Text_Classification

March 11, 2024

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score

# suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
# read the data
df = pd.read_csv('https://raw.githubusercontent.com/nikjohn7/
 ↪Disaster-Tweets-Kaggle/main/data/train.csv')
df.head()
```

[ ]:
```
   id keyword location                                              text  \
0   1     NaN      NaN  Our Deeds are the Reason of this #earthquake M…
1   4     NaN      NaN             Forest fire near La Ronge Sask. Canada
2   5     NaN      NaN  All residents asked to 'shelter in place' are …
3   6     NaN      NaN  13,000 people receive #wildfires evacuation or…
4   7     NaN      NaN  Just got sent this photo from Ruby #Alaska as …

   target
0       1
1       1
2       1
3       1
4       1
```

[ ]:
```
# how many rows and columns are in the data set?
df.shape
```

```
[ ]: (7613, 5)
```

```
[ ]: # how many speeches are there from the United States?
     #df[df['country'] == 'USA'].shape

     #how many targets equal to one?
     df[df['target']==1].shape
```

```
[ ]: (3271, 5)
```

```
[ ]: # how many speeches are there from CANADA?
     #df[df['country'] == 'CAN'].shape

     #how many targets equal to 10?
     df[df['target'] == 0].shape
```

```
[ ]: (4342, 5)
```

```
[ ]: # show rows for the United States
     #df[df['country'] == 'USA']

     #show rows for target 1
     df[df['target'] == 1]
```

```
[ ]:         id keyword location  \
     0        1     NaN      NaN
     1        4     NaN      NaN
     2        5     NaN      NaN
     3        6     NaN      NaN
     4        7     NaN      NaN
     …      …     …       …
     7608  10869     NaN      NaN
     7609  10870     NaN      NaN
     7610  10871     NaN      NaN
     7611  10872     NaN      NaN
     7612  10873     NaN      NaN

                                                      text  target
     0     Our Deeds are the Reason of this #earthquake M…       1
     1                Forest fire near La Ronge Sask. Canada       1
     2     All residents asked to 'shelter in place' are …       1
     3     13,000 people receive #wildfires evacuation or…       1
     4     Just got sent this photo from Ruby #Alaska as …       1
     …                                                   …     …
     7608  Two giant cranes holding a bridge collapse int…       1
     7609  @aria_ahrary @TheTawniest The out of control w…       1
     7610  M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt…       1
     7611  Police investigating after an e-bike collided …       1
```

```
7612  The Latest: More Homes Razed by Northern Calif…        1

[3271 rows x 5 columns]
```

```
[ ]: # show rows for target 0
     df[df["target"] == 0 ]
```

```
[ ]:          id  keyword location  \
     15       23      NaN      NaN
     16       24      NaN      NaN
     17       25      NaN      NaN
     18       26      NaN      NaN
     19       28      NaN      NaN
     …      …      …      …
     7581  10833  wrecked  Lincoln
     7582  10834  wrecked      NaN
     7584  10837      NaN      NaN
     7587  10841      NaN      NaN
     7593  10848      NaN      NaN

                                                        text  target
     15                                        What's up man?       0
     16                                         I love fruits       0
     17                                      Summer is lovely       0
     18                                     My car is so fast       0
     19                          What a goooooooaaaaaal!!!!!!       0
     …                                                    …     …
     7581  @engineshed Great atmosphere at the British Li…       0
     7582  Cramer: Iger's 3 words that wrecked Disney's s…       0
     7584  These boxes are ready to explode! Exploding Ki…       0
     7587                                   Sirens everywhere!       0
     7593  I just heard a really loud bang and everyone i…       0

     [4342 rows x 5 columns]
```

```
[ ]: import nltk        #including and excluding stopwords nltk library   #we will␣
      ↪probably use this later
     nltk.download('stopwords')

     stopwords = set(nltk.corpus.stopwords.words('english'))

     include_stopwords =␣
      ↪{'normal','routine','everyday','regular','common','typical','usual','standard','average','g
      ↪'often'}
     #exclude_stopwords = {'against'}

     stopwords |= include_stopwords
```

```
#stopwords -= exclude_stopwords
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\naumh\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
[ ]: # build a text processing and classifier pipeline
     # to predict the country (USA or Canada) of a speech

     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn import svm
     from sklearn.model_selection import train_test_split
     from sklearn.pipeline import Pipeline
     from sklearn.metrics import classification_report

     #df2 = df[df['country'].isin(['USA', 'CAN'])]
     df2 = df[df['target'].isin([0,1])]



     # Split the dataset into training and test sets
     X_train, X_test, y_train, y_test = train_test_split(df2['text'], df2['target'],␣
      ↪test_size=0.4)

     # Create a pipeline that first transforms the text data into TF-IDF vectors,␣
      ↪then applies SVM
     text_clf = Pipeline([
         ('tfidf', TfidfVectorizer(stop_words=list(stopwords))), #sparce vectors␣
      ↪what if words not in classical vocab- words that show up in classical␣
      ↪collection
         ('clf', svm.SVC()),
     ]) #two weights for each word in the document. TF means term frequency, idf␣
      ↪means inverse document frequency, rarer word, higher it is in the document

     #"terms" are just words. Weighting scemet to capture how important word is in␣
      ↪respect to document. Used in search engines, text classification, other␣
      ↪natural language processing tasks

     #weight words by how often appear in document
     #how rare they are across all documents in a collection IDF

     #TF(t,d) = Number of times term t occurs in document d / total number of terms␣
      ↪in document d IDF is inverse fraction of documents that contain word, whic␣
      ↪his then scaled logarithmically


     #IDF(t) = log((1+n)/1+df(T)) + 1 where n is #doc. df is #occur of t in␣
      ↪collection
```

```
#df(t) is the number of documents in the collection that conatin term t

#TFIDF(t,d) = TF(t,d) * IDF(t)

#check 'toy' example in PDF

#2.1 COSINE SIMULARITY (scale invariant) based on size of vocabularity. IT'S␣
↪VERY IMPORTANT, differnt than eucledean.

#we use tf idf vectorizer in notebook



# Train the classifier
text_clf.fit(X_train, y_train)

# Predict the test set results
y_pred = text_clf.predict(X_test)

# Print the classification report
print(classification_report(y_test, y_pred, target_names=['0', '1']))
```

```
              precision    recall  f1-score   support

           0       0.78      0.87      0.82      1749
           1       0.79      0.67      0.72      1297

    accuracy                           0.78      3046
   macro avg       0.78      0.77      0.77      3046
weighted avg       0.78      0.78      0.78      3046
```

[ ]:
```
# This script creates a new column 'sentiment' in the dataframe,
# which contains the sentiment score of the text.
# The sentiment score is a float within the range [-1.0, 1.0],
# where -1.0 denotes a very negative sentiment,
# 1.0 denotes a very positive sentiment,
# and values around 0 denote a neutral sentiment.

from textblob import TextBlob

# Define a function to apply sentiment analysis to a text
def get_sentiment(text):
    blob = TextBlob(text)
    return blob.sentiment.polarity  # returns a value between -1 and 1
```

```
# Create a new column 'sentiment' in the DataFrame
#df2['sentiment'] = df2['text'].apply(get_sentiment)
df2['sentiment'] = df2['text'].apply(get_sentiment)

# Display the DataFrame
df2
```

[ ]:
```
          id keyword location  \
0          1     NaN      NaN
1          4     NaN      NaN
2          5     NaN      NaN
3          6     NaN      NaN
4          7     NaN      NaN
…        …       …        …
7608   10869     NaN      NaN
7609   10870     NaN      NaN
7610   10871     NaN      NaN
7611   10872     NaN      NaN
7612   10873     NaN      NaN

                                                   text  target  sentiment
0     Our Deeds are the Reason of this #earthquake M…       1   0.000000
1                Forest fire near La Ronge Sask. Canada       1   0.100000
2     All residents asked to 'shelter in place' are …       1  -0.018750
3     13,000 people receive #wildfires evacuation or…       1   0.000000
4     Just got sent this photo from Ruby #Alaska as …       1   0.000000
…                                                   …     …         …
7608  Two giant cranes holding a bridge collapse int…       1   0.000000
7609  @aria_ahrary @TheTawniest The out of control w…       1   0.150000
7610  M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt…       1   0.000000
7611  Police investigating after an e-bike collided …       1  -0.260417
7612  The Latest: More Homes Razed by Northern Calif…       1   0.500000

[7613 rows x 6 columns]
```

[ ]:
```
# find average sentiment for each country in df2
#df2.groupby('country')['sentiment'].mean()
df2.groupby('target')['sentiment'].mean()
```

[ ]:
```
target
0    0.070622
1    0.018631
Name: sentiment, dtype: float64
```

[ ]:
```
# find average sentiment for each speaker in df2
# order the results from most positive to most negative
```

```
#df2.groupby('speaker')['sentiment'].mean().sort_values(ascending=False).head(5)
df2.groupby('keyword')['sentiment'].mean().sort_values(ascending=False).head(5)
```

```
[ ]: keyword
     hazardous    0.457891
     razed        0.418946
     outbreak     0.312661
     mayhem       0.277262
     wreckage     0.273440
     Name: sentiment, dtype: float64
```

```
[ ]: #df2.groupby('year')['sentiment'].mean().sort_values(ascending=False)
     df2.groupby('location')['sentiment'].mean().sort_values(ascending=False)
```

```
[ ]: location
     The Waystone Inn            1.0
     Florida USA                 1.0
     Paranaque City              1.0
     West Palm Beach, Florida    1.0
     Washington, DC 20009        1.0
                                ...
     Mumbai india               -1.0
     Thibodaux, LA              -1.0
     fujo garbage heaven        -1.0
     sri lanka                  -1.0
     Milton Keynes, England     -1.0
     Name: sentiment, Length: 3341, dtype: float64
```