ABHINAV RANJAN
RA1911003010003
CSE A1 SECTION
SRMIST , KTR

AI LAB EXP 9
NATURAL LANGUAGE PROGRAMMING

PROBLEM STATEMENT :

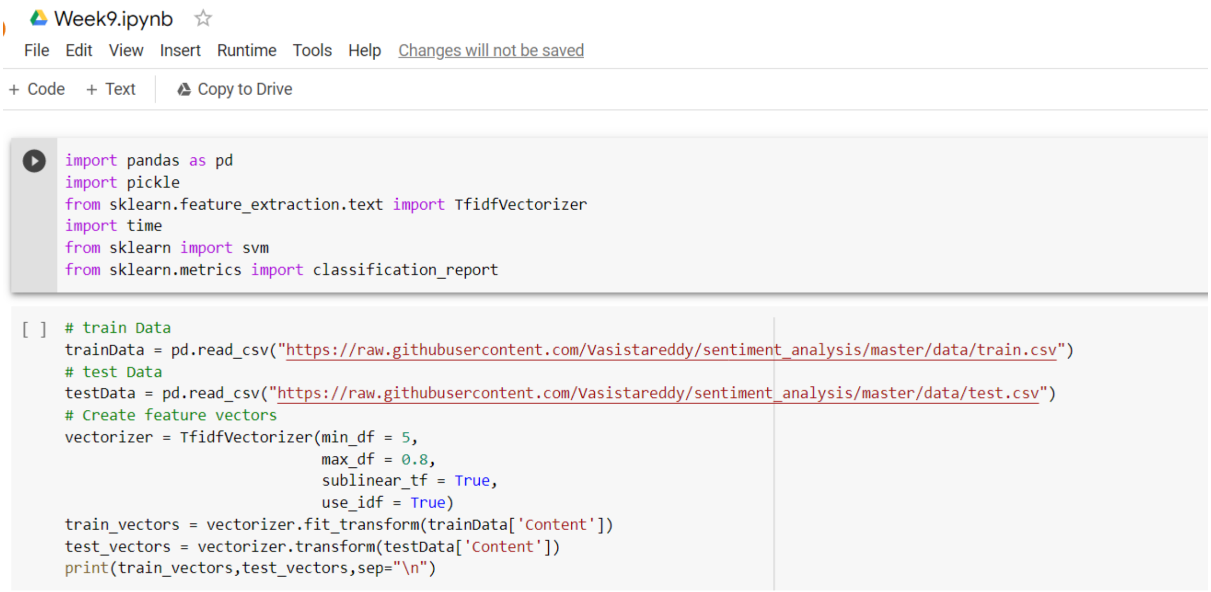
To implement sentiment analysis on a given text message and check if it is a negative statement

TOOLS USED : python3 , colab , pandas

ALGORITHM :

1. Load the dataset using pandas
2. Vectorize the dataset using TFIDF vectorizer
3. Split the dataset into train and test data
4. Split the text into separate tokens using tokenizer
5. Train a SVM model to classify text
6. Train the data and keep improving the model's accuracy
7. Save model through pickle and run from file directly

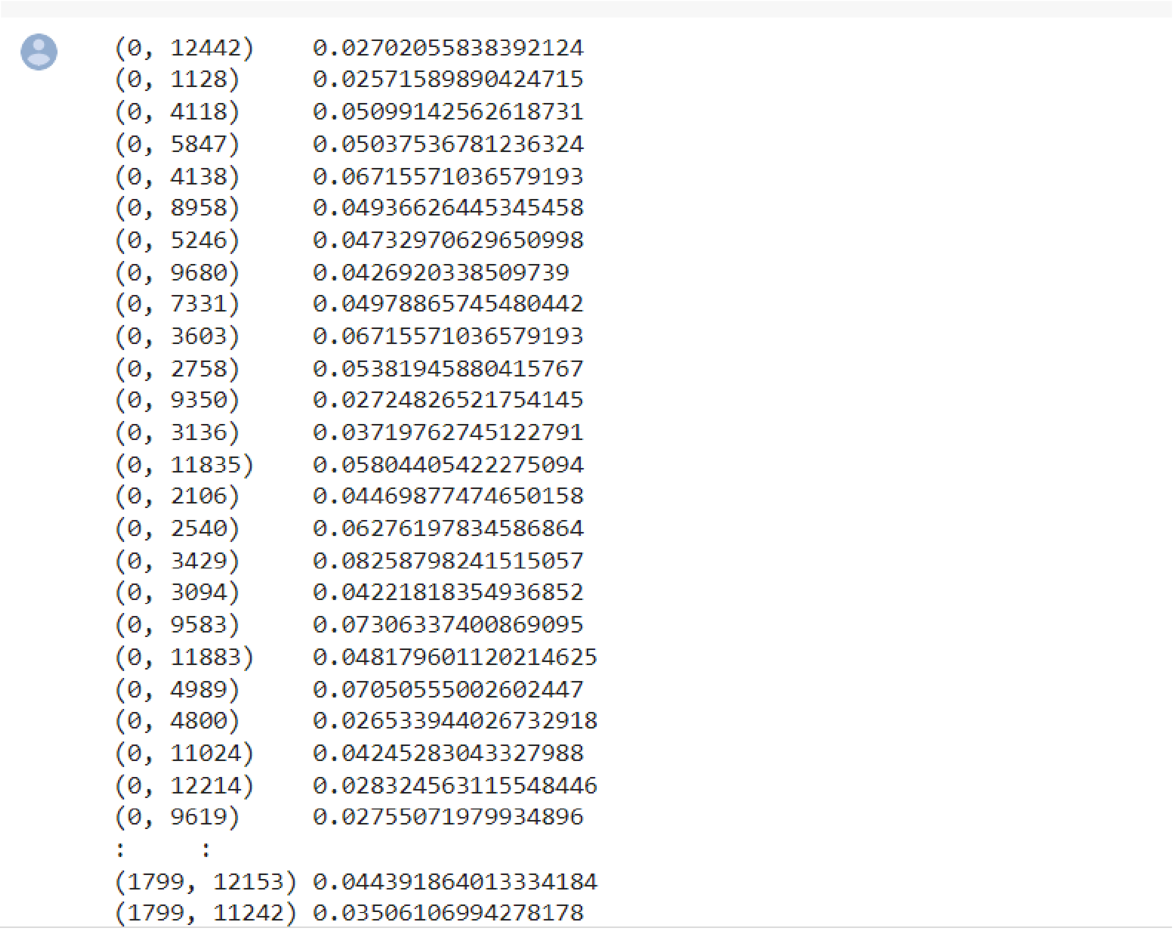
CODE :



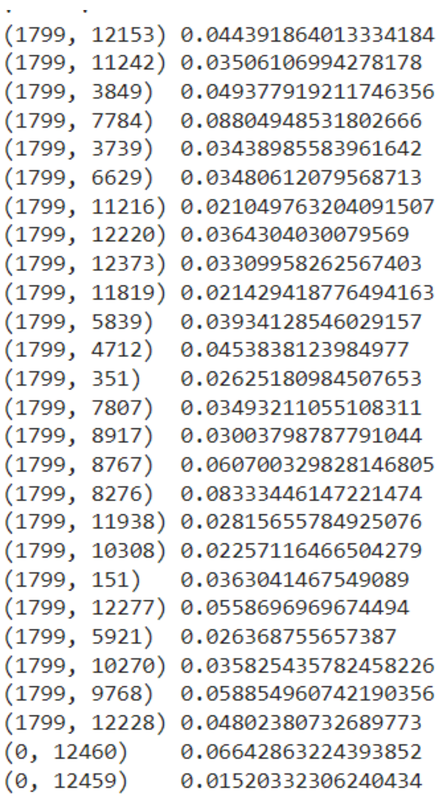
Week9.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text
Copy to Drive

```
import pandas as pd
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
import time
from sklearn import svm
from sklearn.metrics import classification_report
```

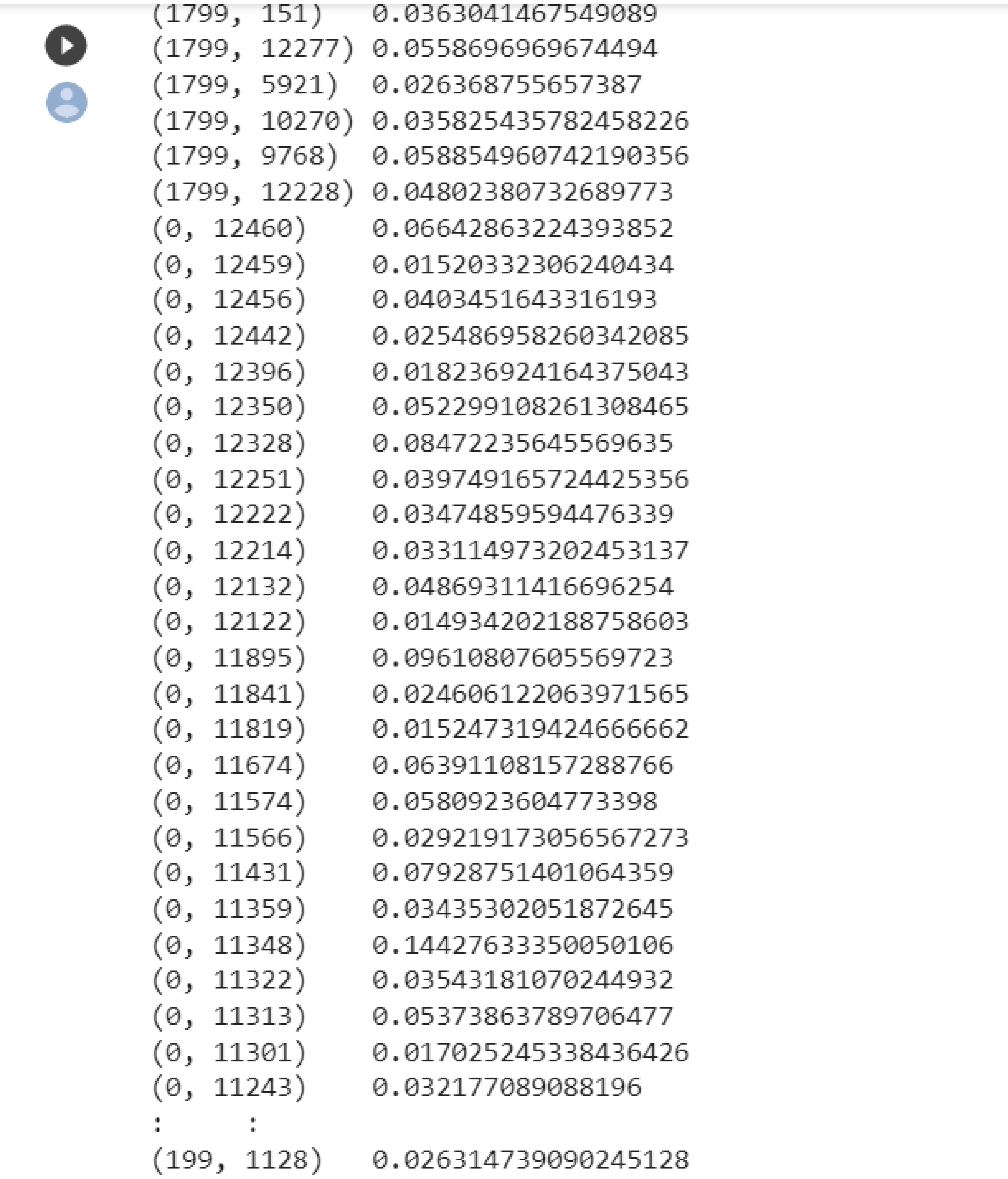
```
[ ] # train Data
    trainData = pd.read_csv("https://raw.githubusercontent.com/Vasistareddy/sentiment_analysis/master/data/train.csv")
    # test Data
    testData = pd.read_csv("https://raw.githubusercontent.com/Vasistareddy/sentiment_analysis/master/data/test.csv")
    # Create feature vectors
    vectorizer = TfidfVectorizer(min_df = 5,
                                 max_df = 0.8,
                                 sublinear_tf = True,
                                 use_idf = True)
    train_vectors = vectorizer.fit_transform(trainData['Content'])
    test_vectors = vectorizer.transform(testData['Content'])
    print(train_vectors,test_vectors,sep="\n")
```



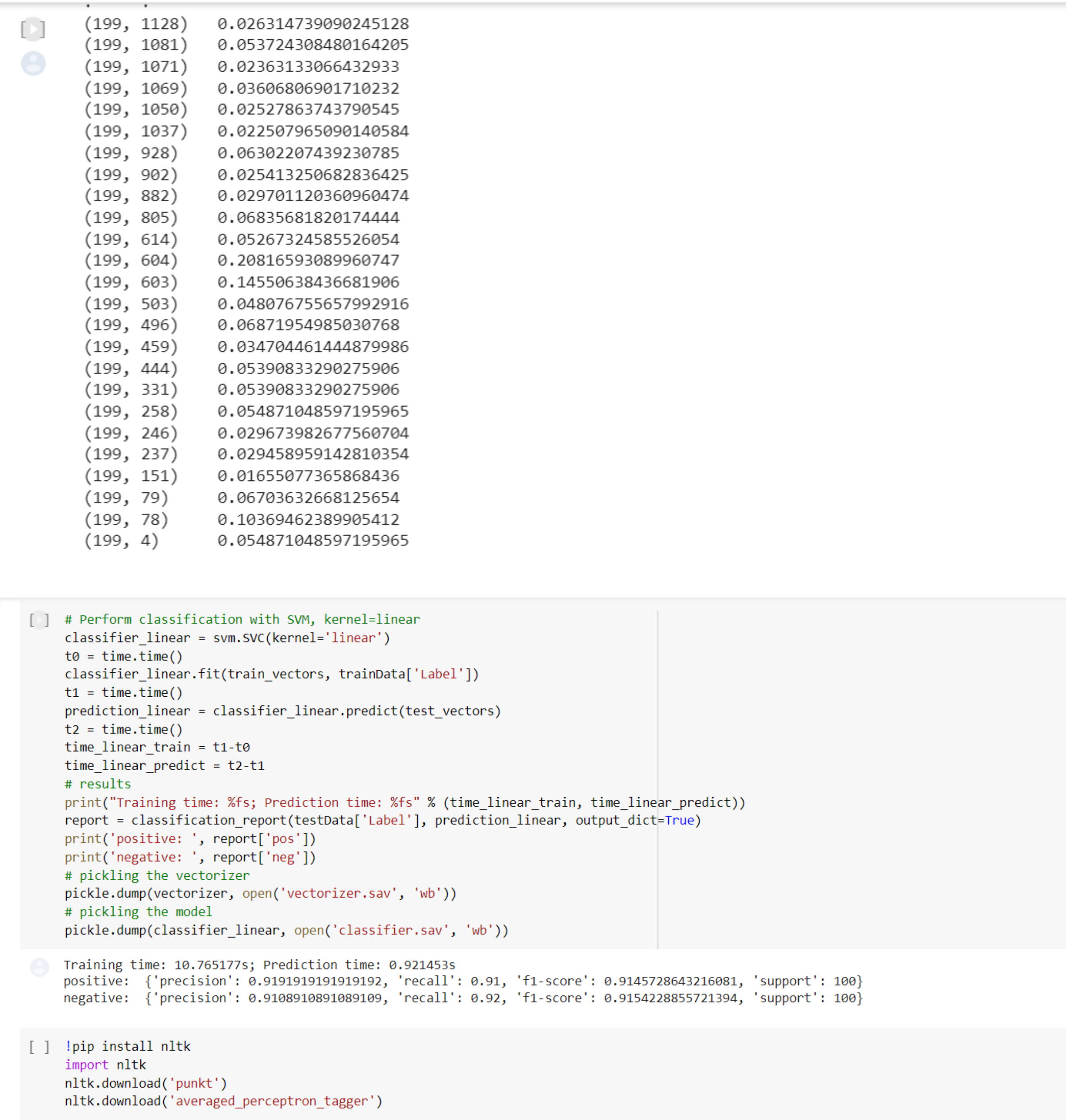
```
  (0, 12442)	0.027020558383921 24
  (0, 1128)	0.025715898904247 15
  (0, 4118)	0.05099142562618731
  (0, 5847)	0.05037536781236324
  (0, 4138)	0.0671557103657919 3
  (0, 8958)	0.04936626445345458
  (0, 5246)	0.04732970629650998
  (0, 9680)	0.042692033850973 9
  (0, 7331)	0.04978865745480442
  (0, 3603)	0.0671557103657919 3
  (0, 2758)	0.05381945880415767
  (0, 9350)	0.02724826521754145
  (0, 3136)	0.03719762745122791
  (0, 11835)	0.05804405422275094
  (0, 2106)	0.04469877474650158
  (0, 2540)	0.06276197834586864
  (0, 3429)	0.08258798241515057
  (0, 3094)	0.04221818354936852
  (0, 9583)	0.07306337400869095
  (0, 11883)	0.048179601120214625
  (0, 4989)	0.07050555002602447
  (0, 4800)	0.026533944026732918
  (0, 11024)	0.04245283043327988
  (0, 12214)	0.028324563115548446
  (0, 9619)	0.02755071979934896
  :	:
  (1799, 12153)	0.044391864013334184
  (1799, 11242)	0.03506106994278178
```



```
(1799, 12153) 0.04439186401334184
(1799, 11242) 0.03506106994278178
(1799, 3849)  0.049377919211746356
(1799, 7784)  0.08804948531802666
(1799, 3739)  0.03438985583961642
(1799, 6629)  0.03480612079568713
(1799, 11216) 0.021049763204091507
(1799, 12220) 0.0364304030079569
(1799, 12373) 0.03309958262567403
(1799, 11819) 0.021429418776494163
(1799, 5839)  0.03934128546029157
(1799, 4712)  0.0453838123984977
(1799, 351)   0.02625180984507653
(1799, 7807)  0.03493211055108311
(1799, 8917)  0.03003798787791044
(1799, 8767)  0.060700329828146805
(1799, 8276)  0.08333446147221474
(1799, 11938) 0.02815655784925076
(1799, 10308) 0.02257116466504279
(1799, 151)   0.0363041467549089
(1799, 12277) 0.0558696969674494
(1799, 5921)  0.026368755657387
(1799, 10270) 0.035825435782458226
(1799, 9768)  0.058854960742190356
(1799, 12228) 0.04802380732689773
(0, 12460)    0.06642863224393852
(0, 12459)    0.01520332306240434
```



```
(1799, 151)   0.0363041467549089
(1799, 12277) 0.0558696969674494
(1799, 5921)  0.026368755657387
(1799, 10270) 0.035825435782458226
(1799, 9768)  0.058854960742190356
(1799, 12228) 0.0480238073268977З
(0, 12460)    0.06642863224393852
(0, 12459)    0.01520332306240434
(0, 12456)    0.0403451643316193
(0, 12442)    0.025486958260342085
(0, 12396)    0.018236924164375043
(0, 12350)    0.052299108261308465
(0, 12328)    0.08472235645569635
(0, 12251)    0.039749165724425356
(0, 12222)    0.03474859594476339
(0, 12214)    0.033114973202453137
(0, 12132)    0.04869311416696254
(0, 12122)    0.014934202188758603
(0, 11895)    0.09610807605569723
(0, 11841)    0.024606122063971565
(0, 11819)    0.015247319424666662
(0, 11674)    0.06391108157288766
(0, 11574)    0.0580923604773398
(0, 11566)    0.029219173056567273
(0, 11431)    0.07928751401064359
(0, 11359)    0.03435302051872645
(0, 11348)    0.14427633350050106
(0, 11322)    0.03543181070244932
(0, 11313)    0.05373863789706477
(0, 11301)    0.017025245338436426
(0, 11243)    0.032177089088196
:     :
(199, 1128)   0.026314739090245128
```

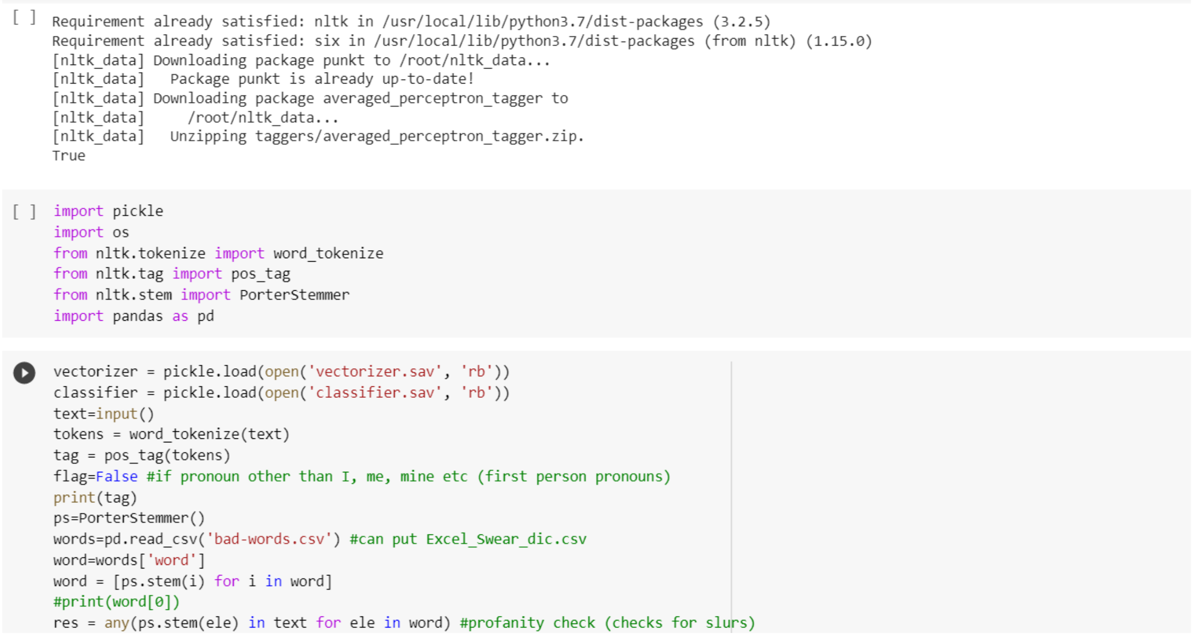


```
  (199, 1128)	0.026314739090245128
  (199, 1081)	0.053724308480164205
  (199, 1071)	0.02363133066432933
  (199, 1069)	0.03606806901710232
  (199, 1050)	0.02527863743790545
  (199, 1037)	0.022507965090140584
  (199, 928)	0.06302207439230785
  (199, 902)	0.025413250682836425
  (199, 882)	0.029701120360960474
  (199, 805)	0.06835681820174444
  (199, 614)	0.05267324585526054
  (199, 604)	0.20816593089960747
  (199, 603)	0.14550638436681906
  (199, 503)	0.048076755657992916
  (199, 496)	0.06871954985030768
  (199, 459)	0.034704461444879986
  (199, 444)	0.05390833290275906
  (199, 331)	0.05390833290275906
  (199, 258)	0.054871048597195965
  (199, 246)	0.029673982677560704
  (199, 237)	0.029458959142810354
  (199, 151)	0.01655077365868436
  (199, 79)	0.06703632668125654
  (199, 78)	0.10369462389905412
  (199, 4)	0.054871048597195965
```

```python
# Perform classification with SVM, kernel=linear
classifier_linear = svm.SVC(kernel='linear')
t0 = time.time()
classifier_linear.fit(train_vectors, trainData['Label'])
t1 = time.time()
prediction_linear = classifier_linear.predict(test_vectors)
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_linear_predict))
report = classification_report(testData['Label'], prediction_linear, output_dict=True)
print('positive: ', report['pos'])
print('negative: ', report['neg'])
# pickling the vectorizer
pickle.dump(vectorizer, open('vectorizer.sav', 'wb'))
# pickling the model
pickle.dump(classifier_linear, open('classifier.sav', 'wb'))
```

```
Training time: 10.765177s; Prediction time: 0.921453s
positive:  {'precision': 0.9191919191919192, 'recall': 0.91, 'f1-score': 0.9145728643216081, 'support': 100}
negative:  {'precision': 0.9108910891089109, 'recall': 0.92, 'f1-score': 0.9154228855721394, 'support': 100}
```

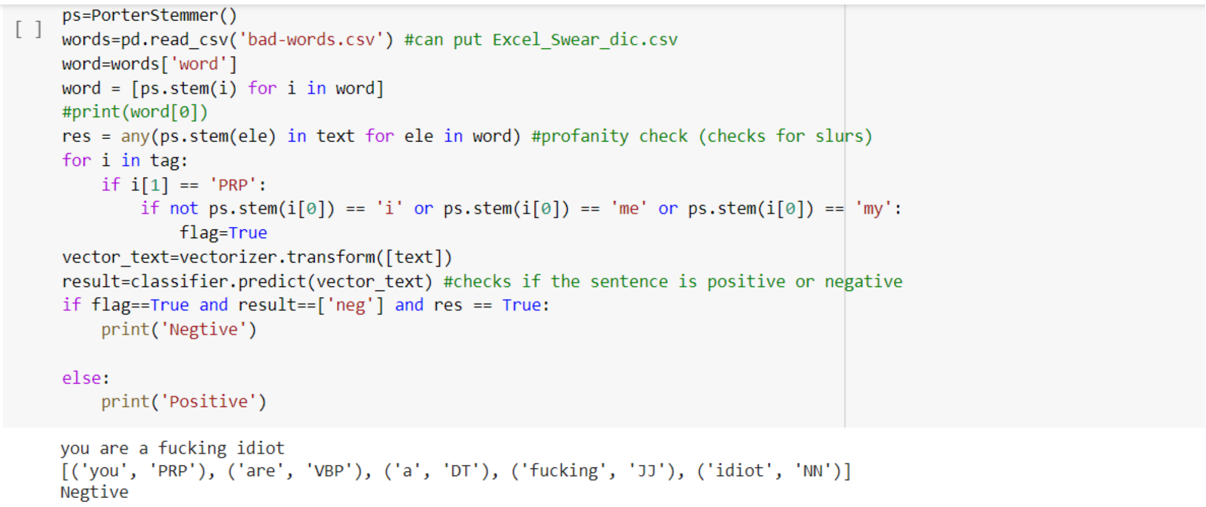
```python
!pip install nltk
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```



```
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk) (1.15.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
import pickle
import os
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.stem import PorterStemmer
import pandas as pd
```

```
vectorizer = pickle.load(open('vectorizer.sav', 'rb'))
classifier = pickle.load(open('classifier.sav', 'rb'))
text=input()
tokens = word_tokenize(text)
tag = pos_tag(tokens)
flag=False #if pronoun other than I, me, mine etc (first person pronouns)
print(tag)
ps=PorterStemmer()
words=pd.read_csv('bad-words.csv') #can put Excel_Swear_dic.csv
word=words['word']
word = [ps.stem(i) for i in word]
#print(word[0])
res = any(ps.stem(ele) in text for ele in word) #profanity check (checks for slurs)
```



```
ps=PorterStemmer()
words=pd.read_csv('bad-words.csv') #can put Excel_Swear_dic.csv
word=words['word']
word = [ps.stem(i) for i in word]
#print(word[0])
res = any(ps.stem(ele) in text for ele in word) #profanity check (checks for slurs)
for i in tag:
    if i[1] == 'PRP':
        if not ps.stem(i[0]) == 'i' or ps.stem(i[0]) == 'me' or ps.stem(i[0]) == 'my':
            flag=True
vector_text=vectorizer.transform([text])
result=classifier.predict(vector_text) #checks if the sentence is positive or negative
if flag==True and result==['neg'] and res == True:
    print('Negtive')

else:
    print('Positive')
```

```
you are a fucking idiot
[('you', 'PRP'), ('are', 'VBP'), ('a', 'DT'), ('fucking', 'JJ'), ('idiot', 'NN')]
Negtive
```

SAMPLE INPUT/OUTPUT :

INPUT - life is bad
OUTPUT - negative

RESULT :

Thus we have successfully implemented sentiment analysis and displayed the concept of NLP