CapstoneProject

September 27, 2019

[1]:
```python
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')
```

[2]:
```python
train = pd.read_csv('input/train.csv')
train = train.sample(frac=0.5)
```

[3]:
```python
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
```

[4]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 750000 entries, 0 to 749999
Data columns (total 20 columns):
msno                          750000 non-null object
song_id                       750000 non-null object
source_system_tab             747715 non-null object
source_screen_name            715634 non-null object
source_type                   748026 non-null object
target                        750000 non-null int64
```

```
song_length               749990 non-null float64
genre_ids                 739214 non-null object
artist_name               749990 non-null object
composer                  587654 non-null object
lyricist                  437735 non-null object
language                  749987 non-null float64
city                      750000 non-null int64
bd                        750000 non-null int64
gender                    452914 non-null object
registered_via            750000 non-null int64
registration_init_time    750000 non-null int64
expiration_date           750000 non-null int64
name                      749934 non-null object
isrc                      690836 non-null object
dtypes: float64(2), int64(6), object(12)
memory usage: 120.2+ MB
```

[5]:
```
train.describe()
```

[5]:
```
              target   song_length       language           city  \
count  750000.000000  7.499900e+05  749987.000000  750000.000000
mean        0.665412  2.455128e+05      18.499262       7.572321
std         0.471847  6.187241e+04      21.173175       6.585903
min         0.000000  2.716000e+03      -1.000000       1.000000
25%         0.000000  2.152020e+05       3.000000       1.000000
50%         1.000000  2.423110e+05       3.000000       5.000000
75%         1.000000  2.727180e+05      52.000000      13.000000
max         1.000000  8.679526e+06      59.000000      22.000000

                  bd  registered_via  registration_init_time  expiration_date
count  750000.000000   750000.000000            7.500000e+05     7.500000e+05
mean       17.482087        6.772887            2.012781e+07     2.017149e+07
std        21.575944        2.299888            2.980763e+04     3.891803e+03
min       -43.000000        3.000000            2.004033e+07     2.004102e+07
25%         0.000000        4.000000            2.011072e+07     2.017091e+07
50%        21.000000        7.000000            2.013102e+07     2.017093e+07
75%        28.000000        9.000000            2.015101e+07     2.017101e+07
max      1030.000000       13.000000            2.016121e+07     2.020102e+07
```

[6]:
```
train.head(10)
```

[6]:
```
                                           msno  \
0  zxyFUnD5Dxv8HMn9Ric1Qp6Q2WfvTpT270Ot6zj/7TI=
1  m1uNTJWSyEEZvnOpIEyGBNtqMrNK3Z93sV9k6NiR3cA=
2  wlFjuBSYftzA+svI+bo1jnHQvVF4iU9W44EG9TeBdZc=
3  mC4Ck1dP8BES2Je+6wKj7RXeRuNaxdIG/JMY7/7VwoQ=
4  SXoI2cPZIFNecwiZGdWd63KiVa/R3Ip5RBmWfzbfUKU=
5  2j5pmg0elytu/2Rv6Fo/vE8zyQalUduBMNHADeEZD1g=
6  yiAhcTphg5RLb/u96sNO8ksIuAXkKZgjyMl0guTNquc=
7  Jhan0r76zXl0HlhwzVrW/afa18uFvIVcgAn04284WQ8=
8  fjFTnfTIOGZZMCsFdl96bdKtNCZQQif7o4SXK5/swdQ=
9  7mQsBq3osC5B3CxbnzuYSkxXCRRVdDnGCNTpm7TdXOM=

                                        song_id source_system_tab  \
0  KhvN3eYZFeeyY+zbYKROx3qxUDOjdiPi+1kFuaVk3ac=        my library
1  /Iv1qeEEoA2ha4jkxY1Jly4AZr8+8AnbSz0OH1fsf0o=        my library
2  Kpo1j5e2JvO0iHC+a014/nRcXcrN4xBHMx2BasxEXpo=            search
3  5XyHXKU9D+weKQ/5WjCPUAA4MLwZjoStrRY9tmtDE2U=        my library
4  h45pWoMzCvsq3e3rBIuHggNB/3NG06/SIVDPEPOF1Gc=        my library
5  +Gh6hEya3f5ffyPcEJ9AR3nuRe2rFtcbCi64TG0GJKU=        my library
6  YKXNGyMdm+M370+YcJdqTDhPN1gJgBj+F5rmrdnPZT4=          discover
7  OakhL7CLirelAGEP9sYyP6fmTa8HV1mD/qVdpM6o5uE=          discover
8  /9OwQZIRPYFhTp/x0GyCR4/GIS9qDDP+cajVG76gXOA=        my library
9  J4qKkLIoW7aYACuTupHLAPZYmRp08en1AEux+GSUzdw=          discover

     source_screen_name          source_type  target  song_length  genre_ids  \
0   Local playlist more       local-playlist       1     209397.0        465
1   Local playlist more        local-library       0     156630.0        465
2            Album more                album       1     236669.0   458|1287
3   Local playlist more        local-library       0     193933.0        465
4   Local playlist more        local-library       1     194455.0        465
5   Local playlist more        local-library       1     271986.0        465
6                   NaN  song-based-playlist       0     247013.0        465
7  Online playlist more      online-playlist       0     258821.0        NaN
8   Local playlist more       local-playlist       1     297482.0        465
9  Online playlist more      online-playlist       1     212750.0  1616|1609

            artist_name                                          composer  \
0     (Khalil Fong)                                                NaN
1          (Mayday)                                                NaN
2                 Leo                                               Leo
3              PRINGLEZ                                                NaN
4          CHARLIE PUTH                                                NaN
5    (Abin Fang)                                               NaN
6             (Della)
7             G.E.M.                                                NaN
8         (Jay Chou)
9          Alan Walker  Alan Walker| Jesper Borgen| Anders Froen| Gunn...

                                            lyricist  language  city  bd  \
0                                                NaN       3.0    10  25
1                                                NaN       3.0     5   0
2                                           Leo/PNC       3.0     1  25
3                                                NaN      52.0     1   0
4                                                NaN      52.0     1   0
5                                                NaN       3.0    13  36
```

```
6                                                       3.0    15  30
7                                                NaN       3.0    13  33
8                                                       3.0     5  29
9  Alan Walker| Jesper Borgen| Anders Froen| Gunn...      52.0     1   0

  gender  registered_via  registration_init_time  expiration_date  \
0   male               9                20080220         20170917
1    NaN               3                20130105         20180126
2   male               3                20130415         20180128
3    NaN               7                20151106         20171006
4    NaN               7                20141129         20170918
5   male               3                20130919         20170913
6   male               3                20131023         20170924
7   male               7                20111006         20180625
8   male               3                20151125         20170918
9    NaN               7                20150923         20170922

                                     name          isrc
0  Nothing's gonna change my love for you  HKI490967103
1                                        TWA459962207
2                           Jam All Night  TWI451600052
3                              Love Story  GBKPL1518158
4                           One Call Away  USAT21502703
5                                       TWI430900307
6                 (Love Myself More)  TWK231609103
7                                        HKI111200214
8                                      TWK970300602
9                                   Faded  NOG841549010
```

[7]:
```python
train.isnull().sum()
```

[7]:
```
msno                           0
song_id                        0
source_system_tab           2285
source_screen_name         34366
source_type                 1974
target                         0
song_length                   10
genre_ids                  10786
artist_name                   10
composer                  162346
lyricist                  312265
language                      13
city                           0
bd                             0
gender                    297086
registered_via                 0
registration_init_time         0
```

```
expiration_date                 0
name                           66
isrc                        59164
dtype: int64
```

[8]:
```
for i in train.select_dtypes(include=['object']).columns:
    train[i][train[i].isnull()] = 'unknown'
train = train.fillna(value=0)
```

[9]:
```
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']
train.head(10)
```

[9]:
```
                                           msno  \
0  zxyFUnD5Dxv8HMn9Ric1Qp6Q2WfvTpT270Ot6zj/7TI=
1  m1uNTJWSyEEZvnOpIEyGBNtqMrNK3Z93sV9k6NiR3cA=
2  wlFjuBSYftzA+svI+bo1jnHQvVF4iU9W44EG9TeBdZc=
3  mC4Ck1dP8BES2Je+6wKj7RXeRuNaxdIG/JMY7/7VwoQ=
4  SXoI2cPZIFNecwiZGdWd63KiVa/R3Ip5RBmWfzbfUKU=
5  2j5pmg0elytu/2Rv6Fo/vE8zyQalUduBMNHADeEZD1g=
6  yiAhcTphg5RLb/u96sNO8ksIuAXkKZgjyMl0guTNquc=
7  Jhan0r76zXl0HlhwzVrW/afa18uFvIVcgAn04284WQ8=
8  fjFTnfTIOGZZMCsFdl96bdKtNCZQQif7o4SXK5/swdQ=
9  7mQsBq3osC5B3CxbnzuYSkxXCRRVdDnGCNTpm7TdXOM=

                                        song_id source_system_tab  \
0  KhvN3eYZFeeyY+zbYKROx3qxUDOjdiPi+1kFuaVk3ac=        my library
1  /Iv1qeEEoA2ha4jkxY1Jly4AZr8+8AnbSz0OH1fsf0o=        my library
2  Kpo1j5e2JvO0iHC+a014/nRcXcrN4xBHMx2BasxEXpo=            search
3  5XyHXKU9D+weKQ/5WjCPUAA4MLwZjoStrRY9tmtDE2U=        my library
4  h45pWoMzCvsq3e3rBIuHggNB/3NG06/SIVDPEPOF1Gc=        my library
5  +Gh6hEya3f5ffyPcEJ9AR3nuRe2rFtcbCi64TG0GJKU=        my library
6  YKXNGyMdm+M370+YcJdqTDhPN1gJgBj+F5rmrdnPZT4=          discover
7  OakhL7CLirelAGEP9sYyP6fmTa8HV1mD/qVdpM6o5uE=          discover
8  /9OwQZIRPYFhTp/x0GyCR4/GIS9qDDP+cajVG76gXOA=        my library
9  J4qKkLIoW7aYACuTupHLAPZYmRp08en1AEux+GSUzdw=          discover
```

```
     source_screen_name          source_type  target  song_length  genre_ids  \
0   Local playlist more       local-playlist       1     209397.0        465   
1   Local playlist more        local-library       0     156630.0        465   
2            Album more                album       1     236669.0   458|1287   
3   Local playlist more        local-library       0     193933.0        465   
4   Local playlist more        local-library       1     194455.0        465   
5   Local playlist more        local-library       1     271986.0        465   
6               unknown  song-based-playlist       0     247013.0        465   
7  Online playlist more      online-playlist       0     258821.0    unknown   
8   Local playlist more       local-playlist       1     297482.0        465   
9  Online playlist more      online-playlist       1     212750.0  1616|1609   

           artist_name                                           composer  \
0     (Khalil Fong)                                            unknown   
1          (Mayday)                                            unknown   
2                  Leo                                               Leo   
3              PRINGLEZ                                            unknown   
4          CHARLIE PUTH                                            unknown   
5    (Abin Fang)                                            unknown   
6            (Della)   
7            G.E.M.                                            unknown   
8        (Jay Chou)   
9          Alan Walker  Alan Walker| Jesper Borgen| Anders Froen| Gunn...   

   ...   gender  registered_via                                    name  \
0  ...     male               9  Nothing's gonna change my love for you   
1  ...  unknown               3   
2  ...     male               3                           Jam All Night   
3  ...  unknown               7                              Love Story   
4  ...  unknown               7                           One Call Away   
5  ...     male               3   
6  ...     male               3               (Love Myself More)   
7  ...     male               7   
8  ...     male               3   
9  ...  unknown               7                                   Faded   

           isrc registration_init_time_year  registration_init_time_month  \
0  HKI490967103                        2008                             2   
1  TWA459962207                        2013                             1   
2  TWI451600052                        2013                             4   
3  GBKPL1518158                        2015                            11   
4  USAT21502703                        2014                            11   
5  TWI430900307                        2013                             9   
6  TWK231609103                        2013                            10   
7  HKI111200214                        2011                            10   
8  TWK970300602                        2015                            11   
```

```
9  NOG841549010                        2015                             9

  registration_init_time_day expiration_date_year  expiration_date_month  \
0                          20                 2017                      9
1                           5                 2018                      1
2                          15                 2018                      1
3                           6                 2017                     10
4                          29                 2017                      9
5                          19                 2017                      9
6                          23                 2017                      9
7                           6                 2018                      6
8                          25                 2017                      9
9                          23                 2017                      9

   expiration_date_day
0                   17
1                   26
2                   28
3                    6
4                   18
5                   13
6                   24
7                   25
8                   18
9                   22

[10 rows x 24 columns]
```

[10]:
```
categorical_feature = train.dtypes==object
categorical_cols = train.columns[categorical_feature].tolist()
categorical_cols
```

[10]:
```
['msno',
 'song_id',
 'source_system_tab',
 'source_screen_name',
 'source_type',
 'genre_ids',
 'artist_name',
 'composer',
 'lyricist',
 'gender',
 'name',
 'isrc']
```

[11]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```python
train[categorical_cols] = train[categorical_cols].apply(lambda col: le.
 ↪fit_transform(col))
train[categorical_cols].head(10)
```

[11]:

```
    msno  song_id  source_system_tab  source_screen_name  source_type  \
0  18533    35429                  3                   8            4   
1  14468     2067                  3                   8            3   
2  17633    35612                  6                   0            0   
3  14526    11742                  3                   8            3   
4   8686    70341                  3                   8            3   
5   1325      480                  3                   8            3   
6  18150    56617                  0                  20            8   
7   6137    41513                  0                  11            5   
8  12670     1818                  3                   8            4   
9   2745    32974                  0                  11            5   

   genre_ids  artist_name  composer  lyricist  gender   name   isrc  
0        218        12746     22565      8796       1  26395  17340  
1        218        11663     22565      8796       2  69326  40665  
2        214         5962     13216      5027       1  18985  52064  
3        218         7691     22565      8796       2  22432  11885  
4        218         1663     22565      8796       2  27075  61370  
5        218        12751     22565      8796       1  67024  51753  
6        218        11607     23400     12585       1  49760  53283  
7        370         3626     22565      8796       1  62196  17074  
8        218        12038     23439     13914       1  59613  53800  
9         91          413       725       238       2  11477  32999  
```

[12]:

```python
#train.to_csv('train_data.csv')
train.head(10)
```

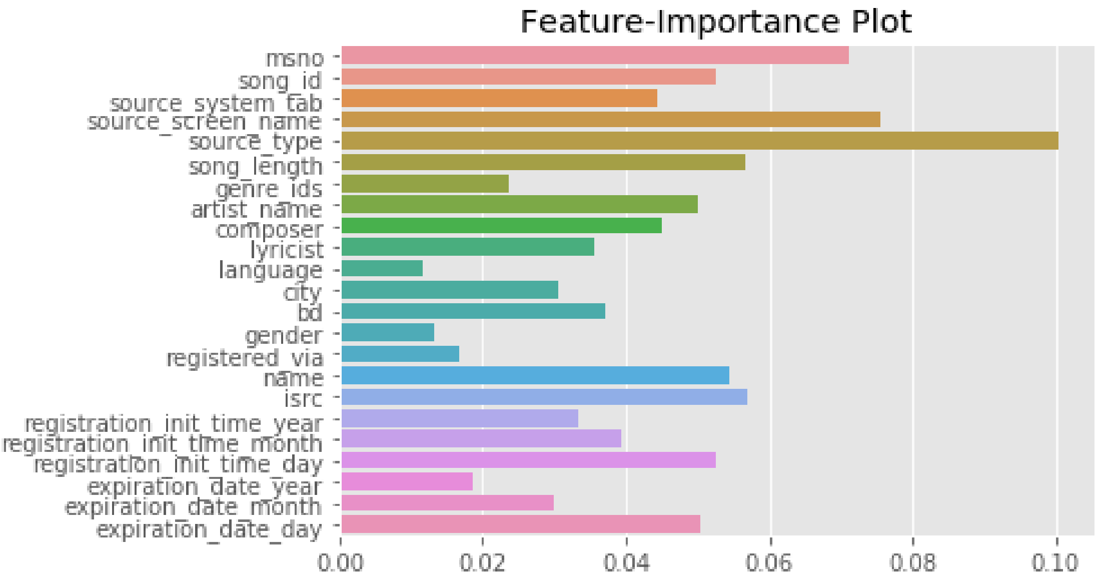
[12]:

```
    msno  song_id  source_system_tab  source_screen_name  source_type  target  \
0  18533    35429                  3                   8            4       1   
1  14468     2067                  3                   8            3       0   
2  17633    35612                  6                   0            0       1   
3  14526    11742                  3                   8            3       0   
4   8686    70341                  3                   8            3       1   
5   1325      480                  3                   8            3       1   
6  18150    56617                  0                  20            8       0   
7   6137    41513                  0                  11            5       0   
8  12670     1818                  3                   8            4       1   
9   2745    32974                  0                  11            5       1   

   song_length  genre_ids  artist_name  composer  ...  gender  registered_via  \
0     209397.0        218        12746     22565  ...       1               9   
1     156630.0        218        11663     22565  ...       2               3   
2     236669.0        214         5962     13216  ...       1               3   
3     193933.0        218         7691     22565  ...       2               7   
```

```
4     194455.0        218         1663      22565  ...       2               7
5     271986.0        218        12751      22565  ...       1               3
6     247013.0        218        11607      23400  ...       1               3
7     258821.0        370         3626      22565  ...       1               7
8     297482.0        218        12038      23439  ...       1               3
9     212750.0         91          413        725  ...       2               7

    name   isrc  registration_init_time_year  registration_init_time_month  \
0  26395  17340                         2008                             2
1  69326  40665                         2013                             1
2  18985  52064                         2013                             4
3  22432  11885                         2015                            11
4  27075  61370                         2014                            11
5  67024  51753                         2013                             9
6  49760  53283                         2013                            10
7  62196  17074                         2011                            10
8  59613  53800                         2015                            11
9  11477  32999                         2015                             9

   registration_init_time_day  expiration_date_year  expiration_date_month  \
0                          20                  2017                      9
1                           5                  2018                      1
2                          15                  2018                      1
3                           6                  2017                     10
4                          29                  2017                      9
5                          19                  2017                      9
6                          23                  2017                      9
7                           6                  2018                      6
8                          25                  2017                      9
9                          23                  2017                      9

   expiration_date_day
0                   17
1                   26
2                   28
3                    6
4                   18
5                   13
6                   24
7                   25
8                   18
9                   22

[10 rows x 24 columns]
```

[13]:
```
X = train[train.columns[train.columns != 'target']]
y = train.target
```

```
model = ensemble.RandomForestClassifier(n_estimators=100, max_depth=25)
model.fit(X, y)

features = train.columns[train.columns != 'target']
importance_values = model.feature_importances_

sns.barplot(x = importance_values, y =features )
plt.title('Feature-Importance Plot')
plt.show()
```

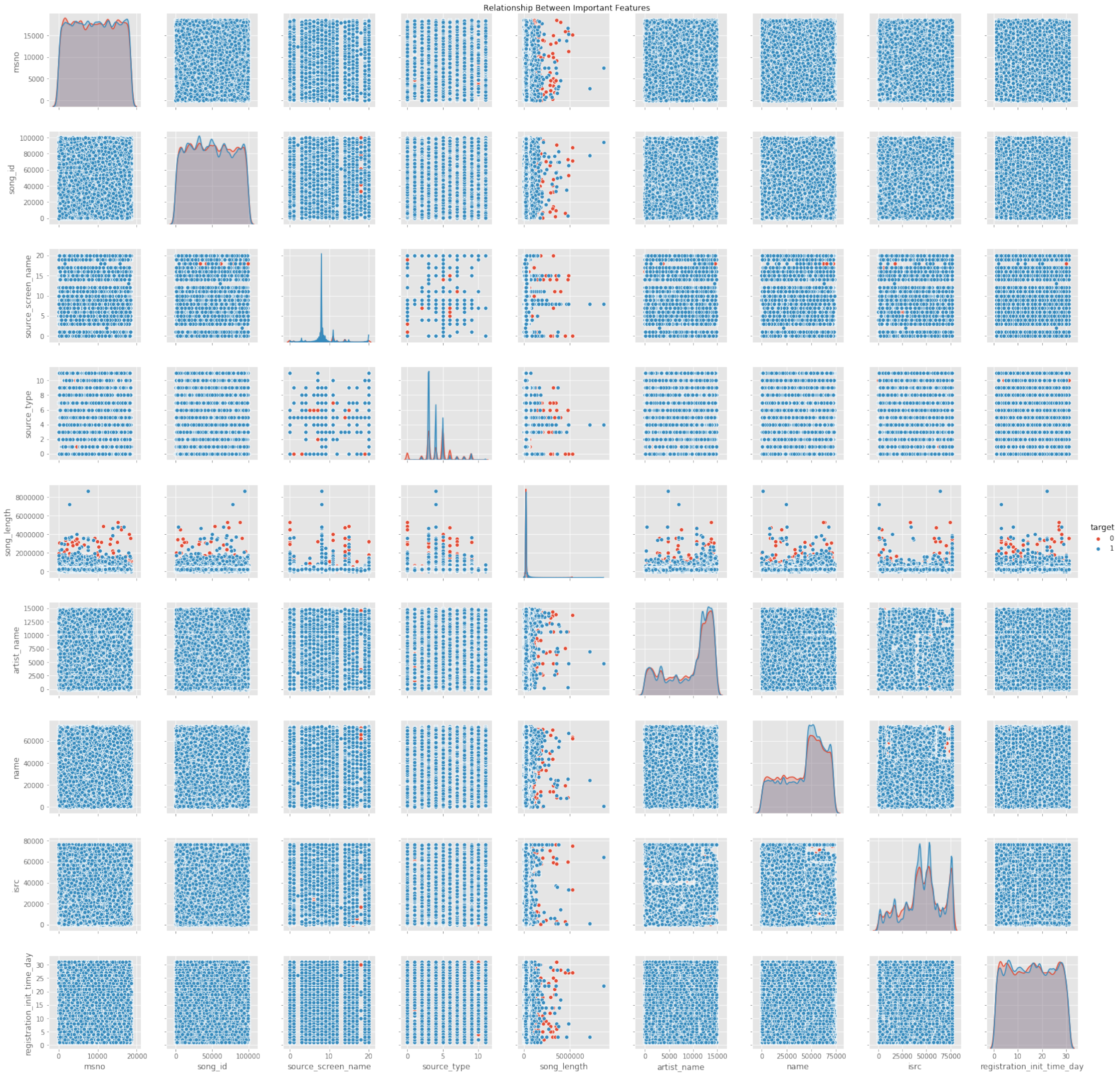


[14]:
```
imporant_feat = pd.concat([(features.to_series().reset_index(drop=True)), pd.
↪DataFrame(importance_values)], axis=1)
imporant_feat.columns = ['features', 'importance_values']
imporant_feat[importance_values>0.05]
```

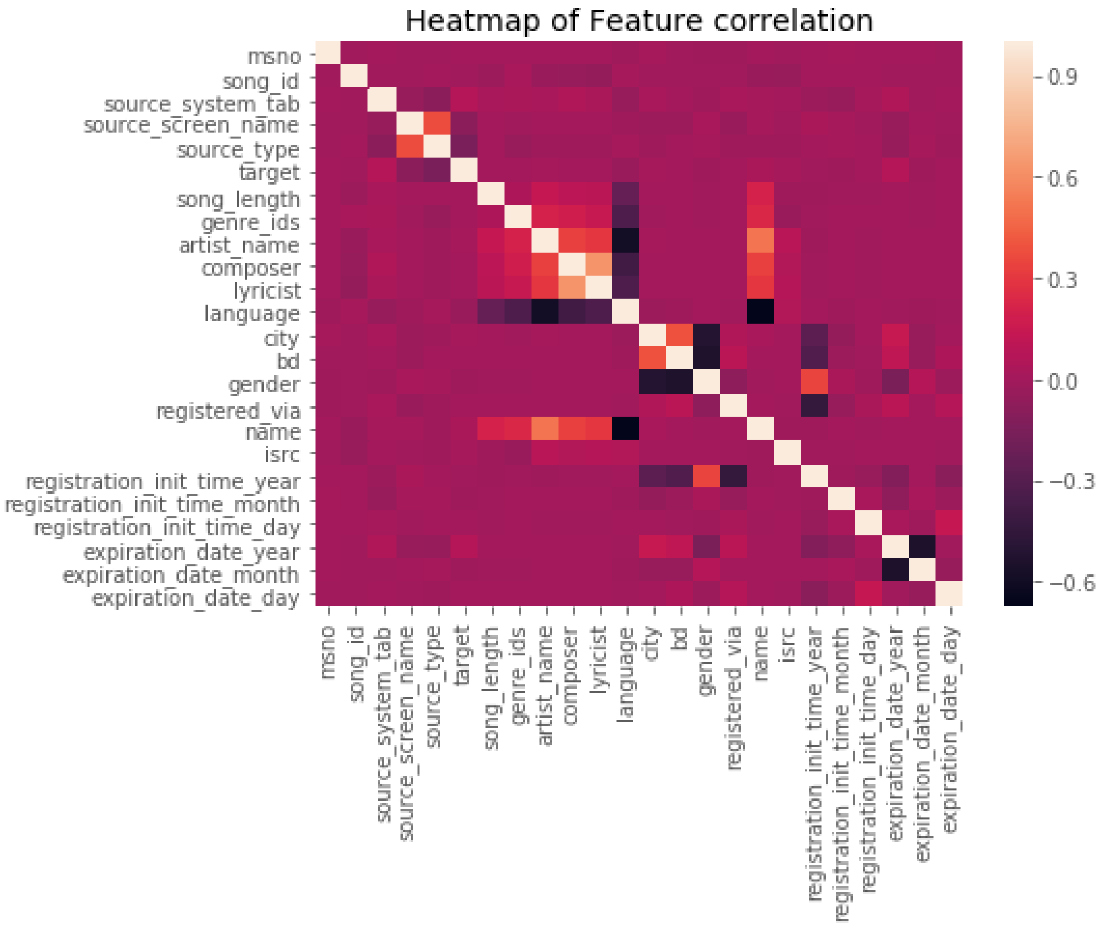
[14]:
```
                   features  importance_values
0                      msno           0.071027
1                   song_id           0.052548
3        source_screen_name           0.075662
4               source_type           0.100382
5               song_length           0.056540
7               artist_name           0.050155
15                     name           0.054579
16                     isrc           0.056940
19  registration_init_time_day        0.052613
22        expiration_date_day         0.050430
```

[15]:
```
# To have a look at relationship between the important Features >0.05
imporant_features = ['msno', 'song_id', 'source_screen_name','source_type',␣
 ↪'song_length','artist_name','name', 'isrc', 'registration_init_time_day']
pair_plot_imp = sns.pairplot(train, vars=imporant_features, hue='target')
pair_plot_imp.fig.suptitle("Relationship Between Important Features", y=1)
```

[15]:
```
Text(0.5, 1, 'Relationship Between Important Features')
```



[16]:
```
# Heatmap of the Feature correlation
plt.figure(figsize=[7,5])
sns.heatmap(train.corr())
plt.title('Heatmap of Feature correlation')
plt.show()
```



[17]:
```
train.columns
#train.count(axis='columns')
```

[17]:
```
Index(['msno', 'song_id', 'source_system_tab', 'source_screen_name',
       'source_type', 'target', 'song_length', 'genre_ids', 'artist_name',
       'composer', 'lyricist', 'language', 'city', 'bd', 'gender',
       'registered_via', 'name', 'isrc', 'registration_init_time_year',
       'registration_init_time_month', 'registration_init_time_day',
       'expiration_date_year', 'expiration_date_month', 'expiration_date_day'],
      dtype='object')
```

[18]:
```
target = train.pop('target')
```

[19]:
```
train_data, test_data, train_labels, test_labels = train_test_split(train,␣
 ↪target, test_size = 0.3)
```

[20]:
```
from sklearn.metrics import accuracy_score, log_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

```python
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn import metrics

classifiers = [
    KNeighborsClassifier(3),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    GaussianNB(),
    LinearDiscriminantAnalysis(),
    QuadraticDiscriminantAnalysis()]

for clf in classifiers:
    print("="*30)
    name = clf.__class__.__name__
    print(name)

    clf.fit(train_data, train_labels)
    test_predictions = clf.predict(test_data)
    print(accuracy_score(test_labels, test_predictions))

print("="*30)
```

```
==============================
KNeighborsClassifier
0.6174977777777778
==============================
DecisionTreeClassifier
0.6889688888888889
==============================
RandomForestClassifier
0.7469155555555556
==============================
AdaBoostClassifier
0.7149555555555556
==============================
GradientBoostingClassifier
0.7227822222222222
==============================
GaussianNB
0.6654755555555556
==============================
LinearDiscriminantAnalysis
0.6736977777777777
==============================
QuadraticDiscriminantAnalysis
0.6808177777777777
==============================
```

[21]:

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from mlxtend.classifier import StackingCVClassifier
import numpy as np
import warnings

warnings.simplefilter('ignore')

RANDOM_SEED = 42

first_classifier = GradientBoostingClassifier()
second_classifier = RandomForestClassifier(random_state=RANDOM_SEED)

logist_regression = LogisticRegression()

classifier_stack = StackingCVClassifier(classifiers=[first_classifier,␣
 ↪second_classifier], meta_classifier=logist_regression,␣
 ↪random_state=RANDOM_SEED)

print('Stacking Classifiers')

for clf, label in zip([first_classifier, second_classifier, classifier_stack],
                      ['GradientBoostingClassifier', 'RandomForestClassifier',␣
 ↪'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, train_data,␣
 ↪train_labels,cv=3, scoring='accuracy')
    print("Accuracy: %0.2f [%s]" % (scores.mean(), label))
```

```
Stacking Classifiers
Accuracy: 0.72 [GradientBoostingClassifier]
Accuracy: 0.74 [RandomForestClassifier]
Accuracy: 0.73 [StackingClassifier]
```

[22]:

```
import lightgbm as lgb
from sklearn.metrics import accuracy_score

d_train = lgb.Dataset(train_data, label= train_labels)
```

```python
params = {}
params['learning_rate']= 0.1
params['max_depth']=10
clf= lgb.train(params, d_train)
y_pred = clf.predict(test_data)
y_pred = np.where(y_pred > 0.49, 1, 0)

print(accuracy_score(y_pred, test_labels))
```

```
0.7370488888888889
```

[23]:

```python
model = xgb.XGBClassifier(learning_rate=0.1, max_depth=10, n_estimators=100)
model.fit(train_data, train_labels)
predict_labels = model.predict(test_data)
print(metrics.accuracy_score(test_labels, predict_labels))
```

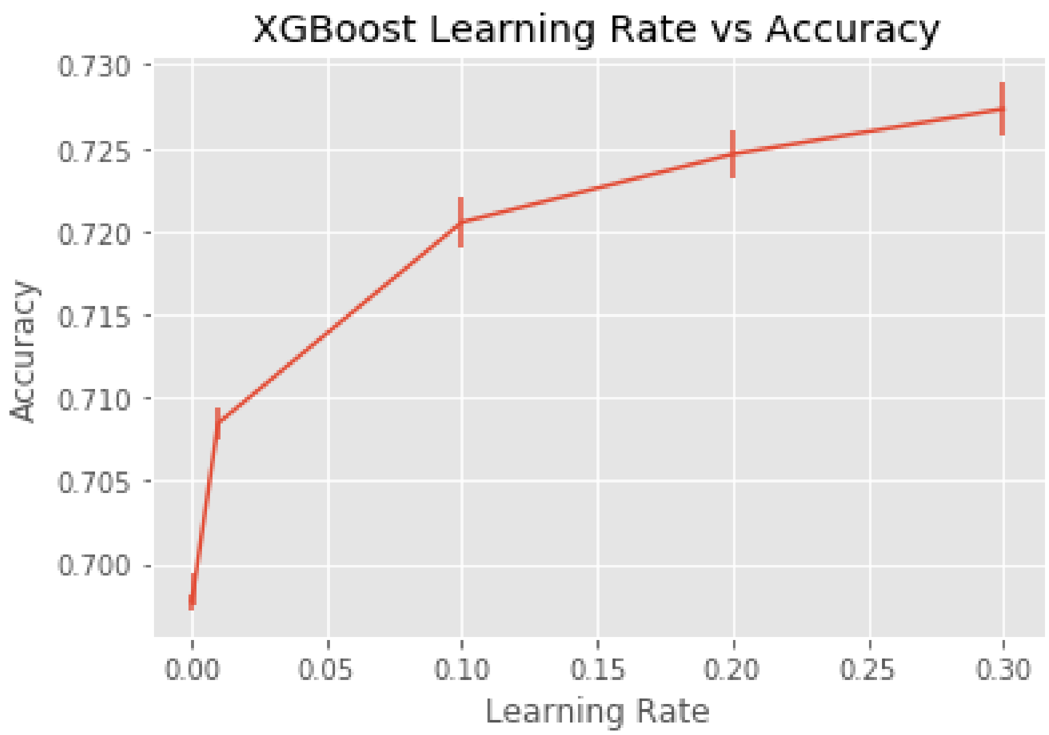
```
0.7647022222222222
```

[24]:

```python
# Tuning the Learning Rate for Accuracy
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

model = xgb.XGBClassifier()
learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
param_grid = dict(learning_rate=learning_rate)

grid_search = GridSearchCV(model, param_grid, scoring="accuracy", n_jobs=-1)
grid_result = grid_search.fit(train_data, train_labels)

print("Best: %f accuracy %s" % (grid_result.best_score_, grid_result.
 ↪best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

plt.errorbar(learning_rate, means, yerr=stds)
plt.title("XGBoost Learning Rate vs Accuracy")
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy')
plt.show()
```

```
Best: 0.727335 accuracy {'learning_rate': 0.3}
0.697745 (0.000511) with: {'learning_rate': 0.0001}
0.698490 (0.000935) with: {'learning_rate': 0.001}
```

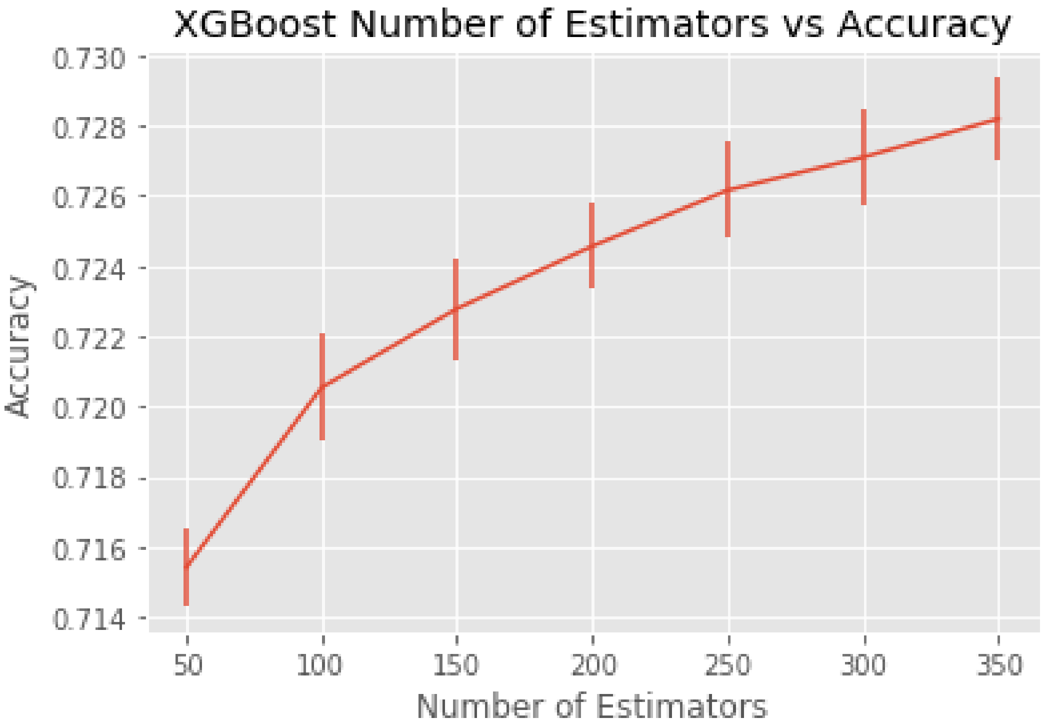
```
0.708467 (0.000938) with: {'learning_rate': 0.01}
0.720539 (0.001524) with: {'learning_rate': 0.1}
0.724629 (0.001386) with: {'learning_rate': 0.2}
0.727335 (0.001579) with: {'learning_rate': 0.3}
```



[25]:
```
# Tuning the Number of Decision Trees for Accuracy
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

model = xgb.XGBClassifier()
n_estimators = range(50, 400, 50)
param_grid = dict(n_estimators=n_estimators)

grid_search = GridSearchCV(model, param_grid, scoring="accuracy", n_jobs=-1)
grid_result = grid_search.fit(train_data, train_labels)

print("Best: %f accuracy %s" % (grid_result.best_score_, grid_result.
 ↪best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
plt.errorbar(n_estimators, means, yerr=stds)
plt.title("XGBoost Number of Estimators vs Accuracy")
plt.xlabel('Number of Estimators')
plt.ylabel('Accuracy')
plt.show()
```

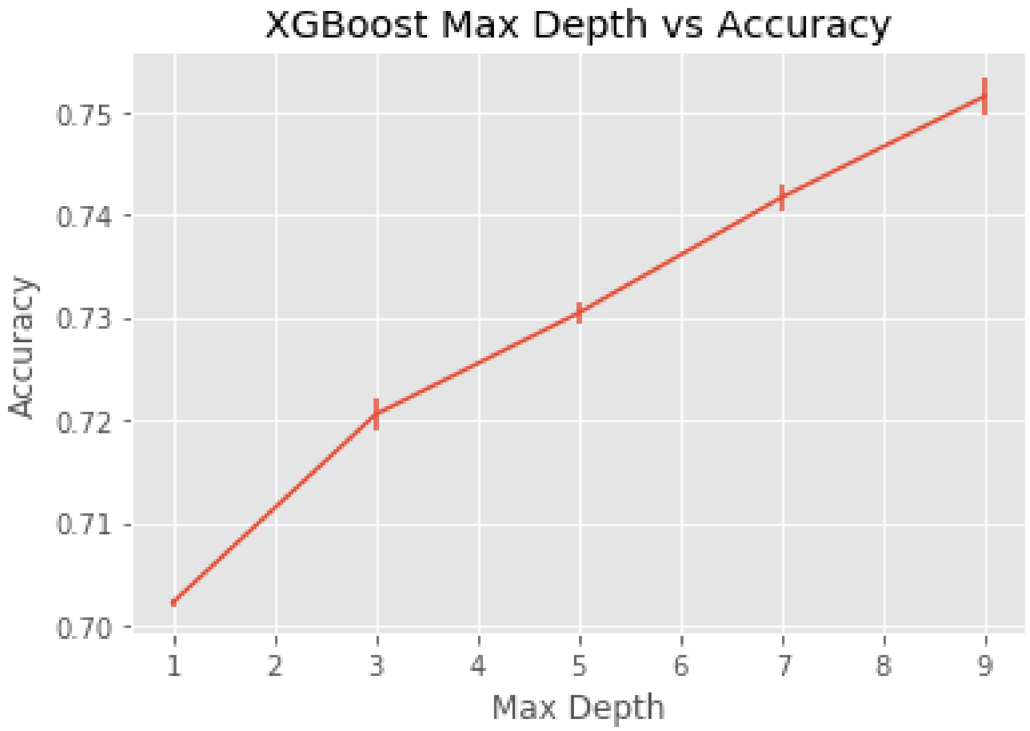
```
Best: 0.728194 accuracy {'n_estimators': 350}
0.715415 (0.001089) with: {'n_estimators': 50}
0.720539 (0.001524) with: {'n_estimators': 100}
0.722787 (0.001432) with: {'n_estimators': 150}
0.724566 (0.001215) with: {'n_estimators': 200}
0.726162 (0.001379) with: {'n_estimators': 250}
0.727101 (0.001363) with: {'n_estimators': 300}
0.728194 (0.001168) with: {'n_estimators': 350}
```



[26]:
```
# Tuning the Size of Decision Trees for Accuracy
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

model = xgb.XGBClassifier()
max_depth = range(1, 11, 2)
param_grid = dict(max_depth=max_depth)

grid_search = GridSearchCV(model, param_grid, scoring="accuracy", n_jobs=-1)
```

```
grid_result = grid_search.fit(train_data, train_labels)

print("Best: %f accuracy %s" % (grid_result.best_score_, grid_result.
 ↪best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

plt.errorbar(max_depth, means, yerr=stds)
plt.title("XGBoost Max Depth vs Accuracy")
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.show()
```

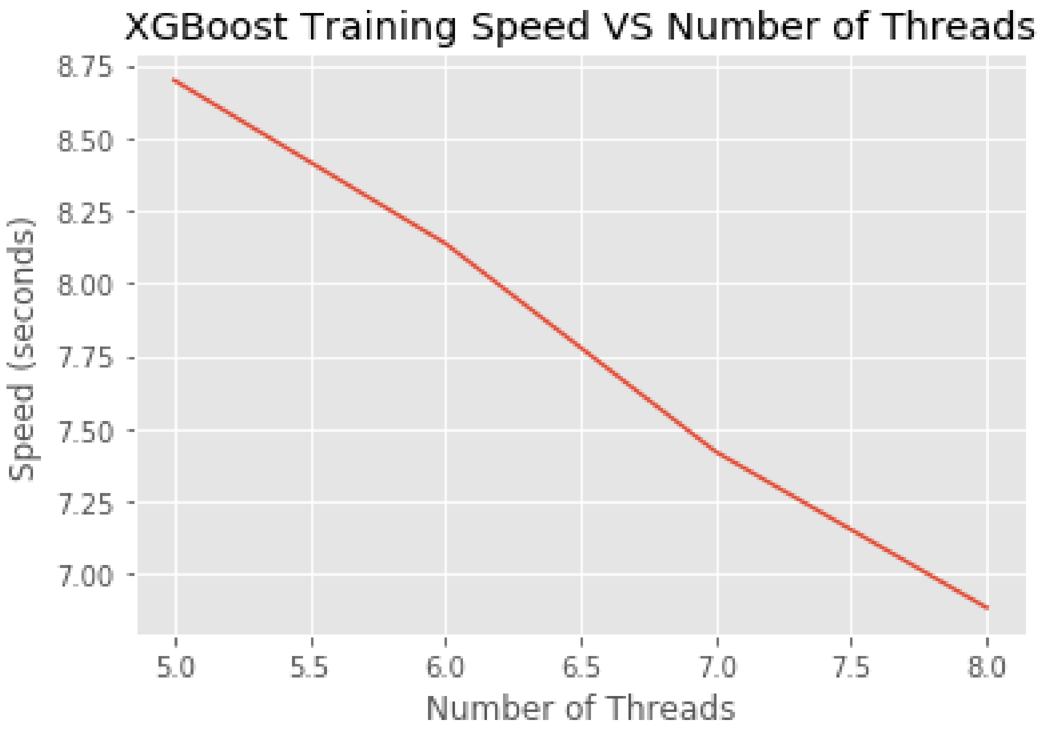
```
Best: 0.751524 accuracy {'max_depth': 9}
0.702236 (0.000403) with: {'max_depth': 1}
0.720539 (0.001524) with: {'max_depth': 3}
0.730425 (0.001129) with: {'max_depth': 5}
0.741688 (0.001292) with: {'max_depth': 7}
0.751524 (0.001729) with: {'max_depth': 9}
```



[27]:

```python
# Plotting training time with Number of threads
import matplotlib.pyplot as plt
import time

results = []
num_jobs = [5, 6, 7,8]
for n in num_jobs:
    start = time.time()
    model = xgb.XGBClassifier(n_jobs=n)
    model.fit(train_data, train_labels)
    elapsed = time.time() - start
    print(n, elapsed)
    results.append(elapsed)

plt.plot(num_jobs, results)
plt.ylabel('Speed (seconds)')
plt.xlabel('Number of Threads')
plt.title('XGBoost Training Speed VS Number of Threads')
plt.show()
```

```
5 8.697835922241211
6 8.137639284133911
7 7.420692443847656
8 6.884056091308594
```



[28]:
```
model = xgb.XGBClassifier(max_depth=20, 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))
```

```
0.7819733333333333
```

[29]:
```
#model = xgb.XGBClassifier(max_depth=20, learning_rate=0.3, min_child_weight=3,␣
 ↪n_estimators=100, scale_pos_weight=1, seed=1)
#model.fit(train_data, train_labels, eval_metric='auc', eval_set=[(test_data,␣
 ↪test_labels)], early_stopping_rounds=100)
```

[30]:
```
model = xgb.XGBClassifier(learning_rate=0.1, max_depth=15, min_child_weight=5,␣
 ↪n_estimators=300)
model.fit(train_data, train_labels)
```

[30]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=15,
              min_child_weight=5, missing=None, n_estimators=300, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

[31]:
```
predict_labels = model.predict(test_data)

print(metrics.classification_report(test_labels, predict_labels))
print(metrics.accuracy_score(test_labels, predict_labels))
print(metrics.roc_auc_score(test_labels, predict_labels))
```

```
              precision    recall  f1-score   support

           0       0.73      0.57      0.64     74929
           1       0.81      0.90      0.85    150071

    accuracy                           0.79    225000
   macro avg       0.77      0.73      0.74    225000
weighted avg       0.78      0.79      0.78    225000

0.7872
0.7319791703101244
```

[32]:
```
from xgboost import plot_tree
import matplotlib.pyplot as plt

plot_tree(model)
```

```
plt.show()
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.0817367 to fit
```

[33]:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, MaxPooling1D
from keras.utils.vis_utils import model_to_dot
from IPython.display import SVG
model = Sequential()
model.add(Dense(64, input_dim=23, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='softmax'))

model.summary()
#SVG(model_to_dot(model).create(prog='dot', format='svg'))
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0927 02:28:26.555955 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0927 02:28:26.570237 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0927 02:28:26.574177 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0927 02:28:26.584730 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:133: The name
```

```
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

W0927 02:28:26.589961 140560318105408 deprecation.py:506] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                1536
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
Total params: 84,097
Trainable params: 84,097
Non-trainable params: 0
_________________________________________________________________
```

[34]:

```
#model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
model.compile(loss='binary_crossentropy', optimizer='rmsprop',␣
 ↪metrics=['accuracy'])
```

```
W0927 02:28:26.676219 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/optimizers.py:790: The name tf.train.Optimizer is deprecated.
```

```
Please use tf.compat.v1.train.Optimizer instead.

W0927 02:28:26.691033 140560318105408 deprecation_wrapper.py:119] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3376: The name tf.log is
deprecated. Please use tf.math.log instead.

W0927 02:28:26.695823 140560318105408 deprecation.py:323] From
/home/deeplearning/anaconda3/envs/udacityml/lib/python3.7/site-
packages/tensorflow/python/ops/nn_impl.py:180:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

[35]:

```python
from keras.callbacks import EarlyStopping
early_stopping_monitor = EarlyStopping(patience=3)
print(train_data.size)
train_data.shape
```

```
12075000
```

[35]:

```
(525000, 23)
```

[36]:

```python
model.fit(train_data, train_labels, epochs=25, batch_size=1000,␣
 ↪callbacks=[early_stopping_monitor])
```

```
Epoch 1/25
525000/525000 [==============================] - 4s 7us/step - loss: 5.3449 -
acc: 0.6647
Epoch 2/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 3/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 4/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 5/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 6/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 7/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
```

```
acc: 0.6647
Epoch 8/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 9/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 10/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 11/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 12/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 13/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 14/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 15/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 16/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 17/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 18/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 19/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 20/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 21/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 22/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 23/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
```

```
acc: 0.6647
Epoch 24/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
Epoch 25/25
525000/525000 [==============================] - 3s 6us/step - loss: 5.3449 -
acc: 0.6647
```

[36]:
```
<keras.callbacks.History at 0x7fd6610bc5c0>
```

[37]:
```python
accuracy = model.evaluate(test_data, test_labels)
```

```
225000/225000 [==============================] - 2s 10us/step
```

[38]:
```python
#print('Accuracy: %.2f' % (accuracy*100))
print(model.metrics_names)
accuracy
```

```
['loss', 'acc']
```

[38]:
```
[5.309097780710856, 0.6669822222222223]
```