Fake Review Prediction

July 29, 2021

[1]:
```
#importing all the required libraries
import pandas as pd
import numpy as np
import sklearn as sk
import pickle
from sklearn.feature_extraction.text import CountVectorizer
```

[2]:
```
df = pd.read_csv('deceptive-opinion.csv')
```

[3]:
```
df.head()
```

[3]:

	deceptive	hotel	polarity	source	text
0	truthful	conrad	positive	TripAdvisor	We stayed for a one night getaway with family …
1	truthful	hyatt	positive	TripAdvisor	Triple A rate with upgrade to view room was le…
2	truthful	hyatt	positive	TripAdvisor	This comes a little late as I'm finally catchi…
3	truthful	omni	positive	TripAdvisor	The Omni Chicago really delivers on all fronts…
4	truthful	hyatt	positive	TripAdvisor	I asked for a high floor away from the elevato…

[4]:
```
df.tail()#Extracting only the requireed features
df1 = df[['deceptive', 'text']]
df1
```

[4]:

	deceptive	text
0	truthful	We stayed for a one night getaway with family …
1	truthful	Triple A rate with upgrade to view room was le…
2	truthful	This comes a little late as I'm finally catchi…
3	truthful	The Omni Chicago really delivers on all fronts…
4	truthful	I asked for a high floor away from the elevato…
…	…	…
1595	deceptive	Problems started when I booked the InterContin…
1596	deceptive	The Amalfi Hotel has a beautiful website and i…

```
1597  deceptive  The Intercontinental Chicago Magnificent Mile …
1598  deceptive  The Palmer House Hilton, while it looks good i…
1599  deceptive  As a former Chicagoan, I'm appalled at the Ama…

[1600 rows x 2 columns]
```

[9]:
```
#filling the categorical variable deceptive with 0 for fake review and 1 for␣
 ↪real review
df1.loc[df1['deceptive'] == 'deceptive'][ 'deceptive'] = 0
df1.loc[df1['deceptive'] == 'truthful'][ 'deceptive'] = 1
```

[10]:
```
#Printing Dataframe1
df1
```

[10]:
```
     deceptive                                               text
0            1  We stayed for a one night getaway with family …
1            1  Triple A rate with upgrade to view room was le…
2            1  This comes a little late as I'm finally catchi…
3            1  The Omni Chicago really delivers on all fronts…
4            1  I asked for a high floor away from the elevato…
…          …                                                 …
1595         0  Problems started when I booked the InterContin…
1596         0  The Amalfi Hotel has a beautiful website and i…
1597         0  The Intercontinental Chicago Magnificent Mile …
1598         0  The Palmer House Hilton, while it looks good i…
1599         0  As a former Chicagoan, I'm appalled at the Ama…

[1600 rows x 2 columns]
```

[11]:
```
#Taking the input and output features seperately
X = df1['text']
Y = np.asarray(df1['deceptive'], dtype = int)
```

[12]:
```
#importing MultinomialNB
from sklearn.naive_bayes import MultinomialNB, GaussianNB
```

[13]:
```
#splitting the data into training and testing set  with test size is 30%
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.
 ↪3,random_state=109) # 70% training and 30% test
```

[22]:
```
X_test
```

[22]:
```
1063    We stayed at the Ritz Carlton two weeks prior,…
21      We went to Chicago to see an exhibit at the Ar…
1480    I recently stayed in The James Hotel in Chicag…
```

```
1215    Hyatt Regency Hotel: Good ole Downtown, Chicag…
459     Me and my husband got married here. We loved t…
                              …
133     Perfect location, clean and courteous staff al…
1252    If you want a 5-star hotel with 1-star service…
254     We had our hotel reservations at another hotel…
386     We became an Ambassador member just before spe…
1240    My experience as Fairmont Chicago Millennium P…
Name: text, Length: 480, dtype: object
```

[14]:
```
y_test
```

[14]:
```
array([1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0])
```

[15]:
```
nb = MultinomialNB()
```

[16]:
```
#Converting the review (text feature) to numerical features
cv = CountVectorizer()
x = cv.fit_transform(X_train)
y = cv.transform(X_test)
```

[17]:
```
# Fitting the model
nb.fit(x, y_train)
pickle.dump(nb,open('model.pkl','wb'))
model=pickle.load(open('model.pkl','rb'))
```

[18]:
```
nb.predict(y)
```

[18]:
```
array([1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0])
```

[19]:
```
# Training Accuracy
nb.score(x, y_train)
```

[19]:
```
0.9714285714285714
```

[22]:
```
# Testing Accuracy
nb.score(y, y_test)
```

[22]:
```
0.85625
```

[32]:
```
# Implementing with pywedge
import pywedge as pw
```

[33]:
```
#blm = pw.baseline_model(x, y_train)
```

[34]:
```
#blm.classification_summary()
```

[35]:
```
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel
```

[36]:
```
#Train the model using the training sets
clf.fit(x, y_train)
```

[36]:
```
SVC(kernel='linear')
```

[40]:
```
#Predict the response for test dataset
y_pred = clf.predict(y)
y_pred
```

[40]:
```
array([1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0])
```

[37]:
```
clf.score(x, y_train) #Training accuracy
```

[37]:
```
1.0
```

[38]:
```
clf.score(y, y_test)
```

[38]:
```
0.8166666666666667
```

[ ]:
```

```