CapstoneProject

September 29, 2019

[47]:
```
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')
```

[48]:
```
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
```

[49]:
```
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       747683 non-null object
source_screen_name      715519 non-null object
source_type             748001 non-null object
```

```
target                    750000 non-null int64
song_length               749983 non-null float64
genre_ids                 739137 non-null object
artist_name               749983 non-null object
composer                  587854 non-null object
lyricist                  437784 non-null object
language                  749981 non-null float64
city                      750000 non-null int64
bd                        750000 non-null int64
gender                    453406 non-null object
registered_via            750000 non-null int64
registration_init_time    750000 non-null int64
expiration_date           750000 non-null int64
name                      749928 non-null object
isrc                      690939 non-null object
dtypes: float64(2), int64(6), object(12)
memory usage: 120.2+ MB
```

[65]:
```
train.describe()
```

[65]:
```
            target  song_length     language         city           bd  \
count  750000.0...  7.499830...  749981.0...  750000.0...  750000.0...
mean       0.666271  2.455737...    18.492366     7.576123    17.501940
std        0.471545  6.115412...    21.175738     6.587155    21.305039
min        0.000000  2.716000...    -1.000000     1.000000   -43.000000
25%        0.000000  2.151960...     3.000000     1.000000     0.000000
50%        1.000000  2.424160...     3.000000     5.000000    21.000000
75%        1.000000  2.727180...    52.000000    13.000000    28.000000
max        1.000000  7.371499...    59.000000    22.000000  1030.000000

       registered_via  registration_init_time  expiration_date
count  750000.0...     7.500000...             7.500000...
mean       6.775328    2.012775...             2.017149...
std        2.298958    2.983396...             3.889016...
min        3.000000    2.004033...             2.004102...
25%        4.000000    2.011071...             2.017091...
50%        7.000000    2.013102...             2.017093...
75%        9.000000    2.015101...             2.017101...
max       13.000000    2.016121...             2.020102...
```

[64]:
```
train.describe().to_html('dataframe_head.html')
```

[66]:
```
train.head(5)
```

[66]:
```
          msno      song_id source_system_tab source_screen_name  source_type  \
0  ZQbQiWQg...  a1iJZwnK...   my library       Local pl...         local-pl...
1  OtYvVpDO...  IIPVkO6E...      explore           Explore         online-p...
2  un+M8wa2...  uWfF+7Tl...   my library       Local pl...         local-li...
3  FYdHbSh9...  EUm43qqC...   my library       Local pl...         local-pl...
```

```
4  pGB6bKP/...  Tlo3ydJu...     discover               NaN        song-bas...

   target  song_length genre_ids  artist_name     composer lyricist  language  \
0       0     238132.0       465   (Jay...                          3.0
1       0     290168.0       458   (Amb...                          3.0
2       1     254755.0       458        A-Lin      Eric  /         3.0
3       1     168228.0       947           Digital ...      NaN      -1.0
4       0     360176.0       465   (Tom...           NaN      NaN       3.0

   city  bd  gender  registered_via  registration_init_time  expiration_date  \
0     4  26  female               3                20160129         20170907
1     5  31    male               9                20100520         20171005
2    13  24    male               9                20150129         20170907
3     1   0     NaN               7                20121220         20170915
4     1   0     NaN               7                20161111         20170910

          name         isrc
0            TWK97040...
1          TWR03160...
2   (Fl...  TWA47160...
3                  NaN
4             TWB51970...
```

[63]:
```
pd.set_option('display.max_colwidth',12)
train.head(5).to_html('headdata.html')
```

[24]:
```
train.isnull().sum()
```

[24]:
```
msno                        0
song_id                     0
source_system_tab        2386
source_screen_name      34346
source_type              2056
target                      0
song_length                10
genre_ids               10800
artist_name                10
composer               162049
lyricist               311502
language                   13
city                        0
bd                          0
gender                 296430
registered_via              0
registration_init_time      0
expiration_date             0
name                       61
isrc                    59148
```

```
dtype: int64
```

[25]:
```
for i in train.select_dtypes(include=['object']).columns:
    train[i][train[i].isnull()] = 'unknown'
train = train.fillna(value=0)
```

[26]:
```
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)
```

[26]:

	msno	song_id	source_system_tab	source_screen_name	source_type	target	song_length	genre_ids
0	9//vyA8a6noe+FZNkyFOMsTBPb5K9TzfDZmxzyIlFpM=	JM8COkiujGseUFvAB43PPmzEtJCOiflFIiRj8H1t6Ms=	my library	Local playlist more	local-playlist	1	282958.0	465
1	ENLrrJOF+atyZSSi7kZTbNjD83wSAGps8uhYjfVnKgo=	msy6vSQ15p6RVJfm/bSgcDwd3YSLp445Pdhzw6CEZRM=	my library					
2	2TZvSesNpTmKloHuNgKGTqRjHitgoJjC9wHK+gqGaxs=	5GK7VfnddYFs21pMCU/lFqJXHtxW5Thx1clB5XBFLsM=	my library					
3	xDcuTUZQhugCvOAmEhtpbtMreJ9oHmDMqREkC1iqZGU=	dRP9ON1AYPdVv1eOZOast5iQLXwl1blkwK5kj76M6TQ=	my library					
4	KfcZSp62/7pK+eK++Wa6IzGFq6z5w/UqvBBmsHjoX3c=	6Bwu//FCFLxNnZYtZuwA3i7tPJCuLtoRNk56Q8IfGpE=	my library					
5	kXkMzhyacFrtSI922IBs0BcmNxKedV4+837lljPNTQA=	E/b+7QWv/HguM/u4uQXY1/2jeiFsxst6FRK2shsNHiU=	discover					
6	jYiR2IiN1N1+5SOrTAHOiEWC1QQtEmGJOvkwhvJRPOI=	Z209vommZHaIoJxBiaFA9hNJbNcTTTXSgSIS58J8r+w=	listen with					
7	eOzOKfyF/jZ2hI84e54q1j0kd+TrF7mPe+ppQiz/Ues=	HWydiZOUWDm26gVTl5h7+Nj7L+SJ/VpNmL6yDSiiHW8=	my library					
8	iaAlXCGCt1oNrW6MR7X2K4gsPWHbmdROPne9w6cpo6k=	x+wwBWqNXLShElFWaLs1ahI15dN4cl4ShC1/Wm4PsH4=	discover					
9	OwIhAhyIWUfyuY8z8wfHd3TLBmaEqbx2fmfUweFLtNQ=	p9ht4E3BUSoGNSQDLqp+CNgZsyajg7FtGWjBSdWNRFQ=	my library					

```
1   Local playlist more   local-playlist       1     194168.0       465
2   Local playlist more    local-library       1     301662.0       465
3  Online playlist more  online-playlist       0     225593.0       465
4   Local playlist more   local-playlist       1     195709.0      2022
5      Discover Feature  online-playlist       1     303438.0       458
6   Others profile more      listen-with       1     241325.0       458
7   Local playlist more   local-playlist       1     276363.0       465
8  Online playlist more  online-playlist       1     273345.0       465
9   Local playlist more    local-library       0     249614.0      1609

          artist_name    composer  ...   gender  registered_via  \
0           (Hebe)     unknown  ...   female               9
1        (R-chord)     R-chord  ...   female               7
2      (Phil Chang)    Zhang Yu  ...     male               9
3                       unknown  ...   female               9
4         ONE OK ROCK     unknown  ...  unknown                 9
5   (Jonathan Lee)           ...   female               9
6        (CoCo Lee)  Adia Chang  ...     male               3
7                     unknown  ...   female                4
8                           ...     male               4
9        Maxi Kingdom     unknown  ...   female                 9

                              name          isrc registration_init_time_year  \
0                                 TWD951040910                         2005
1     (You Know What| Girl?)  TWA531579702                        2011
2         (As Early As Possible)  TWB430415002                         2016
3                                 TWA459649602                         2004
4               Take Me To The Top  USWB11507713                          2015
5          (The Price of Love)  TWK951600221                        2013
6                               TWUM71300058                         2012
7              How Do I Live()       unknown                        2016
8                                 CNA231302848                         2016
9  REWIND THE MUSIC-DJ JERRY ()       unknown                         2014

   registration_init_time_month registration_init_time_day  \
0                            11                          16
1                             5                          21
2                             2                          28
3                             3                          30
4                             5                          21
5                            12                           3
6                             5                          13
7                             6                          23
8                            12                           2
9                             7                          18

  expiration_date_year  expiration_date_month  expiration_date_day
```

```
0                2017                 9                30
1                2017                10                15
2                2017                 7                17
3                2017                11                14
4                2017                 9                13
5                2017                 1                30
6                2017                10                14
7                2017                 9                25
8                2017                 6                10
9                2017                 1                17

[10 rows x 24 columns]
```

[27]:
```
categorical_feature = train.dtypes==object
categorical_cols = train.columns[categorical_feature].tolist()
categorical_cols
```

[27]:
```
['msno',
 'song_id',
 'source_system_tab',
 'source_screen_name',
 'source_type',
 'genre_ids',
 'artist_name',
 'composer',
 'lyricist',
 'gender',
 'name',
 'isrc']
```

[28]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

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

[28]:
```
    msno  song_id  source_system_tab  source_screen_name  source_type  \
0   3102    33345                  3                   7            4
1   4635    79270                  3                   7            4
2   1252    11405                  3                   7            3
3  17799    64609                  3                  10            5
4   6453    12801                  3                   7            4
5  14058    25042                  0                   3            5
6  13765    57720                  2                  11            2
7  12325    30501                  3                   7            4
8  13500    95036                  0                  10            5
9    819    82971                  3                   7            3
```

	genre_ids	artist_name	composer	lyricist	gender	name	isrc
0	206	13394	22452	8736	0	67701	49809
1	206	13931	16882	6537	0	52370	43624
2	206	12390	22167	7211	1	69074	45978
3	206	13728	22452	8736	0	49079	40371
4	106	7405	22452	8736	2	35474	75466
5	203	12771	24492	10937	0	56903	53709
6	203	12813	564	8736	1	66973	56550
7	206	12312	22452	8736	0	15829	76510
8	206	12796	24529	11007	1	61158	1508
9	71	6636	22452	8736	0	29341	76510

[29]:
```
#train.to_csv('train_data.csv')
train.head(10)
```

[29]:

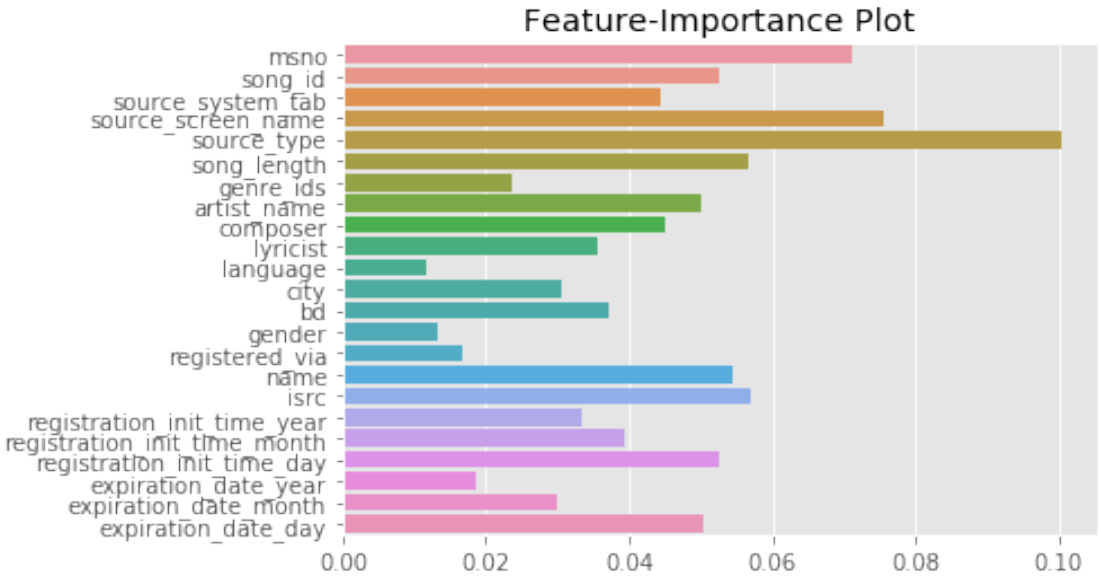
	msno	song_id	source_system_tab	source_screen_name	source_type	target	\
0	3102	33345	3	7	4	1	
1	4635	79270	3	7	4	1	
2	1252	11405	3	7	3	1	
3	17799	64609	3	10	5	0	
4	6453	12801	3	7	4	1	
5	14058	25042	0	3	5	1	
6	13765	57720	2	11	2	1	
7	12325	30501	3	7	4	1	
8	13500	95036	0	10	5	1	
9	819	82971	3	7	3	0	

	song_length	genre_ids	artist_name	composer	...	gender	registered_via	\
0	282958.0	206	13394	22452	...	0	9	
1	194168.0	206	13931	16882	...	0	7	
2	301662.0	206	12390	22167	...	1	9	
3	225593.0	206	13728	22452	...	0	9	
4	195709.0	106	7405	22452	...	2	9	
5	303438.0	203	12771	24492	...	0	9	
6	241325.0	203	12813	564	...	1	3	
7	276363.0	206	12312	22452	...	0	4	
8	273345.0	206	12796	24529	...	1	4	
9	249614.0	71	6636	22452	...	0	9	

	name	isrc	registration_init_time_year	registration_init_time_month	\
0	67701	49809	2005	11	
1	52370	43624	2011	5	
2	69074	45978	2016	2	
3	49079	40371	2004	3	
4	35474	75466	2015	5	
5	56903	53709	2013	12	
6	66973	56550	2012	5	
7	15829	76510	2016	6	

```
8  61158   1508                          2016                            12
9  29341  76510                          2014                             7

   registration_init_time_day  expiration_date_year  expiration_date_month  \
0                          16                  2017                      9
1                          21                  2017                     10
2                          28                  2017                      7
3                          30                  2017                     11
4                          21                  2017                      9
5                           3                  2017                      1
6                          13                  2017                     10
7                          23                  2017                      9
8                           2                  2017                      6
9                          18                  2017                      1

   expiration_date_day
0                   30
1                   15
2                   17
3                   14
4                   13
5                   30
6                   14
7                   25
8                   10
9                   17

[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]
```

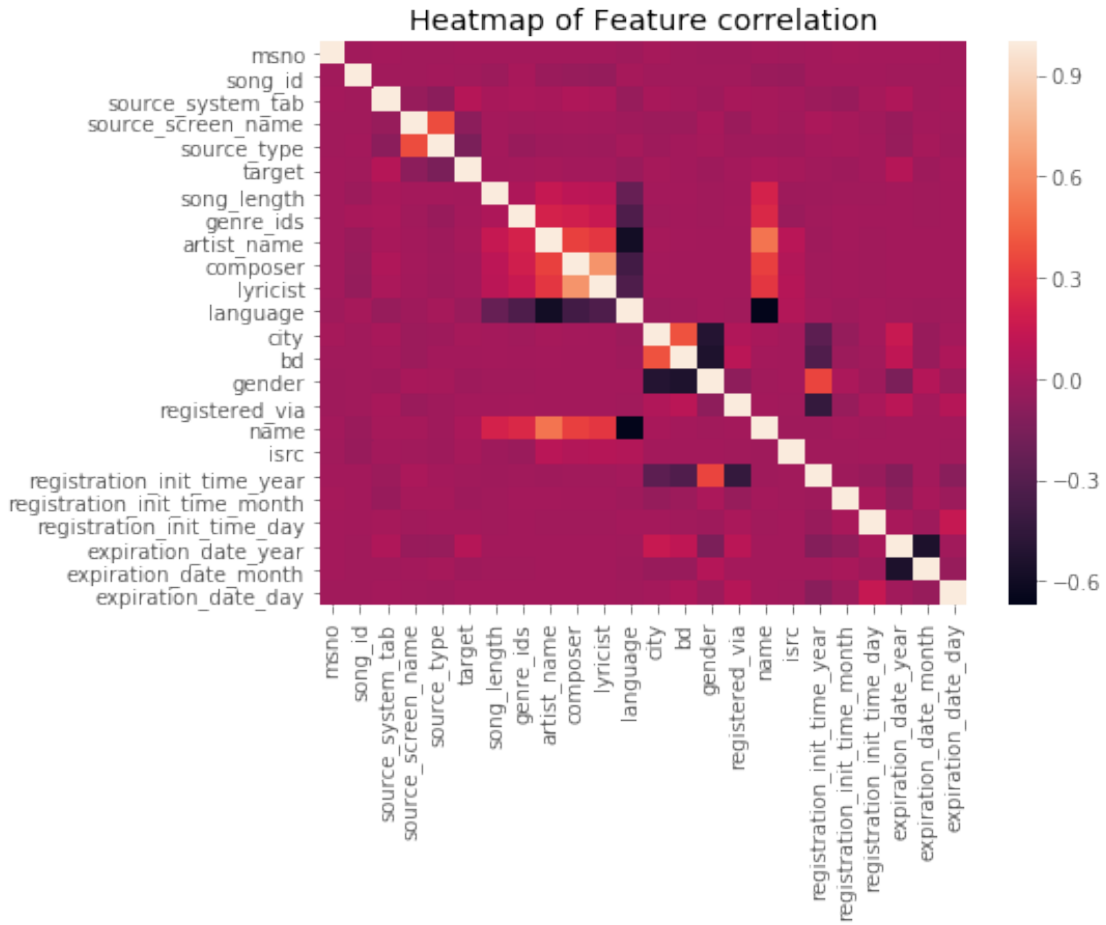
[14]:

	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()
```



[30]:
```
train.columns
#train.count(axis='columns')
```

[30]:
```
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')
```

[31]:
```
target = train.pop('target')
```

[32]:
```
train_data, test_data, train_labels, test_labels = train_test_split(train,␣
↪target, test_size = 0.3)
```

[33]:
```
# To be used as BaseLine
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

```
lrmodel = LinearRegression()
lrmodel.fit(train_data, train_labels)
test_pred = lrmodel.predict(test_data)
test_pred = np.where(test_pred > 0.49, 1, 0)

accuracy_score(test_labels, test_pred)
```

[33]: 0.6695111111111111

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

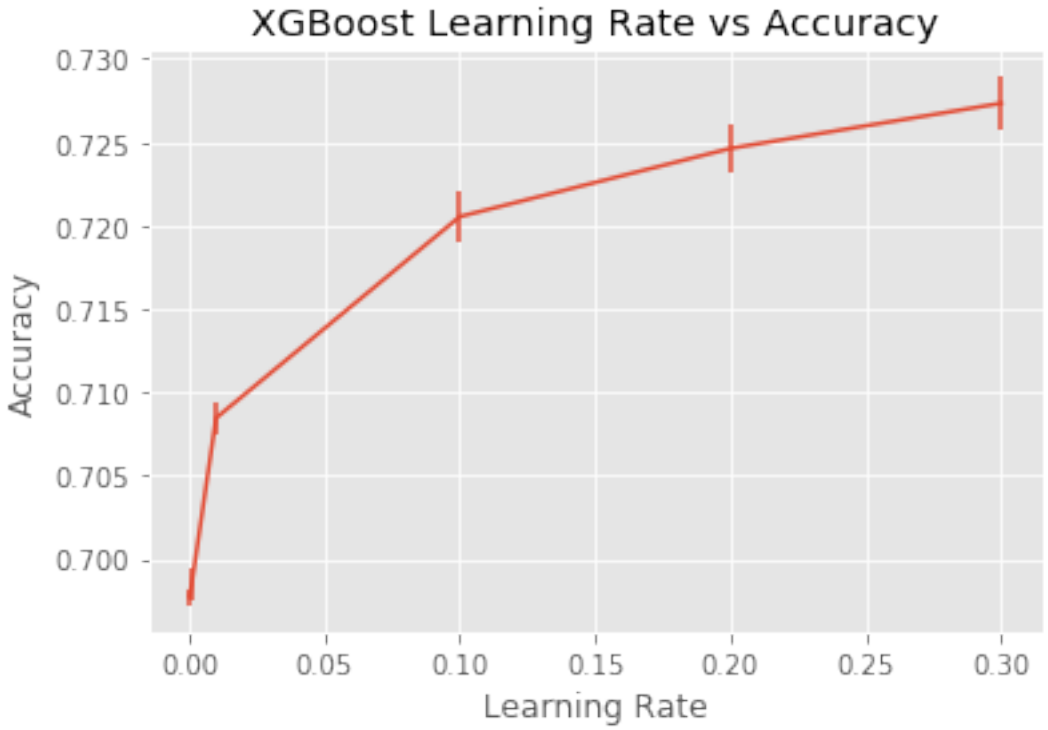
[23]:
```
model = xgb.XGBClassifier(learning_rate=0.1, max_depth=10, n_estimators=100)
model.fit(train_data, train_labels)
predict_labels = model.predict(test_data)
print(metrics.accuracy_score(test_labels, predict_labels))
```

```
0.7647022222222222
```

[24]:
```
# 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()
```

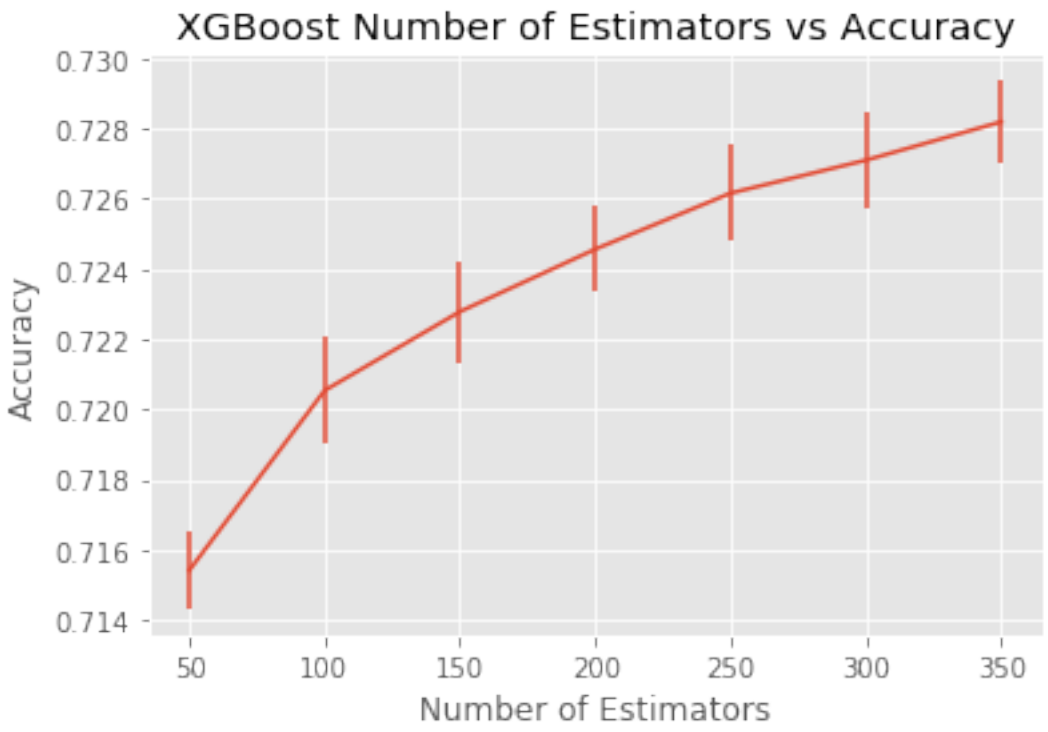
```
Best: 0.727335 accuracy {'learning_rate': 0.3}
0.697745 (0.000511) with: {'learning_rate': 0.0001}
0.698490 (0.000935) with: {'learning_rate': 0.001}
0.708467 (0.000938) with: {'learning_rate': 0.01}
0.720539 (0.001524) with: {'learning_rate': 0.1}
0.724629 (0.001386) with: {'learning_rate': 0.2}
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)
```

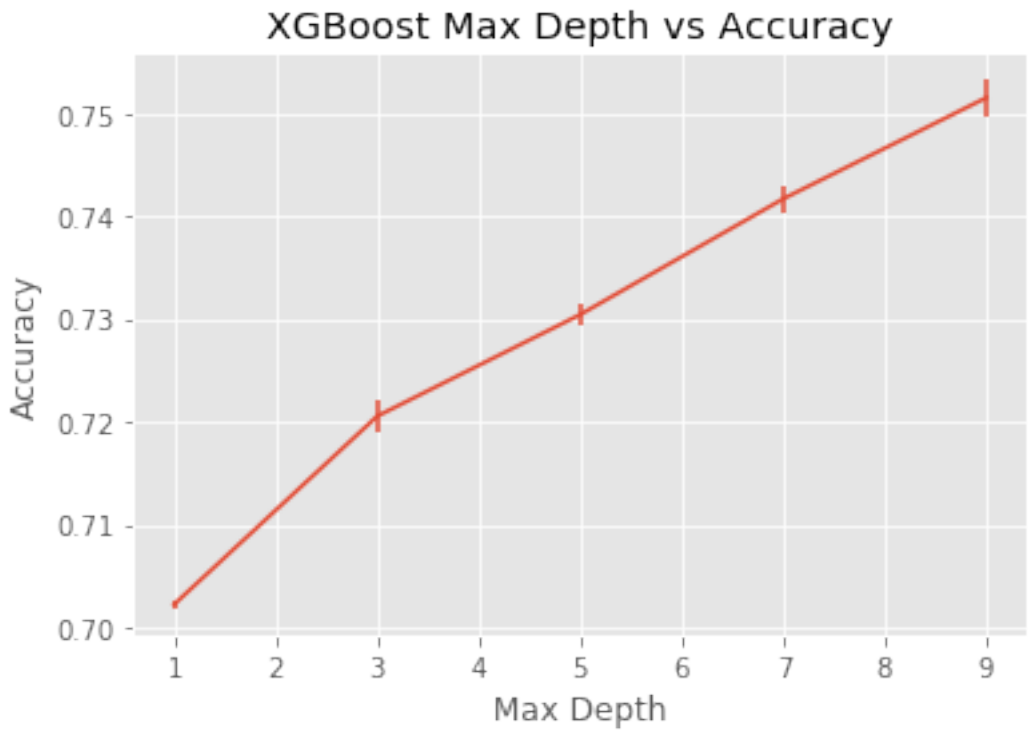
```
grid_search = GridSearchCV(model, param_grid, scoring="accuracy", n_jobs=-1)
grid_result = grid_search.fit(train_data, train_labels)

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}
```



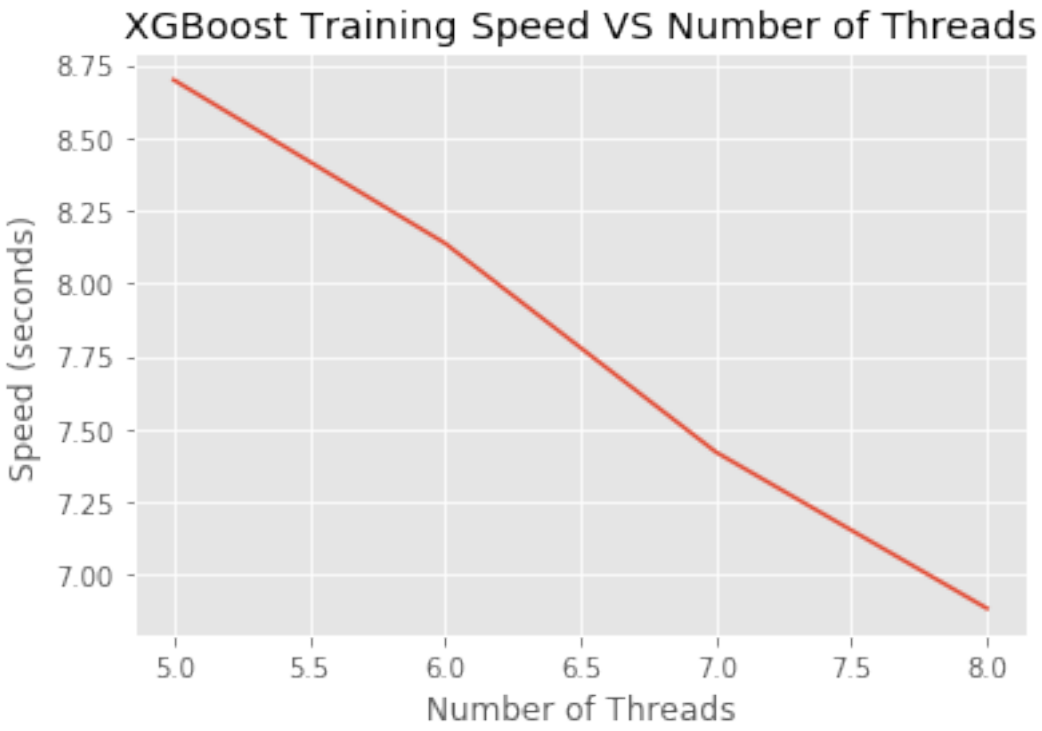
[26]:
```
# Tuning the Size of Decision Trees for Accuracy
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

model = xgb.XGBClassifier()
max_depth = range(12, 22, 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]:
```
# 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
```

0.1 Ignore the below implementation

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

```
12075000
```

[35]:
```
(525000, 23)
```

[36]:
```
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]:
```
accuracy = model.evaluate(test_data, test_labels)
```

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

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

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

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