion of sample was utilized, better accuracy is expected with increased training sample size.
- Differ parameters:
  - Regularizations
  - Adjust other parameters
- New Model:
  - Neural Network

Thank you

Machine Learning
Final Project

01:03

03:17

Group7