Disaster Tweet Analyzing

S. RAHUL VENKAT

215229129

Importing

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv(r"train.csv")
```

In [3]:
```
df1=pd.read_csv(r"test.csv")
```

Preprocessing & Data Cleaning

In [4]:
```
df=df.drop(['keyword','location','id'],axis=1)
```

In [5]:
```
df1=df1.drop(['keyword','location','id'],axis=1)
```

In [6]:
```
df.head()
```

Out[6]:

	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

In [7]:
```
df1.head()
```

Out[7]:

	text
0	Just happened a terrible car crash
1	Heard about #earthquake is different cities, s...
2	there is a forest fire at spot pond, geese are...
3	Apocalypse lighting. #Spokane #wildfires
4	Typhoon Soudelor kills 28 in China and Taiwan

In [8]:
```
import re
import nltk
from nltk.corpus import stopwords
```

In [10]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Asus-2022\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[10]:
```
True
```

In [11]:
```
URL_PATTERN = '((http|https)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-_=#]+\.([a-zA-Z]){2,6}([
all_stopwords = stopwords.words('english')

def process_text(text):
    # remove stopwords
    remove_stop = ' '.join([word for word in text.split() if word not in all_stop
    #remove url
    remove_url = re.sub(URL_PATTERN, '', remove_stop)
    #remove punctuation
    remove_punc = re.sub(r'[^\w\s]', '', remove_url)

    return remove_punc.lower()
```

Tokenization

In [12]:
```
import nltk
from nltk import TweetTokenizer

tokenizer = TweetTokenizer()

df['tokens'] = [tokenizer.tokenize(item) for item in df.text]
df1['tokens'] = [tokenizer.tokenize(item) for item in df1.text]
```

Lemmatization

In [14]:

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Asus-2022\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
```

Out[14]:

```
True
```

In [16]:

```
import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Asus-2022\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\omw-1.4.zip.
```

Out[16]:

```
True
```

In [18]:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatize_item(item):
    new_item = []
    for x in item:
        x = lemmatizer.lemmatize(x)
        new_item.append(x)
    return " ".join(new_item)

df['tokens'] = [lemmatize_item(item) for item in df.tokens]
df1['tokens'] = [lemmatize_item(item) for item in df1.tokens]
```

Vectorization

In [19]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()


X = vectorizer.fit_transform(df.text).toarray()
y = df['target']

test_texts = vectorizer.transform(df1["text"])
```

In [ ]:

```

```

In [20]:
```
#Checking

df.isnull().sum()
```

Out[20]:
```
text      0
target    0
tokens    0
dtype: int64
```

Model Building

In [21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, randon
```

In [22]:
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(loss='hinge',penalty='l2')

sgd.fit(X_train,y_train)

y_pred = sgd.predict(X_test)
```

In [23]:
```
y_pred
```

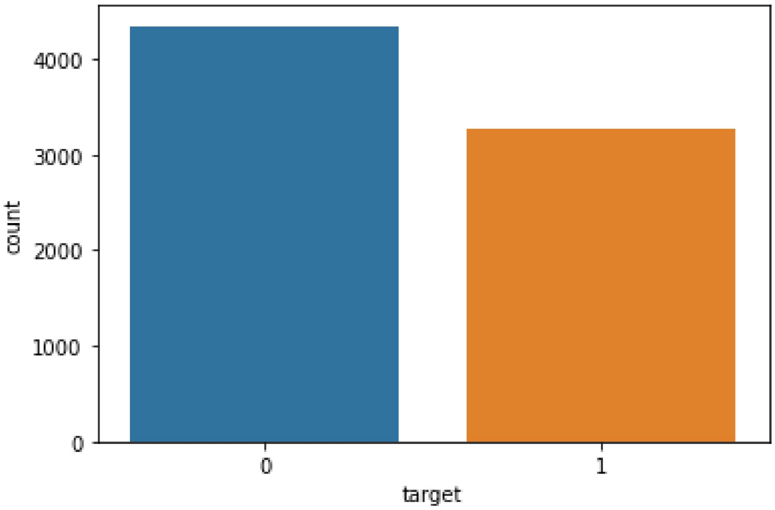
Out[23]:
```
array([0, 1, 0, ..., 0, 0, 1], dtype=int64)
```

EDA

In [24]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='target',data=df)
```

Out[24]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



Metrics

Confusion matrix

In [25]:

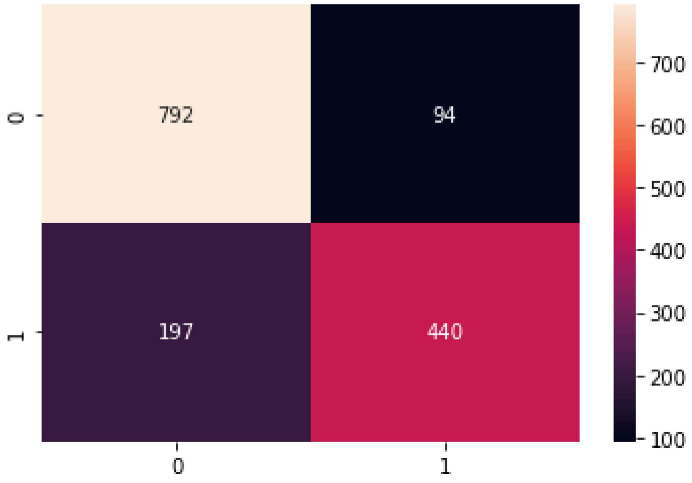
```
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(y_test, y_pred)
cf_matrix
```

Out[25]:

```
array([[792,  94],
       [197, 440]], dtype=int64)
```

In [26]:

```
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='g')
plt.show()
```



Classification report

In [27]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       886
           1       0.82      0.69      0.75       637

    accuracy                           0.81      1523
   macro avg       0.81      0.79      0.80      1523
weighted avg       0.81      0.81      0.81      1523
```

Accuracy

In [28]:
```
from sklearn.metrics import accuracy_score
print('Accuracy: ',accuracy_score(y_test,y_pred))
```

```
Accuracy:  0.8089297439264609
```

F1 Score

In [29]:
```
# finding f1_score
from sklearn.metrics import f1_score

print("F1 Score =", f1_score(y_test, y_pred, average='weighted'))
```

```
F1 Score = 0.8057746317355348
```

Submission file

In [30]:
```
submission=pd.read_csv('sample_submission.csv')
```

In [31]:
```
submission["target"]=sgd.predict(test_texts)
submission[:3]
```

Out[31]:

	id	target
0	0	1
1	2	1
2	3	1

In [32]:
```
submission.to_csv('s+submission.csv', index=False)
```

In [33]:
```
df3=pd.read_csv('s+submission.csv')
```

In [34]:
```
df3.shape
```

Out[34]:
```
(3263, 2)
```

In [35]:
```
df3.head()
```

Out[35]:

	id	target
0	0	1
1	2	1
2	3	1
3	9	1
4	11	1

In [ ]:

In [ ]: