Model_Submit

March 24, 2019

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.ensemble as sk
from sklearn.model_selection import train_test_split
from sklearn_pandas import DataFrameMapper, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
import string
import nltk
from nltk.corpus import stopwords
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn import svm
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

In [2]:
```
df = pd.read_csv('train_file.csv')
df = df.fillna('')
```

In [3]:
```
df.head()
```

Out[3]:

	ID	UsageClass	CheckoutType	CheckoutYear	CheckoutMonth	Checkouts	Title	Creator
0	1	Physical	Horizon	2005	4	1	Tidal wave	
1	2	Physical	Horizon	2005	4	1	London holiday / Richard Peck.	Peck, Richard, 1934-
2	3	Physical	Horizon	2005	4	3	Cinco de Mayo : celebrating Hispanic pride / C...	Gnojewski, Carol
3	4	Physical	Horizon	2005	4	1		
4	5	Physical	Horizon	2005	4	1		

```
3                                        Annapolis
4                                  As a man thinketh

                                                Subjects           Publisher  \
0                Tsunamis, Tsunamis Juvenile literature
1                                                                      Viking,
2  Cinco de Mayo Mexican holiday History Juvenile...  Enslow Publishers,
3  War stories, Historical fiction, Domestic fict...
4                               Thought and thinking

  PublicationYear MaterialType
0                         BOOK
1           1998.         BOOK
2          c2002.         BOOK
3                         BOOK
4                         BOOK
```

In [4]:
```
def text_process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.words('eng
```

In [5]:
```
x = df['Title']
```

In [6]:
```
bow_transformer = TfidfVectorizer(analyzer=text_process).fit(x)
X = bow_transformer.transform(x)
```

In [7]:
```
y = df['MaterialType']
le = LabelEncoder()
y=le.fit_transform(y)
```

In [8]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=
```

In [9]:
```
def fitpred(nb):
    print(nb)
    nb.fit(X_train, y_train)
    preds = nb.predict(X_test)
    mat = confusion_matrix(y_test, preds)
    print(mat)
    acc=(mat[0][0]+mat[1][1])/(mat[0][0]+mat[1][1]+mat[0][1]+mat[1][0])
    print(acc)
    print('\n')
    print(classification_report(y_test, preds))
```

In [11]:
```
nb = MultinomialNB()
fitpred(nb)
nb2 = BernoulliNB()
fitpred(nb2)
```

```
clf = svm.SVC(gamma='scale')
fitpred(clf)
clf2 = xgb.XGBClassifier()
fitpred(clf2)
clf3 = DecisionTreeClassifier()
fitpred(clf3)
clf4 = RandomForestClassifier()
fitpred(clf4)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
[[6576    0    0    0    0    3    0    0]
 [  25    0    0    0    0    0    0    0]
 [ 104    0    0    0    0    2    0    0]
 [  42    0    0    0    0   11    0    0]
 [ 323    0    0    0    0    2    0    0]
 [ 889    0    0    0    0  299    0    0]
 [ 800    0    0    0    0    3    0    0]
 [ 375    0    0    0    0    1    0   41]]
0.9962126950462051
```

	precision	recall	f1-score	support
0	0.72	1.00	0.84	6579
1	0.00	0.00	0.00	25
2	0.00	0.00	0.00	106
3	0.00	0.00	0.00	53
4	0.00	0.00	0.00	325
5	0.93	0.25	0.40	1188
6	0.00	0.00	0.00	803
7	1.00	0.10	0.18	417
micro avg	0.73	0.73	0.73	9496
macro avg	0.33	0.17	0.18	9496
weighted avg	0.66	0.73	0.64	9496

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
[[6532    0    0    0    0   31   13    3]
 [  25    0    0    0    0    0    0    0]
 [  95    0    0    0    0   11    0    0]
 [  24    0    0    0    0   29    0    0]
 [ 322    0    0    0    0    3    0    0]
 [ 838    0    0    0    0  350    0    0]
 [ 800    0    0    0    0    1    2    0]
 [ 318    0    0    0    0    5    4   90]]
0.9961872807686442
```

```
              precision    recall  f1-score   support

           0       0.73      0.99      0.84      6579
           1       0.00      0.00      0.00        25
           2       0.00      0.00      0.00       106
           3       0.00      0.00      0.00        53
           4       0.00      0.00      0.00       325
           5       0.81      0.29      0.43      1188
           6       0.11      0.00      0.00       803
           7       0.97      0.22      0.35       417

   micro avg       0.73      0.73      0.73      9496
   macro avg       0.33      0.19      0.20      9496
weighted avg       0.66      0.73      0.65      9496

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)


/home/khandalaryan/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143
  'precision', 'predicted', average, warn_for)
/home/khandalaryan/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143
  'precision', 'predicted', average, warn_for)


[[6579    0    0    0    0    0    0    0]
 [  25    0    0    0    0    0    0    0]
 [ 106    0    0    0    0    0    0    0]
 [  53    0    0    0    0    0    0    0]
 [ 325    0    0    0    0    0    0    0]
 [1151    0    0    0    0   37    0    0]
 [ 803    0    0    0    0    0    0    0]
 [ 417    0    0    0    0    0    0    0]]
0.9962144155057541


              precision    recall  f1-score   support

           0       0.70      1.00      0.82      6579
           1       0.00      0.00      0.00        25
           2       0.00      0.00      0.00       106
           3       0.00      0.00      0.00        53
           4       0.00      0.00      0.00       325
           5       1.00      0.03      0.06      1188
           6       0.00      0.00      0.00       803
           7       0.00      0.00      0.00       417
```

```
   micro avg       0.70      0.70      0.70      9496
   macro avg       0.21      0.13      0.11      9496
weighted avg       0.61      0.70      0.58      9496

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
[[6562    0    0    5    1    8    2    1]
 [  24    1    0    0    0    0    0    0]
 [  19    0   87    0    0    0    0    0]
 [  11    0    0   37    0    5    0    0]
 [ 314    0    0    0    5    6    0    0]
 [ 797    0    0    3    0  387    0    1]
 [ 788    0    0    0    1    2    3    9]
 [ 299    0    0    0    0    1    0  117]]
0.9963564596933353


              precision    recall  f1-score   support

           0       0.74      1.00      0.85      6579
           1       1.00      0.04      0.08        25
           2       1.00      0.82      0.90       106
           3       0.82      0.70      0.76        53
           4       0.71      0.02      0.03       325
           5       0.95      0.33      0.48      1188
           6       0.60      0.00      0.01       803
           7       0.91      0.28      0.43       417

   micro avg       0.76      0.76      0.76      9496
   macro avg       0.84      0.40      0.44      9496
weighted avg       0.77      0.76      0.69      9496

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
[[4936   12    5   11  100 1128  309   78]
 [  12    3    0    0    0    8    2    0]
 [  13    0   88    0    1    1    3    0]
 [   3    0    1   37    0    9    3    0]
 [ 172    1    0    1   24   89   32    6]
```

```
 [ 283    1    0   11   18  803   50   22]
 [ 340    1    1    1   26  254  142   38]
 [ 147    0    0    0    3   85   49  133]]
0.9951642151924239
```

	precision	recall	f1-score	support
0	0.84	0.75	0.79	6579
1	0.17	0.12	0.14	25
2	0.93	0.83	0.88	106
3	0.61	0.70	0.65	53
4	0.14	0.07	0.10	325
5	0.34	0.68	0.45	1188
6	0.24	0.18	0.20	803
7	0.48	0.32	0.38	417
micro avg	0.65	0.65	0.65	9496
macro avg	0.47	0.46	0.45	9496
weighted avg	0.68	0.65	0.66	9496

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)


/home/khandalaryan/anaconda3/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: Future
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


[[5608    9    0    3   65  698  161   35]
 [  16    3    0    0    0    5    1    0]
 [  16    0   87    1    0    2    0    0]
 [   8    0    0   22    0   23    0    0]
 [ 218    1    0    0   19   63   17    7]
 [ 386    0    0    2   11  747   32   10]
 [ 478    4    1    0   22  191   87   20]
 [ 182    0    0    0    1   70   25  139]]
0.9955642299503193
```

	precision	recall	f1-score	support
0	0.81	0.85	0.83	6579

```
           1       0.18      0.12      0.14        25
           2       0.99      0.82      0.90       106
           3       0.79      0.42      0.54        53
           4       0.16      0.06      0.09       325
           5       0.42      0.63      0.50      1188
           6       0.27      0.11      0.15       803
           7       0.66      0.33      0.44       417

   micro avg       0.71      0.71      0.71      9496
   macro avg       0.53      0.42      0.45      9496
weighted avg       0.69      0.71      0.69      9496
```

In [12]:
```
nbf = xgb.XGBClassifier()
nbf.fit(X, y)
```

Out[12]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='multi:softprob', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [13]:
```
test = pd.read_csv('test_file.csv')
test.fillna('')
X_t = test['Title']
X_t = bow_transformer.transform(X_t)
preds = nbf.predict(X_t)
preds = le.inverse_transform(preds)
```

In [14]:
```
id_ = test['ID']
label = preds
data = { 'ID': id_, 'MaterialType': label}
submission = pd.DataFrame(data)
submission.head(10)
```

Out[14]:

	ID	MaterialType
0	31654	BOOK
1	31655	BOOK
2	31656	SOUNDDISC
3	31657	BOOK
4	31658	BOOK
5	31659	BOOK
6	31660	BOOK
7	31661	BOOK
8	31662	BOOK
9	31663	BOOK

In [15]:
```
submission.to_csv('submit.csv',index=False)
```