LP2.

Assignment 4

Date of Completion:- 24.9.2020

Date of Submission:- 29.9.2020

Title:- Text analysis.

Problem Statement:- Consider a suitable text dataset. Remove stop words, apply stemming and feature selection techniques to represent documents as vectors. Classify documents and evaluate precision recall.

Learning Objectives:- To understand the process of stemming, calculating precision and recall.

Learning Outcomes:- Students will be able to understand stop words, stemming, feature selection techniques and calculate precision and recall.

Software / Hardware requirements :- ~~NLKT~~ ~~NLKL~~ NLTK data / packages, python, Anaconda IDE.

Theory :-

Stop Words

1) The most commonly used words in a language.
2) These words are filtered out before or after the

natural language data (txt) is processed.
3) They do not add much meaning to a sentence.
Eg. as, the, at, a etc.

Stemming:-
1) A process of producing morphological variant of the root/base word.
2) A word is reduced to its root form.
Eg loved → love
played → play
eating → eat.

Precision and Recall:-
1) precision = $\frac{\text{Number of correct triples}}{\text{Number of triples retrieved}} = \frac{Tp}{Tp+fp}$ Tp ⇒ true positive, Fp ⇒ false positive
2) recall = $\frac{\text{Number of correct triples}}{\text{Number of triples in gold set}} = \frac{Tp}{Tp+f_n}$ f_n ⇒ false negative
3) A measure of success of prediction when classes are very imbalance.
4) Precision is measure of result relevancy
5) Recall is measure of how many truly relevant results are returned.

Algorithm:-
1. Import python different packages numpy, pandas, nltk (natural language toolkit), re(RegEx)
2. read the dataset.
3. remove the stopwords from the dataset and perform stemming on the words and form a

corpus (dataset with no stop word and stemmed words)
4. Prepare or vectorize the words. dataset (feature extraction)
5. Split the data in training and test set.
6. Fit classifier model on the dataset (Naive Bayes)
7. Predict value on the test data.
8. Build a confusion matrix
9. Calculate precision and recall.

Conclusion:-

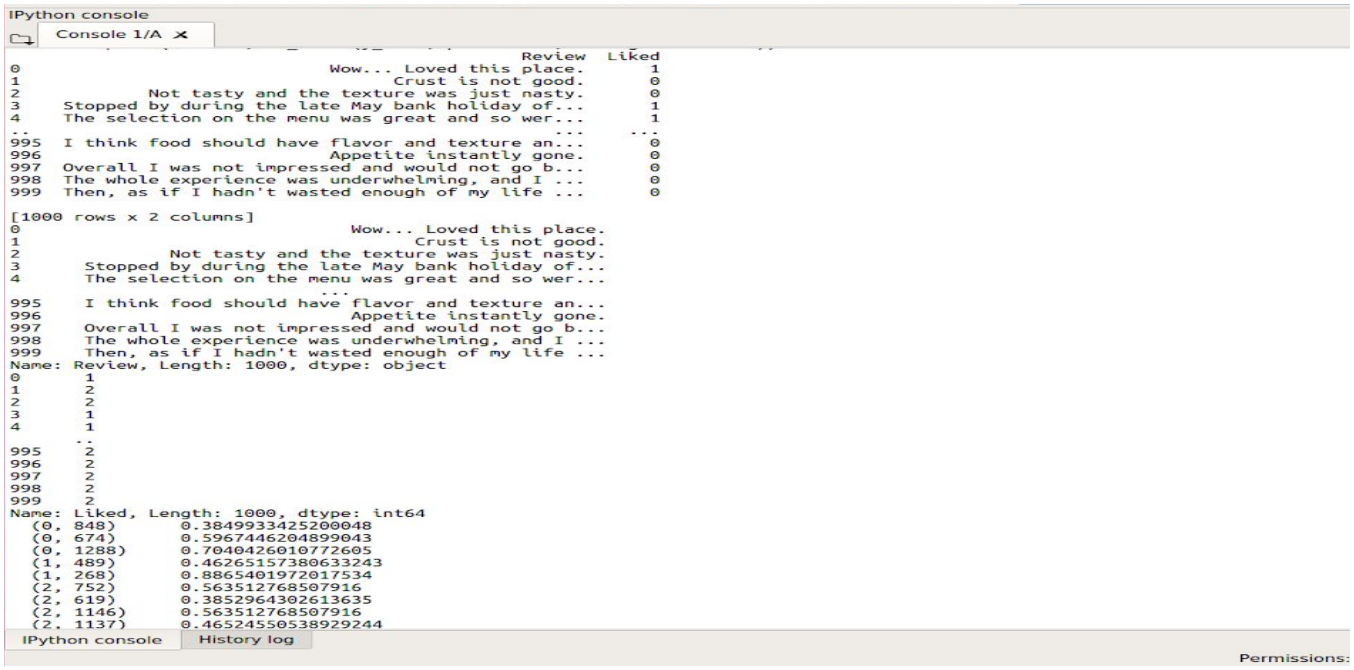
Dataset Used:- Restaurant Reviews.

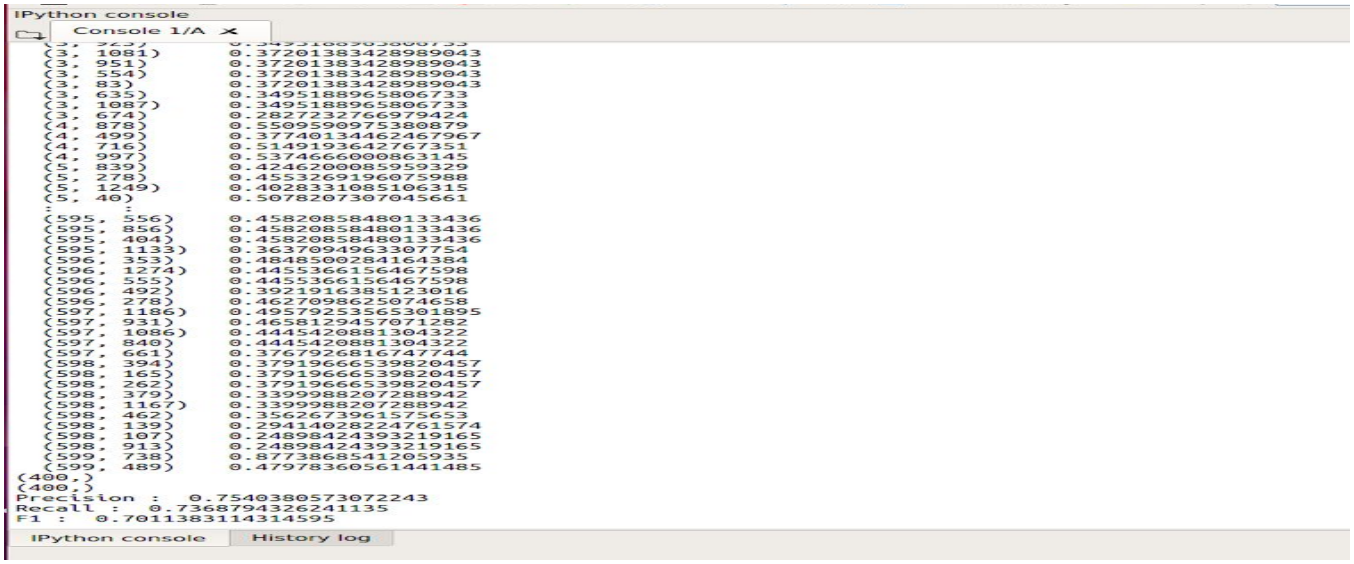
Conclusion:- Thus I have completed this assignment and understood how to calculate precision recall and analyze test.

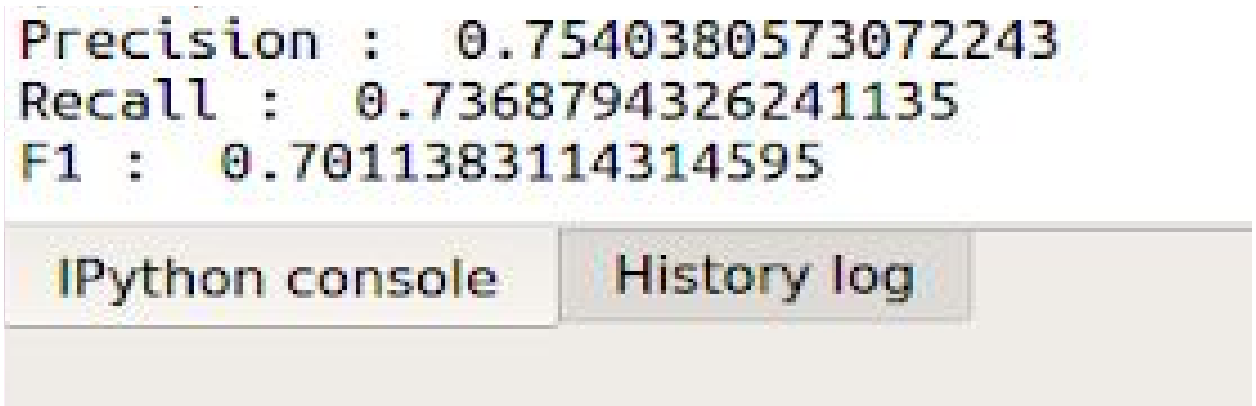
CODE:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_score, recall_score, f1_score
import pandas as pd
import numpy as np

dataset=pd.read_csv('Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)
print(dataset)

dataset.head()

dataset['Liked']=dataset['Liked'].map({1:1,0:2})

x = dataset['Review']
y = dataset['Liked']
print(x)
print(y)

vectorizer=TfidfVectorizer(stop_words = 'english')
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size = 0.4, shuffle = False)
x_train_vec = vectorizer.fit_transform(x_train)
print(x_train_vec)
x_test_vec = vectorizer.transform(x_test)
nb = MultinomialNB()
nb.fit(x_train_vec, y_train)
nb.score(x_test_vec, y_test)
prediction = nb.predict(x_test_vec)
print(prediction.shape)
print(y_test.shape)
np.array(y_test)
print('Precision : ', precision_score(y_test, prediction, average = 'macro'))
print('Recall : ', recall_score(y_test, prediction, average = 'macro'))
print('F1 : ', f1_score(y_test, prediction, average = 'macro'))
```

OUTPUT:


IPython console
Console 1/A
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
.. ..
995 I think food should have flavor and texture an... 0
996 Appetite instantly gone. 0
997 Overall I was not impressed and would not go b... 0
998 The whole experience was underwhelming, and I ... 0
999 Then, as if I hadn't wasted enough of my life ... 0
[1000 rows x 2 columns]
0 Wow... Loved this place.
1 Crust is not good.
2 Not tasty and the texture was just nasty.
3 Stopped by during the late May bank holiday of...
4 The selection on the menu was great and so wer...
...
995 I think food should have flavor and texture an...
996 Appetite instantly gone.
997 Overall I was not impressed and would not go b...
998 The whole experience was underwhelming, and I ...
999 Then, as if I hadn't wasted enough of my life ...
Name: Review, Length: 1000, dtype: object
0 1
1 2
2 2
3 1
4 1
..
995 2
996 2
997 2
998 2
999 2
Name: Liked, Length: 1000, dtype: int64
(0, 848) 0.3849933425200048
(0, 674) 0.5967446204899043
(0, 1288) 0.7040426010772605
(1, 489) 0.46265157380633243
(1, 268) 0.8865401972017534
(2, 752) 0.563512768507916
(2, 619) 0.3852964302613635
(2, 1146) 0.563512768507916
(2, 1137) 0.46524550538929244
IPython console
History log
Permissions:


IPython console
Console 1/A
(3, 1081) 0.3720138342898904 3
(3, 951) 0.3720138342898904 3
(3, 554) 0.3720138342898904 3
(3, 83) 0.3720138342898904 3
(3, 635) 0.3495188965806733
(3, 1087) 0.3495188965806733
(3, 674) 0.2827232766979424
(4, 878) 0.5509590975380879
(4, 499) 0.37740134462467967
(4, 716) 0.5149193642767351
(4, 997) 0.5374666000863145
(5, 839) 0.4246200085959329
(5, 278) 0.4553269196975988
(5, 1249) 0.40283310851063 15
(5, 40) 0.5078207307045661
(595, 556) 0.4582085848013343 6
(595, 856) 0.4582085848013343 6
(595, 404) 0.4582085848013343 6
(595, 1133) 0.3637094963307754
(596, 353) 0.4848500284164384
(596, 1274) 0.4455366156467598
(596, 555) 0.4455366156467598
(596, 492) 0.3922191638512301 6
(596, 278) 0.4627098625074658
(597, 1186) 0.4957925356530189 5
(597, 931) 0.4658129457071282
(597, 1086) 0.4445420881304322
(597, 840) 0.4445420881304322
(597, 661) 0.3767926816747744
(598, 394) 0.3791966653982045 7
(598, 165) 0.3791966653982045 7
(598, 262) 0.3791966653982045 7
(598, 379) 0.3399988207288942
(598, 1167) 0.3399988207288942
(598, 467) 0.3562673961575653
(598, 139) 0.2941402822247615 74
(598, 107) 0.2489842439321916 5
(598, 913) 0.2489842439321916 5
(599, 738) 0.8773866541205935
(599, 489) 0.4797836056144148 5
(400,)
(400,)
Precision : 0.7540380573072243
Recall : 0.7368794326241135
F1 : 0.7011383114314595
IPython console
History log


Precision : 0.7540380573072243
Recall : 0.7368794326241135
F1 : 0.7011383114314595
IPython console
History log