NLP_Project

April 21, 2023

DISASTER TWEETS PREDICTION USING NATURAL LANGUAGE PROCESSING

TEAM MEMBERS:

T.S.S. ABINANDHAN KUMAR | 19MIA1062

NIRANJAN J | 19MIA1003

ALAGARSAMY N | 19MIA1082

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

1 1. Importing the necessary libraries

[4]:
```
import numpy as np
import pandas as pd

# text processing libraries
import re
import string
import nltk
from nltk.corpus import stopwords

# XGBoost
import xgboost as xgb
from xgboost import XGBClassifier

# sklearn
from sklearn import model_selection
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import f1_score
```

```
from sklearn import preprocessing, decomposition, model_selection, metrics,␣
 ↪pipeline
from sklearn.model_selection import␣
 ↪GridSearchCV,StratifiedKFold,RandomizedSearchCV

# matplotlib and seaborn for plotting
import matplotlib.pyplot as plt
import seaborn as sns

# File system manangement
import os

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

2 2. Reading the datasets

[5]:
```
#Training data
train = pd.read_csv('/content/drive/MyDrive/NLP/train.csv')
train.head()
```

[5]:
```
   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
```

[6]:
```
test = pd.read_csv('/content/drive/MyDrive/NLP/test.csv')
test.head()
```

[6]:
```
   id keyword location                                              text
0   0     NaN      NaN                 Just happened a terrible car crash
1   2     NaN      NaN  Heard about #earthquake is different cities, s…
2   3     NaN      NaN  there is a forest fire at spot pond, geese are…
3   9     NaN      NaN           Apocalypse lighting. #Spokane #wildfires
4  11     NaN      NaN      Typhoon Soudelor kills 28 in China and Taiwan
```

3 3. Basic EDA

3.1 Missing values

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

[7]:
```
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
```

[8]:
```
test.isnull().sum()
```

[8]:
```
id             0
keyword       26
location    1105
text           0
dtype: int64
```

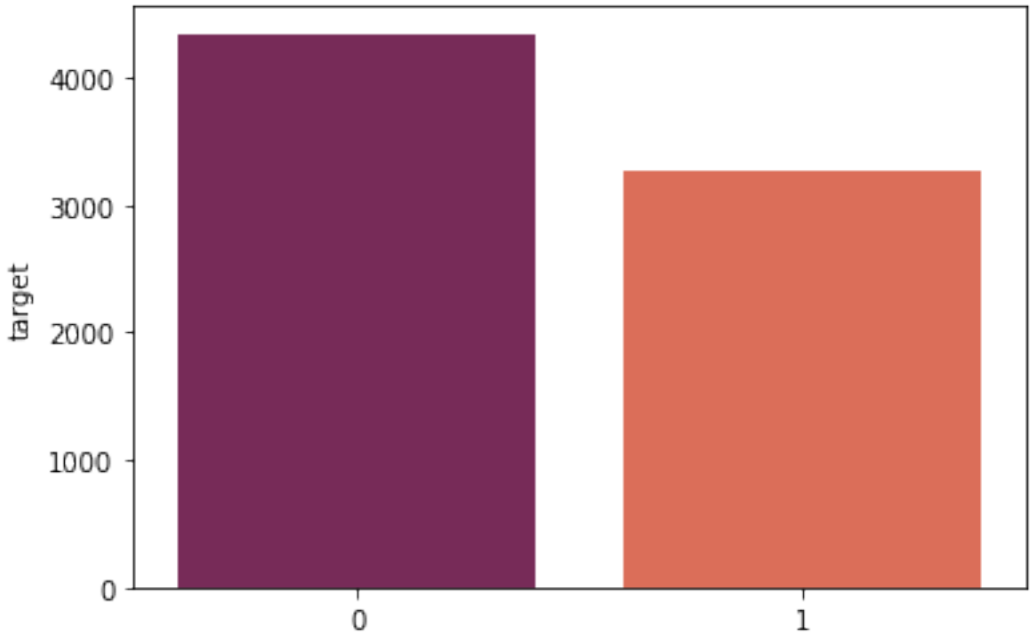
3.2 Exploring the Target Column

[9]:
```
train['target'].value_counts()
```

[9]:
```
0    4342
1    3271
Name: target, dtype: int64
```

[10]:
```
sns.barplot(train['target'].value_counts().index,train['target'].
↪value_counts(),palette='rocket')
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e8c856090>
```



[11]:
```
disaster_tweets = train[train['target']==1]['text']
disaster_tweets.values[1]
```

[11]: 'Forest fire near La Ronge Sask. Canada'

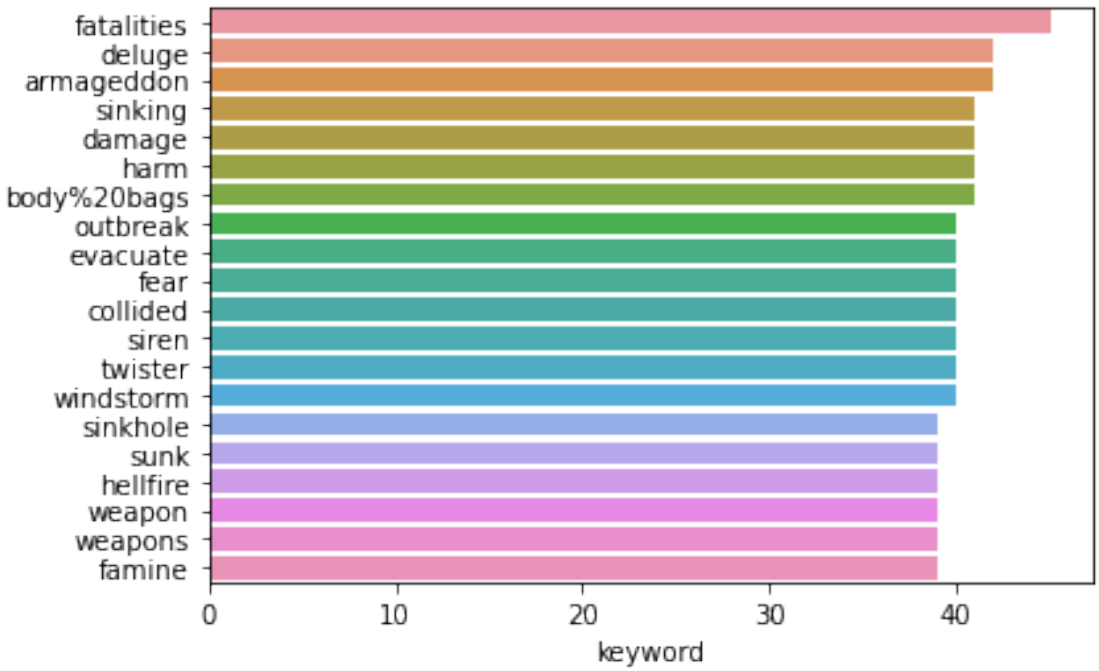
[12]:
```
non_disaster_tweets = train[train['target']==0]['text']
non_disaster_tweets.values[1]
```

[12]: 'I love fruits'

3.3 Exploring the 'keyword' column

[13]:
```
sns.barplot(y=train['keyword'].value_counts()[:20].index,x=train['keyword'].
↪value_counts()[:20],
            orient='h')
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5e8c73c110>



[14]:
```
train.loc[train['text'].str.contains('disaster', na=False, case=False)].target.
↪value_counts()
```

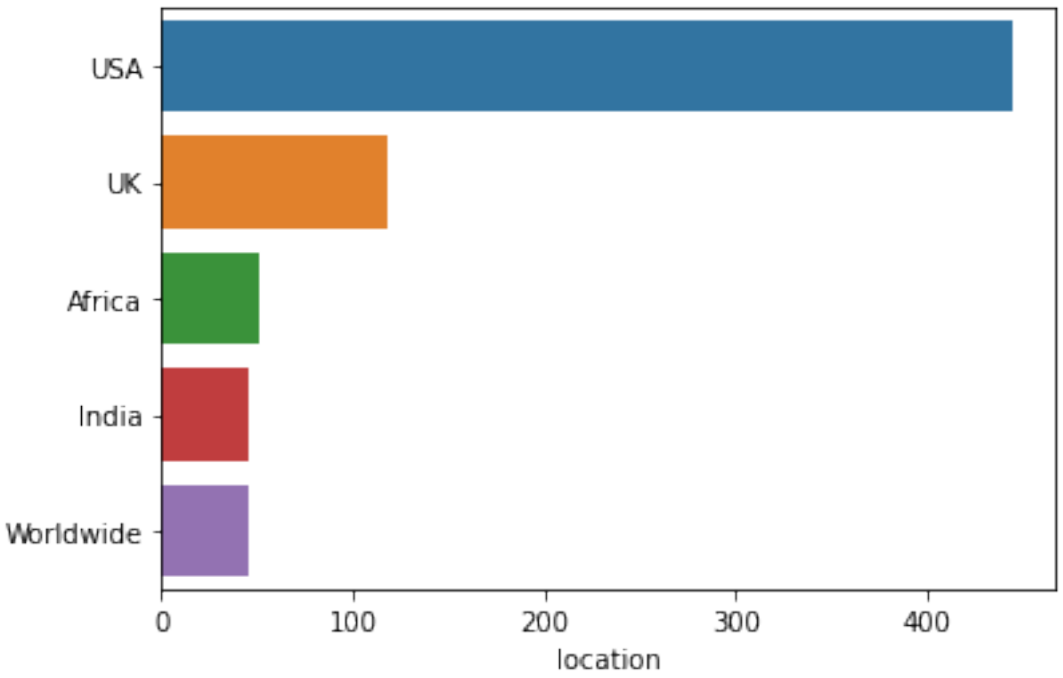
[14]:
```
1    102
0     40
Name: target, dtype: int64
```

3.4 Exploring the 'location' column

[15]:
```
train['location'].replace({'United States':'USA',
                           'New York':'USA',
                            "London":'UK',
                            "Los Angeles, CA":'USA',
                            "Washington, D.C.":'USA',
                            "California":'USA',
                             "Chicago, IL":'USA',
                             "Chicago":'USA',
                            "New York, NY":'USA',
                            "California, USA":'USA',
                            "FLorida":'USA',
                            "Nigeria":'Africa',
                            "Kenya":'Africa',
                            "Everywhere":'Worldwide',
                            "San Francisco":'USA',
                            "Florida":'USA',
```

```
                            "United Kingdom":'UK',
                            "Los Angeles":'USA',
                            "Toronto":'Canada',
                            "San Francisco, CA":'USA',
                            "NYC":'USA',
                            "Seattle":'USA',
                            "Earth":'Worldwide',
                            "Ireland":'UK',
                            "London, England":'UK',
                            "New York City":'USA',
                            "Texas":'USA',
                            "London, UK":'UK',
                            "Atlanta, GA":'USA',
                            "Mumbai":"India"},inplace=True)

sns.barplot(y=train['location'].value_counts()[:5].index,x=train['location'].
 ↪value_counts()[:5],
            orient='h')
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5e8c749f90>



[16]:
```
train['text'][:5]
```

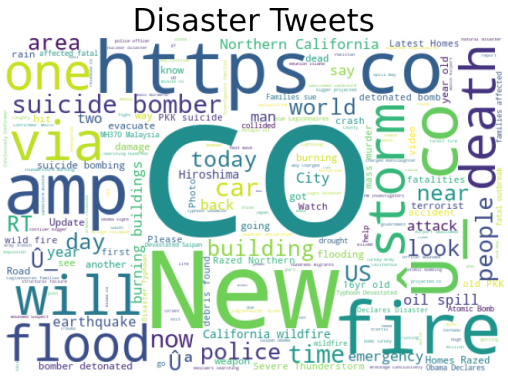
[16]:
```
0    Our Deeds are the Reason of this #earthquake M…
1               Forest fire near La Ronge Sask. Canada
2    All residents asked to 'shelter in place' are …
3    13,000 people receive #wildfires evacuation or…
4    Just got sent this photo from Ruby #Alaska as …
Name: text, dtype: object
```

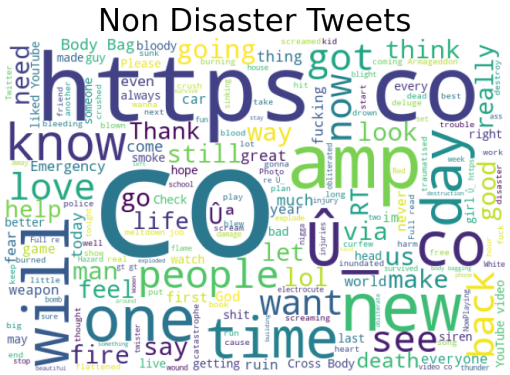
TEXT PREPROCESSING

[17]:
```
def clean_text(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

train['text'] = train['text'].apply(lambda x: clean_text(x))
test['text'] = test['text'].apply(lambda x: clean_text(x))

train['text'].head()
```

[17]:
```
0    our deeds are the reason of this earthquake ma…
1                forest fire near la ronge sask canada
2    all residents asked to shelter in place are be…
3     people receive wildfires evacuation orders in…
4    just got sent this photo from ruby alaska as s…
Name: text, dtype: object
```

[18]:
```
from wordcloud import WordCloud
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[26, 8])
wordcloud1 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(disaster_tweets))
ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Disaster Tweets',fontsize=40);

wordcloud2 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(non_disaster_tweets))
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Non Disaster Tweets',fontsize=40);
```





3.5 2. Tokenization

[19]:
```
text = "Are you coming , aren't you"
tokenizer1 = nltk.tokenize.WhitespaceTokenizer()
tokenizer2 = nltk.tokenize.TreebankWordTokenizer()
tokenizer3 = nltk.tokenize.WordPunctTokenizer()
tokenizer4 = nltk.tokenize.RegexpTokenizer(r'\w+')

print("Example Text: ",text)
print("----------------------------------------------------------------------------------------------------")
print("Tokenization by whitespace:- ",tokenizer1.tokenize(text))
print("Tokenization by words using Treebank Word Tokenizer:- ",tokenizer2.
  ↪tokenize(text))
print("Tokenization by punctuation:- ",tokenizer3.tokenize(text))
print("Tokenization by regular expression:- ",tokenizer4.tokenize(text))
```

```
Example Text:  Are you coming , aren't you
----------------------------------------------------------------------------------------------------
Tokenization by whitespace:-  ['Are', 'you', 'coming', ',', "aren't", 'you']
Tokenization by words using Treebank Word Tokenizer:-  ['Are', 'you', 'coming',
',', 'are', "n't", 'you']
Tokenization by punctuation:-  ['Are', 'you', 'coming', ',', 'aren', "'", 't',
'you']
Tokenization by regular expression:-  ['Are', 'you', 'coming', 'aren', 't',
'you']
```

[20]:
```
tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')
train['text'] = train['text'].apply(lambda x: tokenizer.tokenize(x))
test['text'] = test['text'].apply(lambda x: tokenizer.tokenize(x))
train['text'].head()
```

[20]:
```
0    [our, deeds, are, the, reason, of, this, earth…
1        [forest, fire, near, la, ronge, sask, canada]
2    [all, residents, asked, to, shelter, in, place…
3    [people, receive, wildfires, evacuation, order…
4    [just, got, sent, this, photo, from, ruby, ala…
Name: text, dtype: object
```

3. STOPWORDS REMOVAL

[21]:
```
import nltk
nltk.download('stopwords')
def remove_stopwords(text):

    words = [w for w in text if w not in stopwords.words('english')]
    return words


train['text'] = train['text'].apply(lambda x : remove_stopwords(x))
test['text'] = test['text'].apply(lambda x : remove_stopwords(x))
train.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[21]:

	id	keyword	location	text	target
0	1	NaN	NaN	[deeds, reason, earthquake, may, allah, forgiv…	1
1	4	NaN	NaN	[forest, fire, near, la, ronge, sask, canada]	1
2	5	NaN	NaN	[residents, asked, shelter, place, notified, o…	1
3	6	NaN	NaN	[people, receive, wildfires, evacuation, order…	1
4	7	NaN	NaN	[got, sent, photo, ruby, alaska, smoke, wildfi…	1

3.6 4. Token normalization

Token normalisation means converting different tokens to their base forms. This can be done either by:

- **Stemming** : removing and replacing suffixes to get to the root form of the word, which is called the **stem** for instance cats - cat, wolves - wolv
- **Lemmatization** : Returns the base or dictionary form of a word, which is known as the **lemma**

source

[22]:
```
nltk.download('wordnet')
# Stemming and Lemmatization examples
text = "feet cats wolves talked"

tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)

# Stemmer
stemmer = nltk.stem.PorterStemmer()
print("Stemming the sentence: ", " ".join(stemmer.stem(token) for token in␣
 ↪tokens))

# Lemmatizer
lemmatizer=nltk.stem.WordNetLemmatizer()
print("Lemmatizing the sentence: ", " ".join(lemmatizer.lemmatize(token) for␣
 ↪token in tokens))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Unzipping corpora/wordnet.zip.
Stemming the sentence:  feet cat wolv talk
Lemmatizing the sentence:  foot cat wolf talked
```

[23]:
```
def combine_text(list_of_text):
    '''Takes a list of text and combines them into one large chunk of text.'''
    combined_text = ' '.join(list_of_text)
    return combined_text

train['text'] = train['text'].apply(lambda x : combine_text(x))
test['text'] = test['text'].apply(lambda x : combine_text(x))
train['text']
train.head()
```

[23]:

	id	keyword	location	text	\
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	
1	4	NaN	NaN	forest fire near la ronge sask canada	
2	5	NaN	NaN	residents asked shelter place notified officer…	
3	6	NaN	NaN	people receive wildfires evacuation orders cal…	
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou…	

	target
0	1
1	1
2	1
3	1
4	1

[24]:
```
def text_preprocessing(text):
    """
    Cleaning and parsing the text.

    """
    tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')

    nopunc = clean_text(text)
    tokenized_text = tokenizer.tokenize(nopunc)
    remove_stopwords = [w for w in tokenized_text if w not in stopwords.
 ↪words('english')]
    combined_text = ' '.join(remove_stopwords)
    return combined_text
```

[25]:
```
count_vectorizer = CountVectorizer()
train_vectors = count_vectorizer.fit_transform(train['text'])
test_vectors = count_vectorizer.transform(test["text"])


print(train_vectors[0].todense())
```

```
[[0 0 0 … 0 0 0]]
```

[26]:
```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
train_tfidf = tfidf.fit_transform(train['text'])
test_tfidf = tfidf.transform(test["text"])
```

4 6. Building a Text Classification model

4.1 Logistic Regression Classifier

[27]:
```
clf = LogisticRegression(C=1.0)
scores = model_selection.cross_val_score(clf, train_vectors, train["target"],␣
 ↪cv=5, scoring="f1")
scores
```

[27]:
```
array([0.59865255, 0.49611063, 0.57166948, 0.56290774, 0.68789809])
```

[28]:
```
clf.fit(train_vectors, train["target"])
```

[28]:
```
LogisticRegression()
```

[29]:
```
clf_tfidf = LogisticRegression(C=1.0)
scores = model_selection.cross_val_score(clf_tfidf, train_tfidf,␣
 ↪train["target"], cv=5, scoring="f1")
scores
```

[29]:
```
array([0.57229525, 0.49673203, 0.54277829, 0.46618106, 0.64768683])
```

4.2 Naives Bayes Classifier

[30]:
```
clf_NB = MultinomialNB()
scores = model_selection.cross_val_score(clf_NB, train_vectors,␣
 ↪train["target"], cv=5, scoring="f1")
scores
```

[30]:
```
array([0.63149079, 0.60675773, 0.68575519, 0.64341085, 0.72505092])
```

[31]:
```
clf_NB.fit(train_vectors, train["target"])
```

[31]:
```
MultinomialNB()
```

[32]:
```
clf_NB_TFIDF = MultinomialNB()
scores = model_selection.cross_val_score(clf_NB_TFIDF, train_tfidf,␣
 ↪train["target"], cv=5, scoring="f1")
scores
```

[32]:
```
array([0.57590597, 0.57092511, 0.61135371, 0.5962963 , 0.7393745 ])
```

[33]:
```
clf_NB_TFIDF.fit(train_tfidf, train["target"])
```

[33]:
```
MultinomialNB()
```

4.3 XGBoost

[34]:
```
import xgboost as xgb
clf_xgb = xgb.XGBClassifier(max_depth=7, n_estimators=200, colsample_bytree=0.8,
                        subsample=0.8, nthread=10, learning_rate=0.1)
scores = model_selection.cross_val_score(clf_xgb, train_vectors,␣
 ↪train["target"], cv=5, scoring="f1")
scores
```

[34]:
```
array([0.47379913, 0.37379576, 0.43988816, 0.38900634, 0.53142857])
```

[35]:
```
import xgboost as xgb
clf_xgb_TFIDF = xgb.XGBClassifier(max_depth=7, n_estimators=200,␣
 ↪colsample_bytree=0.8,
                        subsample=0.8, nthread=10, learning_rate=0.1)
scores = model_selection.cross_val_score(clf_xgb_TFIDF, train_tfidf,␣
 ↪train["target"], cv=5, scoring="f1")
scores
```

[35]:
```
array([0.48947951, 0.34406439, 0.43140965, 0.40084388, 0.53014354])
```