Twitter hate

September 21, 2021

[1]:
```
import pandas as pd, numpy as np
import os, re
```

Read in the csv using pandas

[2]:
```
inp_tweets0 = pd.read_csv("TwitterHate.csv")
inp_tweets0.head()
```

[2]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s…
1	2	0	@user @user thanks for #lyft credit i can't us…
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in …
4	5	0	factsguide: society now #motivation

[3]:
```
inp_tweets0.label.value_counts(normalize=True)
```

[3]:
```
0    0.929854
1    0.070146
Name: label, dtype: float64
```

[4]:
```
inp_tweets0.tweet.sample().values[0]
```

[4]:
```
'#adobephotoshop  buffalo simulation: buffalo for you to take in the vicinity
of their homes to do. in this '
```

Get the tweets into a list, for easy text clean up and manipulation

[5]:
```
tweets0 = inp_tweets0.tweet.values
```

[6]:
```
len(tweets0)
```

[6]:
```
31962
```

[7]:
```
tweets0[:5]
```

[7]:
```
array([' @user when a father is dysfunctional and is so selfish he drags his
kids into his dysfunction.   #run',
       "@user @user thanks for #lyft credit i can't use cause they don't offer
wheelchair vans in pdx.    #disapointed #getthanked",
       '  bihday your majesty',
       '#model   i love u take with u all the time in urð\x9f\x93±!!! ð\x9f\x98\
x99ð\x9f\x98\x8eð\x9f\x91\x84ð\x9f\x91\x85ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦  ',
       ' factsguide: society now    #motivation'], dtype=object)
```

The tweets contain - 1. URLs 2. Hashtags 3. User handles 4. 'RT'

0.1 Cleanup

Normalizing case

[8]:
```
tweets_lower = [twt.lower() for twt in tweets0]
```

[9]:
```
tweets_lower[:5]
```

[9]:
```
[' @user when a father is dysfunctional and is so selfish he drags his kids into
his dysfunction.   #run',
 "@user @user thanks for #lyft credit i can't use cause they don't offer
wheelchair vans in pdx.    #disapointed #getthanked",
 '  bihday your majesty',
 '#model   i love u take with u all the time in urð\x9f\x93±!!! ð\x9f\x98\x99ð\x
9f\x98\x8eð\x9f\x91\x84ð\x9f\x91\x85ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦  ',
 ' factsguide: society now    #motivation']
```

Remove user handles, begin with '@'

[10]:
```
import re
```

[11]:
```
re.sub("@\w+","", "@Rahim this course rocks! http://rahimbaig.com/ai")
```

[11]:
```
' this course rocks! http://rahimbaig.com/ai'
```

[12]:
```
tweets_nouser = [re.sub("@\w+","", twt) for twt in tweets_lower]
```

[13]:
```
tweets_nouser[:5]
```

[13]:
```
['  when a father is dysfunctional and is so selfish he drags his kids into his
dysfunction.   #run',
 "  thanks for #lyft credit i can't use cause they don't offer wheelchair vans
in pdx.    #disapointed #getthanked",
 '  bihday your majesty',
 '#model   i love u take with u all the time in urð\x9f\x93±!!! ð\x9f\x98\x99ð\x
9f\x98\x8eð\x9f\x91\x84ð\x9f\x91\x85ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦  ',
```

```
 ' factsguide: society now    #motivation']
```

Remove URLs

[14]:
```
re.sub("\w+://\S+","", "@Rahim this course rocks! http://rahimbaig.com/ai")
```

[14]:
```
'@Rahim this course rocks! '
```

[15]:
```
tweets_nourl = [re.sub("\w+://\S+","", twt) for twt in tweets_nouser]
```

[16]:
```
tweets_nourl[:5]
```

[16]:
```
['  when a father is dysfunctional and is so selfish he drags his kids into his
dysfunction.   #run',
 "  thanks for #lyft credit i can't use cause they don't offer wheelchair vans
in pdx.    #disapointed #getthanked",
 '  bihday your majesty',
 '#model   i love u take with u all the time in urð\x9f\x93±!!! ð\x9f\x98\x99ð\x
9f\x98\x8eð\x9f\x91\x84ð\x9f\x91\x85ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦  ',
 ' factsguide: society now    #motivation']
```

Tokenze using Tweet Tokenizer from NLTK

[17]:
```
from nltk.tokenize import TweetTokenizer
```

[18]:
```
?TweetTokenizer()
```

```
Object `TweetTokenizer()` not found.
```

[19]:
```
tkn = TweetTokenizer()
```

[20]:
```
print(tkn.tokenize(tweets_nourl[0]))
```

```
['when', 'a', 'father', 'is', 'dysfunctional', 'and', 'is', 'so', 'selfish',
'he', 'drags', 'his', 'kids', 'into', 'his', 'dysfunction', '.', '#run']
```

[21]:
```
tweet_token = [tkn.tokenize(sent) for sent in tweets_nourl]
print(tweet_token[0])
```

```
['when', 'a', 'father', 'is', 'dysfunctional', 'and', 'is', 'so', 'selfish',
'he', 'drags', 'his', 'kids', 'into', 'his', 'dysfunction', '.', '#run']
```

0.1.1 Remove punctuations and stop words and other redundant terms tike 'rt', 'amp'

- Also remove hashtags

[22]:
```
from nltk.corpus import stopwords
from string import punctuation
```

[23]:
```
stop_nltk = stopwords.words("english")
stop_punct = list(punctuation)
```

[24]:
```
stop_punct.extend(['...','``',"''",".."])
```

[25]:
```
stop_context = ['rt', 'amp']
```

[26]:
```
stop_final = stop_nltk + stop_punct + stop_context
```

Function to

- remove stop words from a single tokenized sentence
- remove # tags
- remove terms with length = 1

[27]:
```
def del_stop(sent):
    return [re.sub("#","",term) for term in sent if ((term not in stop_final) &␣
↪(len(term)>1))]
```

[28]:
```
del_stop(tweet_token[4])
```

[28]:
```
['factsguide', 'society', 'motivation']
```

[29]:
```
tweets_clean = [del_stop(tweet) for tweet in tweet_token]
```

Check out the top terms in the tweets

[30]:
```
from collections import Counter
```

[31]:
```
term_list = []
for tweet in tweets_clean:
    term_list.extend(tweet)
```

[32]:
```
res = Counter(term_list)
res.most_common(10)
```

[32]:
```
[('love', 2748),
 ('day', 2276),
 ('happy', 1684),
 ('time', 1131),
 ('life', 1118),
 ("it's", 1058),
 ('like', 1047),
```

```
("i'm", 1018),
('today', 1013),
('new', 994)]
```

0.2 Data formatting for predictive modeling

Join the tokens back into strings

[33]:
```
tweets_clean[0]
```

[33]:
```
['father', 'dysfunctional', 'selfish', 'drags', 'kids', 'dysfunction', 'run']
```

[34]:
```
tweets_clean = [" ".join(tweet) for tweet in tweets_clean]
```

[35]:
```
tweets_clean[0]
```

[35]:
```
'father dysfunctional selfish drags kids dysfunction run'
```

0.2.1 Separate X and Y and perform train test split, 70-30

[36]:
```
len(tweets_clean)
```

[36]:
```
31962
```

[37]:
```
len(inp_tweets0.label)
```

[37]:
```
31962
```

[38]:
```
X = tweets_clean
y = inp_tweets0.label.values
```

Train test split

[39]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,␣
↪random_state=42)
```

0.2.2 Create a document term matrix using count vectorizer

[40]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[41]:
```
vectorizer = TfidfVectorizer(max_features = 5000)
```

[42]:
```
len(X_train), len(X_test)
```

[42]:
```
(22373, 9589)
```

[43]:
```
X_train_bow = vectorizer.fit_transform(X_train)

X_test_bow = vectorizer.transform(X_test)
```

[44]:
```
X_train_bow.shape, X_test_bow.shape
```

[44]:
```
((22373, 5000), (9589, 5000))
```

0.2.3 Model building

0.2.4 Using a *simple* Logistic Regression

[48]:
```
from sklearn.linear_model import LogisticRegression
```

[49]:
```
logreg = LogisticRegression()
```

[50]:
```
logreg.fit(X_train_bow, y_train)
```

[50]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

[51]:
```
y_train_pred = logreg.predict(X_train_bow)
y_test_pred = logreg.predict(X_test_bow)
```

[52]:
```
from sklearn.metrics import accuracy_score, classification_report
```

[53]:
```
accuracy_score(y_train, y_train_pred)
```

[53]:
```
0.9558396281231842
```

[54]:
```
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     20815
           1       0.96      0.38      0.55      1558

   micro avg       0.96      0.96      0.96     22373
   macro avg       0.96      0.69      0.76     22373
weighted avg       0.96      0.96      0.95     22373
```

Adjusting for class imbalance

[55]:
```
logreg = LogisticRegression(class_weight="balanced")
```

[56]:
```
logreg.fit(X_train_bow, y_train)
```

[56]:
```
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
          solver='warn', tol=0.0001, verbose=0, warm_start=False)
```

[57]:
```
y_train_pred = logreg.predict(X_train_bow)
y_test_pred = logreg.predict(X_test_bow)
```

[58]:
```
accuracy_score(y_train, y_train_pred)
```

[58]:
```
0.9528002503017029
```

[59]:
```
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.97     20815
           1       0.60      0.98      0.74      1558

   micro avg       0.95      0.95      0.95     22373
   macro avg       0.80      0.96      0.86     22373
weighted avg       0.97      0.95      0.96     22373
```

[60]:
```
from sklearn.model_selection import GridSearchCV, StratifiedKFold
```

[63]:
```
# Create the parameter grid based on the results of random search
param_grid = {
    'C': [0.01,0.1,1,10,100],
    'penalty': ["l1","l2"]
}
```

[64]:
```
?LogisticRegression()
```

```
Object `LogisticRegression()` not found.
```

[65]:
```
classifier_lr = LogisticRegression(class_weight="balanced")
```

[66]:
```
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = classifier_lr, param_grid = param_grid,
                           cv = StratifiedKFold(4), n_jobs = -1, verbose = 1,␣
 ↪scoring = "recall" )
```

[69]:
```
grid_search.fit(X_train_bow, y_train)
```

```
Fitting 4 folds for each of 10 candidates, totalling 40 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    2.4s finished
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

[69]:
```
GridSearchCV(cv=StratifiedKFold(n_splits=4, random_state=None, shuffle=False),
       error_score='raise-deprecating',
       estimator=LogisticRegression(C=1.0, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
          solver='warn', tol=0.0001, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='recall', verbose=1)
```

[71]:
```
grid_search.best_estimator_
```

[71]:
```
LogisticRegression(C=1, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
          solver='warn', tol=0.0001, verbose=0, warm_start=False)
```

0.2.5 Using the best estimator to make predictions on the test set

[72]:
```
y_test_pred = grid_search.best_estimator_.predict(X_test_bow)
```

[73]:
```
y_train_pred = grid_search.best_estimator_.predict(X_train_bow)
```

[74]:
```
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96      8905
           1       0.49      0.78      0.60       684

   micro avg       0.93      0.93      0.93      9589
   macro avg       0.74      0.86      0.78      9589
weighted avg       0.95      0.93      0.93      9589
```

[ ]: