Natural-Language-Processing

January 18, 2025

[ ]:
```
#   Classifying Restaurant's Reviews if Positive or Negative
```

[2]:
```
#   Importing Libraries -->

import pandas as pd
import numpy as np
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪confusion_matrix
from sklearn.feature_extraction.text import CountVectorizer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\krish\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[3]:
```
#   Importing Dataset -->

data = pd.read_csv('Data/Restaurant_Reviews.tsv', delimiter='\t', quoting=3)
data.head(10)
```

[3]:

	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
5	Now I am getting angry and I want my damn pho.	0
6	Honeslty it didn't taste THAT fresh.)	0
7	The potatoes were like rubber and you could te…	0
8	The fries were great too.	1
9	A great touch.	1

[4]:
```
#   Cleaning The Text -->

corpus = []

for i in range(0,1000):
    review = re.sub('[^a-zA-Z]', ' ', data['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    all_sw = stopwords.words('english')
    all_sw.remove('not')
    review = [ps.stem(word) for word in review if not word in set(all_sw)]
    review = ' '.join(review)
    corpus.append(review)

for i in range(0,11):
    print(corpus[i])
```

```
wow love place
crust not good
not tasti textur nasti
stop late may bank holiday rick steve recommend love
select menu great price
get angri want damn pho
honeslti tast fresh
potato like rubber could tell made ahead time kept warmer
fri great
great touch
servic prompt
```

[5]:
```
#   Creating Bag Of Words Model -->

cv = CountVectorizer(max_features=1500)
x_data = cv.fit_transform(corpus).toarray()
y_data = data.iloc[:, -1].values
```

[6]:
```
#   Splitting The Dataset -->

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.
↪2, random_state=42)
```

[7]:
```
#   Training Logistic Regression Model On Training Set -->

model = LogisticRegression(random_state=42)
model.fit(x_train, y_train)
```

[7]:
```
LogisticRegression(random_state=42)
```

[8]:
```
#   Predicting Results -->

y_pred = model.predict(x_test)
```

[9]:
```
#   Checking Accuracy -->

acc_score = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

[10]:
```
print("Accuracy Score --> ", acc_score)
```

```
Accuracy Score -->  0.765
```

[11]:
```
print("Confusion Matrix -->\n\n", conf_matrix)
```

```
Confusion Matrix -->

 [[81 15]
 [32 72]]
```

[12]:
```
print("Classification Report -->\n\n", class_report)
```

```
Classification Report -->

              precision    recall  f1-score   support

           0       0.72      0.84      0.78        96
           1       0.83      0.69      0.75       104

    accuracy                           0.77       200
   macro avg       0.77      0.77      0.76       200
weighted avg       0.77      0.77      0.76       200
```