RobustTest

September 29, 2019

[1]:
```
from sklearn import metrics, ensemble
from sklearn.model_selection import cross_validate,GridSearchCV,train_test_split
import xgboost as xgb
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

[5]:
```
print('Reading and Merging Data')
train = pd.read_csv('input/train.csv')
train = train.sample(frac=0.5)
songs = pd.read_csv('input/songs.csv')
train = pd.merge(train, songs, on='song_id', how='left')
del songs
members = pd.read_csv('input/members.csv')
train = pd.merge(train, members, on='msno', how='left')
del members
song_extra_info = pd.read_csv('input/song_extra_info.csv')
train = pd.merge(train, song_extra_info, on='song_id', how='left')
del song_extra_info
print('Reading complete')
print('Handle Nulls')
for i in train.select_dtypes(include=['object']).columns:
    train[i][train[i].isnull()] = 'unknown'
train = train.fillna(value=0)

print('Flatten the dates')
train.registration_init_time = pd.to_datetime(train.registration_init_time,␣
 ↪format='%Y%m%d', errors='ignore')
train['registration_init_time_year'] = train['registration_init_time'].dt.year
train['registration_init_time_month'] = train['registration_init_time'].dt.month
train['registration_init_time_day'] = train['registration_init_time'].dt.day
```

```
train.expiration_date = pd.to_datetime(train.expiration_date,  format='%Y%m%d',␣
 ↪errors='ignore')
train['expiration_date_year'] = train['expiration_date'].dt.year
train['expiration_date_month'] = train['expiration_date'].dt.month
train['expiration_date_day'] = train['expiration_date'].dt.day
del train['registration_init_time']
del train['expiration_date']

print('Handling the categorical Features')
categorical_feature = train.dtypes==object
categorical_cols = train.columns[categorical_feature].tolist()

print('Label Encode the values')
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train[categorical_cols] = train[categorical_cols].apply(lambda col: le.
 ↪fit_transform(col))
```

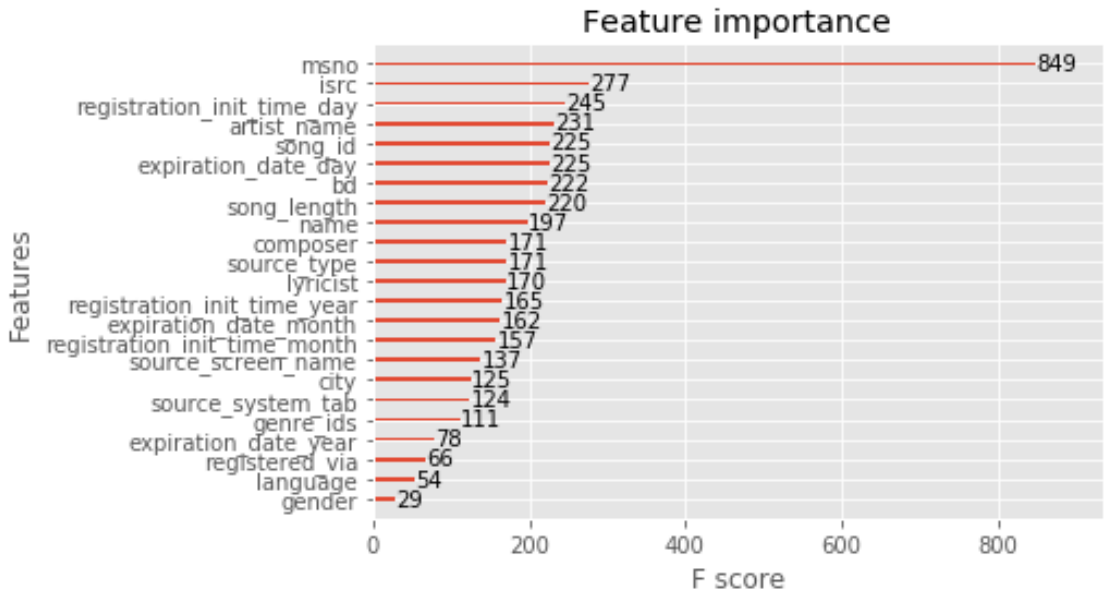
```
Reading and Merging Data
Reading complete
Handle Nulls
Flatten the dates
Handling the categorical Features
Label Encode the values
```

[9]:
```
print('Training and Calculating Accuracy')
#target = train.pop('target')
train_data, test_data, train_labels, test_labels = train_test_split(train,␣
 ↪target, test_size = 0.3)
model = xgb.XGBClassifier(max_depth=3, learning_rate=0.3, n_estimators=300,␣
 ↪n_jobs=8)
model.fit(train_data, train_labels)
predict_labels = model.predict(test_data)
print(metrics.accuracy_score(test_labels, predict_labels))
print(metrics.balanced_accuracy_score(test_labels, predict_labels))
print(metrics.average_precision_score(test_labels, predict_labels))
```

```
Training and Calculating Accuracy
0.7359555555555556
0.6639711308959847
0.7490609914956048
```

[66]:
```
xgb.plot_importance(model)
```

[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2963d7c860>
```



[98]:
```
graph_to_save = xgb.to_graphviz(model, rankdir='LR')
graph_to_save.format = 'png'
graph_to_save.render('xgb_tree')
```

[98]: 'xgb_tree.png'

[103]:
```
#=test_data.iloc[0-5]

predictest=test_data.head(n=5)
```

[104]:
```
predictest.to_csv('predictest.csv')
(predictest)
```

[104]:

	msno	song_id	source_system_tab	source_screen_name	source_type	\
431412	12101	8209	0	10	5	
1474	4792	41923	6	0	0	
81622	13243	28169	3	7	3	
127565	7959	11154	3	7	3	
384004	9401	70094	3	7	3	

	song_length	genre_ids	artist_name	composer	lyricist	...	gender	\
431412	326243.0	204	14583	21207	8046	...	2	
1474	221727.0	204	11979	23434	9664	...	2	
81622	281573.0	209	3071	22536	8735	...	0	
127565	290586.0	209	13073	27130	13715	...	0	
384004	262112.0	209	12786	17175	7229	...	1	

	registered_via	name	isrc	registration_init_time_year	\
431412	7	47012	36945	2010	

```
1474                 7  52702  45737                          2016
81622                9  37498   7481                          2005
127565               9  61717  37812                          2010
384004               3  63446  42654                          2016

        registration_init_time_month  registration_init_time_day  \
431412                             5                          29
1474                               2                           6
81622                              7                          19
127565                             5                          23
384004                            11                          12

        expiration_date_year  expiration_date_month  expiration_date_day
431412                  2017                     10                    1
1474                    2018                      2                    4
81622                   2017                      9                   30
127565                  2017                      9                   26
384004                  2016                     12                    3

[5 rows x 23 columns]
```

[105]:
```
pred = model.predict(predictest)
```

[106]:
```
print(pred)
```

```
[1 1 1 1 0]
```