text-classification-1

March 5, 2024

[1]:
```
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')
```

[2]:
```
# read the data
df = pd.read_csv('https://raw.githubusercontent.com/nikjohn7/
 ↪Disaster-Tweets-Kaggle/main/data/train.csv')
df.head()
```

[2]:

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

[3]:
```
# how many rows and columns are in the data set?
df.shape
```

[3]:
```
(7613, 5)
```

[11]:
```
import nltk
nltk.download('stopwords')

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

# Include: Common informal terms and fillers might be irrelevant
include_stopwords = {'lol', 'haha', 'um', 'uh', 'like', 'oh', 'hey', 'hi',␣
 ↪'please'}

# Exclude: Words that could be significant in the context of disasters
exclude_stopwords = {'urgent', 'immediate', 'help', 'need', 'missing',␣
 ↪'trapped', 'now', 'today', 'tonight', 'alert', 'emergency', 'evacuate',␣
 ↪'evacuation'}

stopwords |= include_stopwords
stopwords -= exclude_stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[12]:
```
# build a text processing and classifier pipeline
# to predict the if a tweet is a disaster or non disaster

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

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['target'],␣
 ↪test_size=0.2)

# 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))),
    ('clf', svm.SVC()),
])

# 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=['Non-Disaster',␣
 ↪'Disaster']))
```

```
              precision    recall  f1-score   support

Non-Disaster       0.81      0.90      0.85       906
    Disaster       0.83      0.69      0.75       617

    accuracy                           0.82      1523
   macro avg       0.82      0.80      0.80      1523
weighted avg       0.82      0.82      0.81      1523
```