

Lab7 : Sentiment Analysis on Movie Reviews

SURIYA S

225229140

Exercise-1

In [21]: `import pandas as pd`

In [22]: `df = pd.read_csv("train.tsv", sep='\t')`

In [23]: `df.head()`

Out[23]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

In [24]: `df.shape`

Out[24]: (156060, 4)

In [25]: `df.describe()`

Out[25]:

	PhraseId	SentenceId	Sentiment
count	156060.000000	156060.000000	156060.000000
mean	78030.500000	4079.732744	2.063578
std	45050.785842	2502.764394	0.893832
min	1.000000	1.000000	0.000000
25%	39015.750000	1861.750000	2.000000
50%	78030.500000	4017.000000	2.000000
75%	117045.250000	6244.000000	3.000000
max	156060.000000	8544.000000	4.000000

In [26]: `df.columns`

Out[26]: Index(['PhraseId', 'SentenceId', 'Phrase', 'Sentiment'], dtype='object')

```
In [27]: df['Sentiment'].value_counts()
```

```
Out[27]: 2    79582
         3    32927
         1    27273
         4     9206
         0     7072
         Name: Sentiment, dtype: int64
```

Exercise-2

```
In [28]: zero = df.loc[df.Sentiment == 0]
         one = df.loc[df.Sentiment == 1]
         two = df.loc[df.Sentiment == 2]
         three = df.loc[df.Sentiment == 3]
         four = df.loc[df.Sentiment == 4]
```

```
In [29]: small_rotten_train = pd.concat([zero[:200], one[:200], two[:200], three[:200], four[:200]])
```

Exercise-3

1.open the file. "small_rotten_train.csv"

```
In [30]: small_rotten_train.to_csv("small_rotten_train.csv")
```

2. The reivew text are stored in "Phrase"

```
In [31]: X = small_rotten_train.Phrase
```

3.The "Sentiment" columns is your target, say "y"

```
In [32]: y = small_rotten_train.Sentiment
```

```
In [33]: import nltk
         from nltk.corpus import stopwords
         nltk.download('stopwords')
         nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
```

```
Out[33]: True
```

4. Pre-processing

```
In [34]: stop_words = set(stopwords.words('english'))
```

```
In [39]: from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

```
In [40]: def clean_review(review):
tokens = review.lower().split()
filtered_tokens = [lemmatizer.lemmatize(w)
                    for w in tokens if w not in stop_words]
return " ".join(filtered_tokens)
```

5. Apply the above function to X

```
In [43]: import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data] C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
```

```
Out[43]: True
```

```
In [44]: t = X.tolist()
f = []
```

```
In [45]: for i in t:
f.append(clean_review(i))
n = pd.Series(f)
```

6. Split X and Y for Trainig and testing (Use 20% for testing)

```
In [46]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(n,y,test_size=0.20,random_state=42)
```

7.Create tfidfVectorizer as below and perform vectorization on X_train using fit_perform() method

```
In [47]: from sklearn.feature_extraction.text import TfidfVectorizer
TfidfVectorizer(min_df =3,max_features =None,
                ngram_range = (1,2), use_idf=1)
```

```
Out[47]: TfidfVectorizer(min_df=3, ngram_range=(1, 2), use_idf=1)
```

```
In [48]: from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
```

```
In [49]: X_train_NB = cv.fit_transform(X_train)
X_test_NB = cv.transform(X_test)
```

8. Create MultinomialNB model and perform training using X_train_lemmatized and y_train.

```
In [50]: from sklearn.naive_bayes import MultinomialNB
```

```
In [51]: mb = MultinomialNB()
mb.fit(X_train_NB,y_train)
```

```
Out[51]: MultinomialNB()
```

9.Validation on X_test lemmatized and predict output

```
In [52]: y_pred_NB= mb.predict(X_test_NB)
```

10.Classification_report and Accuracy_score

```
In [53]: from sklearn.metrics import accuracy_score,classification_report
```

```
In [54]: acc = accuracy_score(y_test,y_pred_NB)
print("Accuracy score :",acc)
```

Accuracy score : 0.67

```
In [55]: print("Classification Report :\n",classification_report(y_test,y_pred_NB))
```

```
Classification Report :
              precision    recall  f1-score   support

     0       0.71      0.76      0.74         33
     1       0.70      0.67      0.68         48
     2       0.62      0.57      0.59         37
     3       0.60      0.66      0.62         38
     4       0.72      0.70      0.71         44

 accuracy                   0.67         200
 macro avg       0.67      0.67      0.67         200
 weighted avg    0.67      0.67      0.67         200
```

Exercise -4

1.open "rotten_tomato_test.tsv" file into Dataframe

```
In [56]: df1 = pd.read_csv("test.tsv",sep='\t')
```

```
In [57]: df1.head()
```

```
Out[57]:
```

	Phraseld	Sentenceld	Phrase
0	156061	8545	An intermittently pleasing but mostly routine ...
1	156062	8545	An intermittently pleasing but mostly routine ...
2	156063	8545	An
3	156064	8545	intermittently pleasing but mostly routine effort
4	156065	8545	intermittently pleasing but mostly routine

```
In [58]: X2 = df1["Phrase"]
```

2. Clean this test data, using the function `clean_review()`, as before

```
In [59]: X2 = X2.apply(lambda X2: clean_review(X2))
```

3. build TFIDF values using `transform()` method

```
In [60]: X2_test = cv.transform(X2)
```

4. Perform using `predict()` method

```
In [61]: y_pred_2 = mb.predict(X2_test)
```

```
In [62]: y_pred_2
```

```
Out[62]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [ ]:
```