PRACTICAL NO: 10

AIM: Perform Sentiment Analysis for Twitter

CODE and OUTPUT

```
import pandas as pd
import nltk
import string
from nltk.tokenize import TweetTokenizer

nltk.download('stopwords')
from nltk.corpus import stopwords

nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import time
from sklearn import svm
from sklearn.metrics import classification_report
import csv
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
tweetFile = pd.read_csv("/content/Tweets-Data.csv")
dataFrame = pd.DataFrame(tweetFile[['tweet_data']])
tweetData = tweetFile['tweet_data']

tknzr = TweetTokenizer()
stopWords = set(stopwords.words("english"))

# words = word_tokenize(data[0]) #For 1 line

cleanedData = []
cleaned = []
```

```
print("cleanedData:",cleanedData[:5])
```

```
cleanedData: []
```

```
for line in tweetData:
    tweet = tknzr.tokenize(str(line))

    for word in tweet:
        if word not in string.punctuation:
```

```
            if '@' not in word:
                cleaned.append(word)

    cleanedData.append(cleaned)
    cleaned = []

sentencedData = []
```

```
print("sentencedData:",sentencedData[:5])
```

```
sentencedData: []
```

```
for sentence in cleanedData:
    sentencedData.append(" ".join(sentence))

tweetFile.insert(4, "clean_data", "")

cleanData = tweetFile['clean_data']
i = 0
```

```
for row in sentencedData:
    cleanData[i] = sentencedData[i]
    i = i + 1

loopData = [0, 1, 2, 3, 4]
time_linear_train = []
time_linear_predict = []
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
for loop in loopData:
    t0 = 0
    t1 = 0
    t2 = 0

    tweetDataCopy = tweetFile.copy()

    trainedTweetData = tweetDataCopy.sample(frac=.8, random_state=0)
    testTweetData = tweetDataCopy.drop(trainedTweetData.index)

    sid = SentimentIntensityAnalyzer()
    i = 0
    sentimentData = []

    for sentence in trainedTweetData['clean_data']:
        sentimentData.append(sid.polarity_scores(sentence)['compound'])
```

```
sentimentLabel = []

for sentiment in sentimentData:
    if sentiment >= 0.05:
        sentimentLabel.append("pos")
    elif sentiment <= -0.05:
        sentimentLabel.append("neg")
    else:
        sentimentLabel.append("neu")

i = 0
sentimentTestData = []

for sentence in testTweetData['clean_data']:
    sentimentTestData.append(sid.polarity_scores(sentence)['compound'])

sentimentForTestLabel = []

for sentiment in sentimentTestData:
    if sentiment >= 0.05:
        sentimentForTestLabel.append("pos")
    elif sentiment <= -0.05:
        sentimentForTestLabel.append("neg")
    else:
        sentimentForTestLabel.append("neu")

data = {'clean_data': testTweetData.clean_data, 'sentiment': sentimentForTestLabel}
df = pd.DataFrame(data)
df.to_csv('test-data.csv')

data = {'clean_data': trainedTweetData.clean_data, 'sentiment': sentimentLabel}
df = pd.DataFrame(data)
df.to_csv('train-data.csv')

testData = pd.read_csv('test-data.csv')
trainData = pd.read_csv('train-data.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['clean_data'].values.astype('U'))
test_vectors = vectorizer.transform(testData['clean_data'].values.astype('U'))

# Perform classification with SVM, kernel=linear
classifier_linear = svm.SVC(kernel='linear')

t0 = time.time()

classifier_linear.fit(train_vectors, trainData['sentiment'])

t1 = time.time()

prediction_linear = classifier_linear.predict(test_vectors)

t2 = time.time()
```

```
    time_linear_train.append(t1 - t0)
    time_linear_predict.append(t2 - t1)

    # results
    print("Training time: %fs; Prediction time: %fs" % (time_linear_train[loop], time_line
    report = classification_report(testData['sentiment'], prediction_linear, output_dict=T

    print('positive: ', report['pos'])
    print('negative: ', report['neg'])

totalTrainTime = 0
totalPredictTime = 0
```

```
Training time: 0.646518s; Prediction time: 0.111754s
positive:  {'precision': 0.6641221374045801, 'recall': 0.5370370370370371, 'f1-score
negative:  {'precision': 0.6294964028776978, 'recall': 0.7543103448275862, 'f1-score
Training time: 0.633165s; Prediction time: 0.110381s
positive:  {'precision': 0.6641221374045801, 'recall': 0.5370370370370371, 'f1-score
negative:  {'precision': 0.6294964028776978, 'recall': 0.7543103448275862, 'f1-score
Training time: 0.638685s; Prediction time: 0.106753s
positive:  {'precision': 0.6641221374045801, 'recall': 0.5370370370370371, 'f1-score
negative:  {'precision': 0.6294964028776978, 'recall': 0.7543103448275862, 'f1-score
Training time: 0.631853s; Prediction time: 0.106404s
positive:  {'precision': 0.6641221374045801, 'recall': 0.5370370370370371, 'f1-score
negative:  {'precision': 0.6294964028776978, 'recall': 0.7543103448275862, 'f1-score
Training time: 0.633849s; Prediction time: 0.106184s
positive:  {'precision': 0.6641221374045801, 'recall': 0.5370370370370371, 'f1-score
negative:  {'precision': 0.6294964028776978, 'recall': 0.7543103448275862, 'f1-score
```

```
for i in loopData:
    totalTrainTime = totalTrainTime + time_linear_train[i]
    totalPredictTime = totalPredictTime + time_linear_predict[i]

print("Average training time: %fs" % (totalTrainTime / 5))
print("Average prediction time: %fs" % (totalPredictTime / 5))
```

```
Average training time: 0.636814s
Average prediction time: 0.108295s
```

Colab paid products - Cancel contracts here