Reviews

December 16, 2023

0.0.1 Data Gathering

Downloading the data

[1]:
```
!kaggle datasets download -d d4rklucif3r/restaurant-reviews
```

```
restaurant-reviews.zip: Skipping, found more recently modified local copy (use
--force to force download)
```

Extracting the compressed file

[2]:
```
from zipfile import ZipFile
dataset = ("C:\\Users\\tejus\\Desktop\\6th Sem Project\\Restaurant Reviews␣
 ↪(NLP)\\restaurant-reviews.zip")
with ZipFile(dataset, 'r') as file:
    file.extractall()
    print("The dataset is extracted")
```

```
The dataset is extracted
```

0.0.2 Importing the Dependencies

[3]:
```
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import re
```

Creating a dataframe

[4]:
```
tsv_file_path = ('C:\\Users\\tejus\\Desktop\\6th Sem Project\\Restaurant␣
 ↪Reviews (NLP)\\Restaurant_Reviews.tsv')
r_reviews = pd.read_csv(tsv_file_path, sep='\t')
```

[5]:
```
r_reviews.size
```

[5]:
```
2000
```

[6]:
```
r_reviews.shape
```

[6]:
```
(1000, 2)
```

[7]:
```
r_reviews.head()
```

[7]:
```
                                              Review  Liked
0                           Wow… Loved this place.      1
1                                 Crust is not good.      0
2          Not tasty and the texture was just nasty.      0
3  Stopped by during the late May bank holiday of…      1
4  The selection on the menu was great and so wer…      1
```

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

[8]:
```
Review    0
Liked     0
dtype: int64
```

[9]:
```
r_reviews['Liked'].value_counts()
```

[9]:
```
Liked
1    500
0    500
Name: count, dtype: int64
```

[10]:
```
r_reviews.head()
```

[10]:
```
                                              Review  Liked
0                           Wow… Loved this place.      1
1                                 Crust is not good.      0
2          Not tasty and the texture was just nasty.      0
3  Stopped by during the late May bank holiday of…      1
4  The selection on the menu was great and so wer…      1
```

0.0.3 Data Preprocessing

Stemming converts a word to its root word

[11]:
```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]', ' ', content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split(sep=' ')
    stemmed_content = [PorterStemmer().stem(word) for word in stemmed_content␣
 ↪if word not in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
```

```
    return stemmed_content
```

creating a new column for the stemmed reviews

[12]:
```
r_reviews['Stemmed_Reviews'] = r_reviews['Review'].apply(stemming)
```

[13]:
```
r_reviews.head()
```

[13]:
```
                                              Review  Liked  \
0                           Wow… Loved this place.      1
1                                 Crust is not good.      0
2          Not tasty and the texture was just nasty.      0
3  Stopped by during the late May bank holiday of…      1
4  The selection on the menu was great and so wer…      1

                                    Stemmed_Reviews
0                                 wow    love place
1                                        crust good
2                                tasti textur nasti
3  stop late may bank holiday rick steve recommen…
4                           select menu great price
```

0.0.4 Splitting the data into Training set and Testing set

[14]:
```
X = r_reviews['Stemmed_Reviews'].values
Y = r_reviews['Liked'].values
```

[15]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.
↪2,stratify=Y,random_state=2)
```

Converting words into vectors

[16]:
```
vectorizer = TfidfVectorizer()
```

Saving the vectorizer to be used in the app

[17]:
```
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

import pickle

filename = 'vectorizer.pkl'
pickle.dump(vectorizer, open(filename, 'wb'))
```

[18]:
```
print(X_train)
```

```
  (0, 892)      0.3292152335601234
  (0, 365)      0.3905782593861427
  (0, 30)       0.3905782593861427
```

```
(0, 1372)     0.28602868670508325
(0, 1369)     0.35186250062333335
(0, 300)      0.30656796649691337
(0, 391)      0.290499474797314
(0, 392)      0.30656796649691337
(0, 76)       0.2780357843366841
(0, 478)      0.176729367184008
(1, 311)      0.6913616616002672
(1, 834)      0.651273787246651
(1, 478)      0.31282823867854653
(2, 1168)     0.508602366570897
(2, 566)      0.4174849217841828
(2, 680)      0.42869679219480356
(2, 615)      0.508602366570897
(2, 239)      0.35294853149107464
(3, 1131)     0.5525769612282082
(3, 1250)     0.5205363708646262
(3, 1204)     0.5525769612282082
(3, 107)      0.34403385072792303
(4, 72)       0.3256194422164578
(4, 142)      0.3256194422164578
(4, 1108)     0.29135594162429207
:     :
(795, 1190)   0.3053019226759833
(795, 1229)   0.3402973136508531
(795, 985)    0.255638003441639
(795, 590)    0.3313973884561668
(795, 513)    0.26528290970241614
(795, 914)    0.18205428020544953
(795, 525)    0.19845710492961235
(796, 569)    0.792390353732586
(796, 745)    0.6100143664796324
(797, 1320)   0.6668466950220887
(797, 977)    0.5794663922337555
(797, 1057)   0.468544753048962
(798, 1001)   0.3768528122914655
(798, 417)    0.35500140075386427
(798, 897)    0.33949757753596876
(798, 1277)   0.35500140075386427
(798, 54)     0.2901166432215648
(798, 145)    0.33949757753596876
(798, 874)    0.33949757753596876
(798, 693)    0.20905858539086558
(798, 1322)   0.25276140846606804
(798, 239)    0.26151991305768607
(799, 1116)   0.6556077434275575
(799, 1073)   0.5526064269822653
(799, 146)    0.514591705739337
```

[19]:
```
print(X_test)
```

```
  (0, 1012)     1.0
  (1, 1322)     1.0
  (2, 1236)     0.423284623964311
  (2, 1076)     0.3760578251624208
  (2, 291)      0.6227159389161827
  (2, 76)       0.5400328681378932
  (3, 74)       1.0
  (4, 1225)     0.5199124394067061
  (4, 492)      0.40808338783594494
  (4, 233)      0.41684062678974443
  (4, 207)      0.48976586677556594
  (4, 163)      0.38669405415860425
  (5, 239)      0.7810980019325923
  (5, 74)       0.6244084491556086
  (6, 1292)     0.4633955888426175
  (6, 913)      0.37961404738057924
  (6, 525)      0.2621377899243931
  (6, 467)      0.5332730613009541
  (6, 163)      0.39663125253043513
  (6, 23)       0.3615867447417573
  (7, 1175)     0.7366661152982865
  (7, 538)      0.1998466746585458
  (7, 525)      0.18105923718083705
  (7, 426)      0.36833305764914326
  (7, 414)      0.3104649342041163
  :     :
  (193, 406)    0.5445116292761637
  (193, 397)    0.5129386986511408
  (194, 1177)   0.8619831211522782
  (194, 1076)   0.5069369771959595
  (195, 1346)   0.36547429159010664
  (195, 1236)   0.2696614628437105
  (195, 1078)   0.43539042746189216
  (195, 1075)   0.33957759871549603
  (195, 786)    0.35392845770798087
  (195, 582)    0.41996800674984397
  (195, 402)    0.33957759871549603
  (195, 74)     0.2681082660987881
  (196, 914)    0.6391742977803287
  (196, 538)    0.7690619071680924
  (197, 1107)   0.6185806414059921
  (197, 1076)   0.398311190032479
  (197, 61)     0.6772785143297118
  (198, 1150)   0.39111119944149897
  (198, 1061)   0.4872094265555146
  (198, 918)    0.517198642547495
```

```
(198, 404)    0.33984429608443073
(198, 107)    0.32200734571230993
(198, 23)     0.35068745660501993
(199, 1073)   0.6962561726198929
(199, 604)    0.7177933839822555
```

0.0.5 Model creation and Training

[20]:
```
model = LogisticRegression(max_iter=20)
```

Training the Model

[21]:
```
model.fit(X_train,Y_train)
```

[21]:
```
LogisticRegression(max_iter=20)
```

0.0.6 Evaluating the performance of the Model

[22]:
```
X_train_prediction = model.predict(X_train)
Accuracy = accuracy_score(Y_train,X_train_prediction)
print(f"The accuracy on training data is {Accuracy}")
```

```
The accuracy on training data is 0.95125
```

[23]:
```
X_test_prediction = model.predict(X_test)
Accuracy = accuracy_score(Y_test,X_test_prediction)
print(f"The accuracy on test data is {Accuracy}")
```

```
The accuracy on test data is 0.775
```

0.0.7 Saving the model for further use

[24]:
```
import pickle

filename = 'trained_model.sav'
pickle.dump(model, open(filename, 'wb'))
```