jardo16_semesterproj_csc2730

December 1, 2018

Jake Ardoin semester project for CSC2730.

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [19]:
```
import IPython
import matplotlib
import numpy as np
import pandas as pd
from pprint import pprint
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_files
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import classification_report
from sklearn import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)

pd.options.mode.use_inf_as_na = True
pd.options.mode.chained_assignment = None
```

In [3]:
```
#TA: CHANGE THIS FILE NAME TO THE LOCATION OF THE FILE YOU WISH TO READ FROM
df = pd.read_json('/Users/PCnew/Documents/Datasets/News_Category_Dataset.json',
                  lines=True)
df[0:5]
```

Out[3]:

	authors	category	date	\
0	Melissa Jeltsen	CRIME	2018-05-26	
1	Andy McDonald	ENTERTAINMENT	2018-05-26	
2	Ron Dicker	ENTERTAINMENT	2018-05-26	
3	Ron Dicker	ENTERTAINMENT	2018-05-26	
4	Ron Dicker	ENTERTAINMENT	2018-05-26	

```
                                                 headline  \
0  There Were 2 Mass Shootings In Texas Last Week...
1  Will Smith Joins Diplo And Nicky Jam For The 2...
2    Hugh Grant Marries For The First Time At Age 57
3  Jim Carrey Blasts 'Castrato' Adam Schiff And D...
4  Julianna Margulies Uses Donald Trump Poop Bags...

                                                   link  \
0  https://www.huffingtonpost.com/entry/texas-ama...
1  https://www.huffingtonpost.com/entry/will-smit...
2  https://www.huffingtonpost.com/entry/hugh-gran...
3  https://www.huffingtonpost.com/entry/jim-carre...
4  https://www.huffingtonpost.com/entry/julianna-...

                                    short_description
0  She left her husband. He killed their children...
1                          Of course it has a song.
2  The actor and his longtime girlfriend Anna Ebe...
3  The actor gives Dems an ass-kicking for not fi...
4  The "Dietland" actress said using the bags is ...
```

In [4]:
```
#removing unwanted attributes TA: You can skip this if you used
#the 20k.json file, thought just downloading the actual data
#would be much simpler

df1 = df.drop(["authors", "date", "link"], axis=1)
df1.replace('', np.nan, inplace=True)
df1 = df1.dropna(axis=0)
```

In [5]:
```
df1.shape
```

Out[5]:
```
(105398, 3)
```

In [6]:
```
#my computer cant do 100k of these freaking things
#TA: Chance this slice number to the number of examples
#that you want to test for doing grading
df1=df1[0:20000]
df1.shape
```

Out[6]:
```
(20000, 3)
```

In [7]:
```
#this is explained at the end, it is to get rid of a category with only 1 example
df1 = df1[df1.category != 'GOOD NEWS']
```

In [8]:
```
#now that I've removed the blank spaces I'm left with 105398 rows of data.
#first im going to define a few data groups I need for later
#then i'll make Bag of WOrds out of the headline column + description column
#this will be done over the next handful of cells so examples can be printed of each
headline = df1["headline"]
```

```
description = df1["short_description"]
category = df1["category"]


y = category
print("Sample of y")
print(y[0:5])
y.shape
```

```
Sample of y
0            CRIME
1    ENTERTAINMENT
2    ENTERTAINMENT
3    ENTERTAINMENT
4    ENTERTAINMENT
Name: category, dtype: object
```

Out[8]: (19999,)

In [9]:
```
#here I combine the description and the title together into a single unit
X = headline + " " + description
print("Sample of X")
print(X[0:5])
X.shape
```

```
Sample of X
0    There Were 2 Mass Shootings In Texas Last Week...
1    Will Smith Joins Diplo And Nicky Jam For The 2...
2    Hugh Grant Marries For The First Time At Age 5...
3    Jim Carrey Blasts 'Castrato' Adam Schiff And D...
4    Julianna Margulies Uses Donald Trump Poop Bags...
dtype: object
```

Out[9]: (19999,)

In [10]:
```
docs_train, docs_test, y_train, y_test = train_test_split(X,
                                                          y, test_size = 0.25, random_state = 123)
```

In [11]:
```
#played with the variables to figure out the effects to decide
#on which numbers to test in each variable of the grid_search
vect = TfidfVectorizer(min_df = 0, max_df = 0.6 )
docmatrix = vect.fit_transform(docs_train)
docmatrix.shape
```

Out[11]: (14999, 24423)

In [12]:
```
pipe = Pipeline([
    ('vect', TfidfVectorizer(min_df=10, max_df=0.1)),
    ('svc', LinearSVC(C=1))])
```

```
In [13]: parameters = {
             'vect__min_df': [0, 10],
             'vect__max_df': [0.4, 0.3, 0.2],
             'svc__C': [2, 1, .1]
         }
         grid_search = GridSearchCV(pipe, parameters, cv=5, verbose=0)
         grid_search.fit(docs_train, y_train)

Out[13]: GridSearchCV(cv=5, error_score='raise',
                estimator=Pipeline(memory=None,
              steps=[('vect', TfidfVectorizer(analyzer='word', binary=False, decode_error='str
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=0.1, max_features=None, min_df=10,
                 ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True,
          ...ax_iter=1000,
              multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
              verbose=0))]),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'vect__min_df': [0, 10], 'vect__max_df': [0.4, 0.3, 0.2], 'svc__C'
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring=None, verbose=0)

In [15]: #Lets see what grid search has decided our best params are:
         print(grid_search.best_params_)

{'svc__C': 1, 'vect__max_df': 0.3, 'vect__min_df': 0}


In [16]: model = grid_search.best_estimator_
         y_predicted = model.predict(docs_test)
         print(accuracy_score(y_test, y_predicted))

0.671
```

The results from every test are as follows. I used grid.scores_ because its a bit easier to read than cv_results_ I also included cv_results_ just for completeness.

```
In [29]: df_scores = pd.DataFrame(data = grid_search.cv_results_)
         file = open("df_scores_semesterproject.pdf", "w")
         file.write(str(df_scores))
         file.close()

In [21]: print(df_scores)

    mean_fit_time  std_fit_time  mean_score_time  std_score_time param_svc__C  \
0        0.859183      0.075236         0.062490        0.000019            2
1        0.744985      0.049347         0.056253        0.007645            2
2        0.796692      0.013967         0.056231        0.007638            2
```

```
3        0.712332      0.015930         0.049988        0.006249            2
4        0.802926      0.044835         0.059367        0.006252            2
5        0.706088      0.011681         0.059360        0.006259            2
6        0.727951      0.012501         0.059370        0.006241            1
7        0.627987      0.006243         0.049987        0.006236            1
8        0.712335      0.012494         0.062481        0.000013            1
9        0.612552      0.006160         0.056230        0.007649            1
10       0.696709      0.007641         0.049995        0.006246            1
11       0.615470      0.021197         0.059369        0.006240            1
12       0.621741      0.006241         0.049986        0.006236          0.1
13       0.524882      0.007656         0.062486        0.000001          0.1
14       0.596730      0.006237         0.062494        0.000012          0.1
15       0.521763      0.021183         0.053101        0.007662          0.1
16       0.593621      0.000014         0.059362        0.006237          0.1
17       0.503001      0.006265         0.056236        0.007653          0.1

   param_vect__max_df param_vect__min_df  \
0                 0.4                  0
1                 0.4                 10
2                 0.3                  0
3                 0.3                 10
4                 0.2                  0
5                 0.2                 10
6                 0.4                  0
7                 0.4                 10
8                 0.3                  0
9                 0.3                 10
10                0.2                  0
11                0.2                 10
12                0.4                  0
13                0.4                 10
14                0.3                  0
15                0.3                 10
16                0.2                  0
17                0.2                 10

                                               params  split0_test_score  \
0   {'svc__C': 2, 'vect__max_df': 0.4, 'vect__min_...           0.660359
1   {'svc__C': 2, 'vect__max_df': 0.4, 'vect__min_...           0.616866
2   {'svc__C': 2, 'vect__max_df': 0.3, 'vect__min_...           0.662019
3   {'svc__C': 2, 'vect__max_df': 0.3, 'vect__min_...           0.615206
4   {'svc__C': 2, 'vect__max_df': 0.2, 'vect__min_...           0.648074
5   {'svc__C': 2, 'vect__max_df': 0.2, 'vect__min_...           0.608234
6   {'svc__C': 1, 'vect__max_df': 0.4, 'vect__min_...           0.666335
7   {'svc__C': 1, 'vect__max_df': 0.4, 'vect__min_...           0.634462
8   {'svc__C': 1, 'vect__max_df': 0.3, 'vect__min_...           0.665671
9   {'svc__C': 1, 'vect__max_df': 0.3, 'vect__min_...           0.635126
10  {'svc__C': 1, 'vect__max_df': 0.2, 'vect__min_...           0.654714
```

```
11  {'svc__C': 1, 'vect__max_df': 0.2, 'vect__min_...          0.623838
12  {'svc__C': 0.1, 'vect__max_df': 0.4, 'vect__mi...          0.629814
13  {'svc__C': 0.1, 'vect__max_df': 0.4, 'vect__mi...          0.629150
14  {'svc__C': 0.1, 'vect__max_df': 0.3, 'vect__mi...          0.630146
15  {'svc__C': 0.1, 'vect__max_df': 0.3, 'vect__mi...          0.629814
16  {'svc__C': 0.1, 'vect__max_df': 0.2, 'vect__mi...          0.621514
17  {'svc__C': 0.1, 'vect__max_df': 0.2, 'vect__mi...          0.621182

    split1_test_score       ...         mean_test_score  std_test_score  \
0            0.645452       ...                0.650643        0.007689
1            0.608797       ...                0.609241        0.006882
2            0.641453       ...                0.649643        0.008873
3            0.606465       ...                0.607974        0.006930
4            0.636121       ...                0.640576        0.006544
5            0.599134       ...                0.598440        0.007991
6            0.651783       ...                0.655510        0.006978
7            0.623792       ...                0.626642        0.007638
8            0.650117       ...                0.655844        0.007509
9            0.621126       ...                0.626508        0.007544
10           0.646118       ...                0.650043        0.005160
11           0.619793       ...                0.616441        0.006935
12           0.613129       ...                0.617374        0.009408
13           0.616794       ...                0.619775        0.006485
14           0.613795       ...                0.617374        0.009012
15           0.618461       ...                0.620641        0.006679
16           0.611463       ...                0.611641        0.007972
17           0.610796       ...                0.613908        0.006915

    rank_test_score  split0_train_score  split1_train_score  \
0                 3            0.999917            0.999500
1                16            0.974472            0.977080
2                 5            0.999833            0.999500
3                17            0.974472            0.977746
4                 6            0.999499            0.999583
5                18            0.974139            0.975829
6                 2            0.995078            0.994166
7                 7            0.946108            0.948908
8                 1            0.995078            0.994416
9                 8            0.947526            0.948741
10                4            0.995912            0.995083
11               13            0.946359            0.947241
12               11            0.780929            0.781130
13               10            0.736715            0.741124
14               11            0.782348            0.783547
15                9            0.738634            0.742290
16               15            0.773755            0.774046
17               14            0.730375            0.732705
```

```
    split2_train_score  split3_train_score  split4_train_score  \
0             0.999250            0.999667            0.999500
1             0.974835            0.976173            0.975431
2             0.999333            0.999667            0.999500
3             0.974835            0.976256            0.975348
4             0.999500            0.999583            0.999334
5             0.972752            0.974590            0.974931
6             0.994834            0.994585            0.993920
7             0.947421            0.950679            0.949113
8             0.994667            0.994418            0.993754
9             0.946671            0.951012            0.949113
10            0.994334            0.994335            0.994836
11            0.947254            0.948263            0.949280
12            0.784601            0.782471            0.784626
13            0.739522            0.740815            0.741401
14            0.786601            0.785220            0.785792
15            0.741855            0.741648            0.741734
16            0.775685            0.774973            0.775881
17            0.731772            0.735399            0.734322

    mean_train_score  std_train_score
0           0.999567         0.000220
1           0.975598         0.000938
2           0.999567         0.000170
3           0.975732         0.001172
4           0.999500         0.000091
5           0.974448         0.001013
6           0.994516         0.000424
7           0.948446         0.001560
8           0.994467         0.000431
9           0.948613         0.001482
10          0.994900         0.000583
11          0.947679         0.001002
12          0.782751         0.001610
13          0.739915         0.001725
14          0.784702         0.001545
15          0.741232         0.001318
16          0.774868         0.000851
17          0.732915         0.001786

[18 rows x 23 columns]
```

```
In [30]: from pprint import pprint
         pprint(grid_search.grid_scores_)
```

```
[mean: 0.65064, std: 0.00769, params: {'svc__C': 2, 'vect__max_df': 0.4, 'vect__min_df': 0},
 mean: 0.60924, std: 0.00688, params: {'svc__C': 2, 'vect__max_df': 0.4, 'vect__min_df': 10},
```

```
 mean: 0.64964, std: 0.00887, params: {'svc__C': 2, 'vect__max_df': 0.3, 'vect__min_df': 0},
 mean: 0.60797, std: 0.00693, params: {'svc__C': 2, 'vect__max_df': 0.3, 'vect__min_df': 10},
 mean: 0.64058, std: 0.00654, params: {'svc__C': 2, 'vect__max_df': 0.2, 'vect__min_df': 0},
 mean: 0.59844, std: 0.00799, params: {'svc__C': 2, 'vect__max_df': 0.2, 'vect__min_df': 10},
 mean: 0.65551, std: 0.00697, params: {'svc__C': 1, 'vect__max_df': 0.4, 'vect__min_df': 0},
 mean: 0.62664, std: 0.00764, params: {'svc__C': 1, 'vect__max_df': 0.4, 'vect__min_df': 10},
 mean: 0.65584, std: 0.00751, params: {'svc__C': 1, 'vect__max_df': 0.3, 'vect__min_df': 0},
 mean: 0.62651, std: 0.00754, params: {'svc__C': 1, 'vect__max_df': 0.3, 'vect__min_df': 10},
 mean: 0.65004, std: 0.00516, params: {'svc__C': 1, 'vect__max_df': 0.2, 'vect__min_df': 0},
 mean: 0.61644, std: 0.00694, params: {'svc__C': 1, 'vect__max_df': 0.2, 'vect__min_df': 10},
 mean: 0.61737, std: 0.00941, params: {'svc__C': 0.1, 'vect__max_df': 0.4, 'vect__min_df': 0},
 mean: 0.61977, std: 0.00648, params: {'svc__C': 0.1, 'vect__max_df': 0.4, 'vect__min_df': 10}
 mean: 0.61737, std: 0.00901, params: {'svc__C': 0.1, 'vect__max_df': 0.3, 'vect__min_df': 0},
 mean: 0.62064, std: 0.00668, params: {'svc__C': 0.1, 'vect__max_df': 0.3, 'vect__min_df': 10}
 mean: 0.61164, std: 0.00797, params: {'svc__C': 0.1, 'vect__max_df': 0.2, 'vect__min_df': 0},
 mean: 0.61391, std: 0.00691, params: {'svc__C': 0.1, 'vect__max_df': 0.2, 'vect__min_df': 10}]


C:\Users\PCnew\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:762: DeprecationW
  DeprecationWarning)
```

Below is a list of all the categories. I set this up to figure out which category only had 1 member and was gviing me an error. That is why above I removed the review pertaining to the category "GOOD NEWS". Since the error is fixed the code below is no longer serves a purpose but I left it just for completeness sake. GOOD NEWS won't appear below since I reran the cells, but it was there at first and this is how I figured out which one was screwing with my model.

```
In [31]: unique = []

         for x in y:
             if x not in unique:
                 unique.append(x)

In [32]: #this is the list of all the categories
         unique

Out[32]: ['CRIME',
          'ENTERTAINMENT',
          'WORLD NEWS',
          'IMPACT',
          'POLITICS',
          'WEIRD NEWS',
          'BLACK VOICES',
          'WOMEN',
          'COMEDY',
          'QUEER VOICES',
          'SPORTS',
          'BUSINESS',
```

```
 'TRAVEL',
 'MEDIA',
 'TECH',
 'RELIGION',
 'SCIENCE',
 'LATINO VOICES',
 'EDUCATION',
 'COLLEGE',
 'PARENTS',
 'ARTS & CULTURE',
 'STYLE',
 'GREEN',
 'TASTE',
 'HEALTHY LIVING',
 'THE WORLDPOST']
```

In [33]:
```
len(unique)
```

Out[33]:
```
27
```

In [34]:
```
#creating a dict to hold counts
catd = {"test" :0}
for item in unique:
    catd.update( {str(item) : 0})
catd
```

Out[34]:
```
{'test': 0,
 'CRIME': 0,
 'ENTERTAINMENT': 0,
 'WORLD NEWS': 0,
 'IMPACT': 0,
 'POLITICS': 0,
 'WEIRD NEWS': 0,
 'BLACK VOICES': 0,
 'WOMEN': 0,
 'COMEDY': 0,
 'QUEER VOICES': 0,
 'SPORTS': 0,
 'BUSINESS': 0,
 'TRAVEL': 0,
 'MEDIA': 0,
 'TECH': 0,
 'RELIGION': 0,
 'SCIENCE': 0,
 'LATINO VOICES': 0,
 'EDUCATION': 0,
 'COLLEGE': 0,
 'PARENTS': 0,
 'ARTS & CULTURE': 0,
```

```
 'STYLE': 0,
 'GREEN': 0,
 'TASTE': 0,
 'HEALTHY LIVING': 0,
 'THE WORLDPOST': 0}
```

```
In [35]: for x in y:
             catd[x] += 1
         catd
```

```
Out[35]: {'test': 0,
 'CRIME': 339,
 'ENTERTAINMENT': 3050,
 'WORLD NEWS': 1487,
 'IMPACT': 256,
 'POLITICS': 7007,
 'WEIRD NEWS': 385,
 'BLACK VOICES': 766,
 'WOMEN': 604,
 'COMEDY': 856,
 'QUEER VOICES': 1023,
 'SPORTS': 517,
 'BUSINESS': 240,
 'TRAVEL': 163,
 'MEDIA': 575,
 'TECH': 104,
 'RELIGION': 200,
 'SCIENCE': 74,
 'LATINO VOICES': 223,
 'EDUCATION': 133,
 'COLLEGE': 18,
 'PARENTS': 501,
 'ARTS & CULTURE': 106,
 'STYLE': 292,
 'GREEN': 290,
 'TASTE': 212,
 'HEALTHY LIVING': 546,
 'THE WORLDPOST': 32}
```