BIG DATA PROJECT REPORT
UE19CS322

Name	AMAR S SAJJANSHETTY
Class and Sec	5th A sec
Team ID	BD2_032_150_189
Project Type	Machine Learning With Spark Streaming

Title: Twitter Sentiment Analysis using Pyspark.

Aim:

To implement the model using pyspark which analyze the twitter sentiment in real time.

Dataset Specification:

The data set contains two columns

1) Sentiment rating 0 and 4 (values)
   if 0 then the user is negative
   if 4 then the user is positive
2) Tweets :

This column contains the unprocessed tweets sent by the user. This tweet has important data like hashtags, (recent topics)
etc which is considered as noun.

Language and Libraries used:

1) Pyspark
2) python

important libraries are:

1) Sklearn : To build machine learning models (classifiers)
2) Pandas: To create dataframe for the incomming batch of Dstream
3) Numpy: To apply functions and other necessary computation
for the data. To convert the data format to numpy array and it is given as input to the classifier.
4) nltk: This library is used to preprocess the tweets of the user
   it only extracts the meaningful and precise words to train the model.
5) matplotlib : To plot the graph and analyse which model is performing best with respect to parameters given
6) pickle: This library is used to store the parameters of the model for latter use
7) json: this library is used to convert the text stream to json stream
8) sys: This library is used to declare command line args for execution of the code.
9) re : This is regular expression library for filtering
etc…..

Steps:

To extract and display the streams coming from the stream.py periodically and convert it into dataframe.

Preprocess the text using nltk library and regular expression.

1) convert all the words in lower format

2)Removing # hashtags and emoticons like [:-) :-} :-(:-> :-D] etc..

3)Removing punctuations and other stopping words.

```
  feature0                                                 feature1
0        4                              ohh could life get better?
1        0    say gianti oolll ddddooonnngggg multiexclam url
2        4  atus someon help makeup multistopy got go look...
3        0                atus tell friend fan least multistop
4        0  miss highschool day multistopjust saw c.a.t vi...
5        0  bon dimanche, bouff apr fini le nettoyag youpp...
6        0  atus me, watch traumat video sa mga happen sa ...
7        4         atus url harri awar microphon point mouth?
8        0                             atus sold outttt! depress
9        0           atus multistop+ event, extrem disappoint
  feature0                                                 feature1
0        4  atus white water? and, i'v two glass wine, thi...
1        0              sorbitol cough syrup caus loos bowels.
2        0                 can't sleep late anymor multiexclam
3        4  nighti night twechelon sleep well everybodi sh...
4        4  lvatt everyon got buy cd get itun multiexclam ...
5        0         mental tick book need buy. quit long list.
6        4  atus relax multiexclam yur mean bou tat hole b...
7        4                            play ts. phone multiexclam
8        4                                atus mean i'm invited?
9        4             atus think need talk need ta teh camera
```

4) tokenizing the words (converting string to list of words)

```
21/12/08 09:50:28 INFO BlockManager: Initialized BlockManager:
  feature0                                                 feature1
0        4                      [ohh, could, life, get, better?]
1        0  [say, gianti, oolll, ddddooonnngggg, multiexcl...
2        4  [atus, someon, help, makeup, multistopy, got, ...
3        0        [atus, tell, friend, fan, least, multistop]
4        0  [miss, highschool, day, multistopjust, saw, c....
5        0  [bon, dimanche,, bouff, apr, fini, le, nettoya...
6        0  [atus, me,, watch, traumat, video, sa, mga, ha...
7        4  [atus, url, harri, awar, microphon, point, mou...
8        0                      [atus, sold, outttt!, depress]
9        0     [atus, multistop+, event,, extrem, disappoint]
```

5) Using TfidfVectorizer the tokenized list of words is converted to numeric data (as the model works on numeric data)

It maps the most frequent words occurrence frequency in sparse matrix.

Implement of machine learning models:
Models implemented are:

1) Support Vector machine classifier:

```
def svm_classifier(X_train, X_test, y_train, y_test):
        SVCmodel = svm.LinearSVC()
        SVCmodel.fit(X_train, y_train)
        y_pred2 = SVCmodel.predict(X_test)
        file1 = 'final1_model.sav'
        pickle.dump(SVCmodel, open(file1, 'wb'))
        return accuracy_score(y_test,y_pred2)
```

2) Random Forest classifier:

```
def Rf_classifier(X_train, X_test, y_train, y_test):
        clf=BernoulliNB()
        clf.fit(X_train,y_train)
        y_pred2=clf.predict(X_test)
        file3 = 'final3_model.sav'
        pickle.dump(clf, open(file3, 'wb'))
        return accuracy_score(y_test,y_pred2)
```

3) Bernoulli Naive Bias classifier:

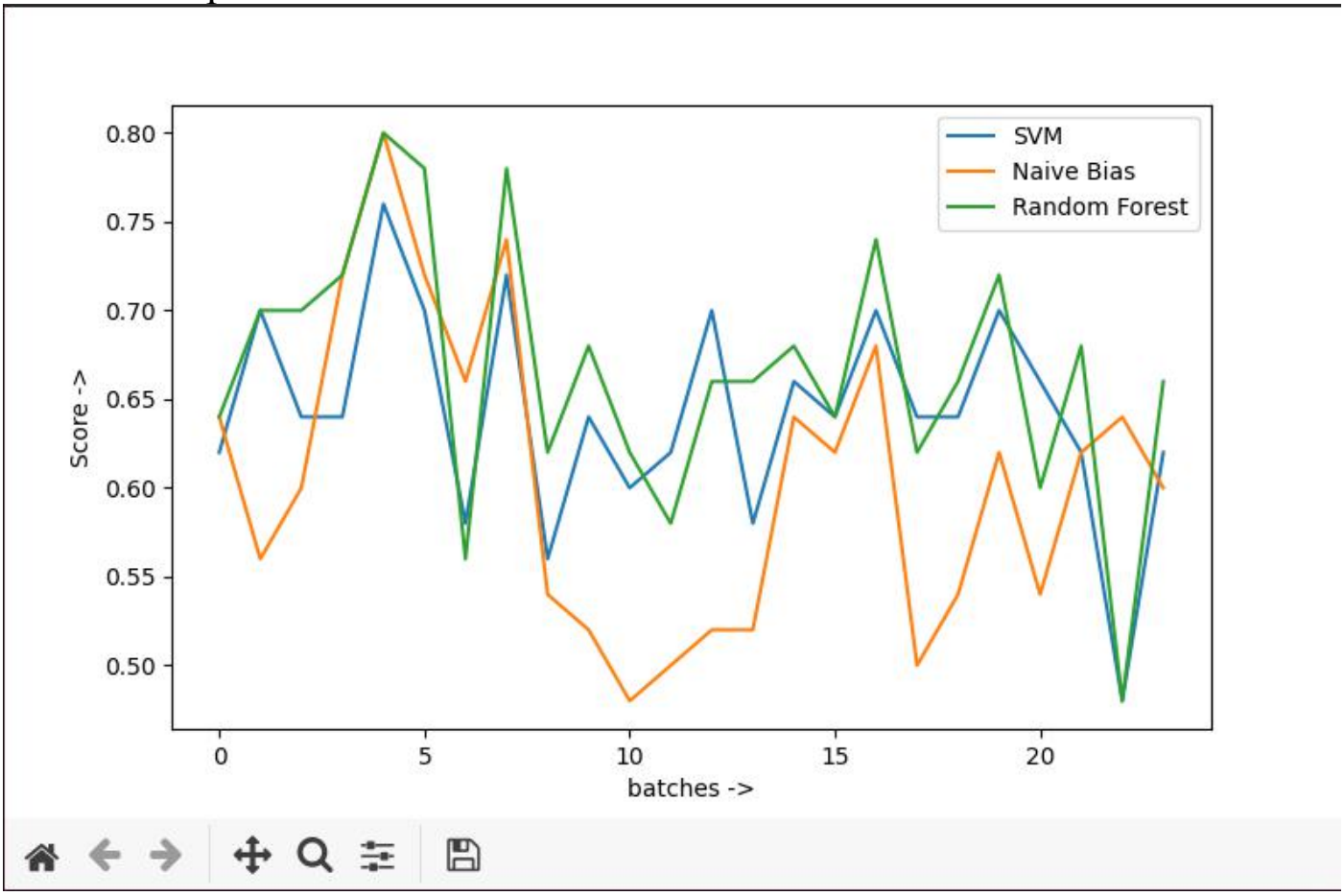
```
def NB_classifier(X_train, X_test, y_train, y_test):
        clf = RandomForestClassifier(max_depth=2, random_state=0)
        clf.fit(X_train,y_train)
        y_pred2= clf.predict(X_test)
        file2 = 'final2_model.sav'
        pickle.dump(clf, open(file2, 'wb'))
        return accuracy_score(y_test,y_pred2)
```

```
Score c1: 0.62  ,Score c2: 0.64  ,Score: 0.64
Score c1: 0.7  ,Score c2: 0.56  ,Score: 0.7
Score c1: 0.64  ,Score c2: 0.6  ,Score: 0.7
Score c1: 0.64  ,Score c2: 0.72  ,Score: 0.72
Score c1: 0.76  ,Score c2: 0.8  ,Score: 0.8
Score c1: 0.7  ,Score c2: 0.72  ,Score: 0.78
Score c1: 0.58  ,Score c2: 0.66  ,Score: 0.56
Score c1: 0.72  ,Score c2: 0.74  ,Score: 0.78
Score c1: 0.56  ,Score c2: 0.54  ,Score: 0.62
Score c1: 0.64  ,Score c2: 0.52  ,Score: 0.68
Score c1: 0.6  ,Score c2: 0.48  ,Score: 0.62
Score c1: 0.62  ,Score c2: 0.5  ,Score: 0.58
Score c1: 0.7  ,Score c2: 0.52  ,Score: 0.66
Score c1: 0.58  ,Score c2: 0.52  ,Score: 0.66
Score c1: 0.66  ,Score c2: 0.64  ,Score: 0.68
Score c1: 0.64  ,Score c2: 0.62  ,Score: 0.64
Score c1: 0.7  ,Score c2: 0.68  ,Score: 0.74
Score c1: 0.64  ,Score c2: 0.5  ,Score: 0.62
Score c1: 0.64  ,Score c2: 0.54  ,Score: 0.66
Score c1: 0.7  ,Score c2: 0.62  ,Score: 0.72
Score c1: 0.66  ,Score c2: 0.54  ,Score: 0.6
```

Here the accuracy scores are stored in track_model.csv file for further computations for each batch processed.

	A	B	C	D
1	SVM_class	NB_Class	RF_class	
2	0.62	0.64	0.64	
3	0.7	0.56	0.7	
4	0.64	0.6	0.7	
5	0.64	0.72	0.72	
6	0.76	0.8	0.8	
7	0.7	0.72	0.78	
8	0.58	0.66	0.56	
9	0.72	0.74	0.78	
10	0.56	0.54	0.62	
11	0.64	0.52	0.68	
12	0.6	0.48	0.62	
13	0.62	0.5	0.58	
14	0.7	0.52	0.66	
15	0.58	0.52	0.66	
16	0.66	0.64	0.68	
17	0.64	0.62	0.64	
18	0.7	0.68	0.74	
19	0.64	0.5	0.62	
20	0.64	0.54	0.66	
21	0.7	0.62	0.72	
22	0.66	0.54	0.6	

Plots to compare different classifiers:



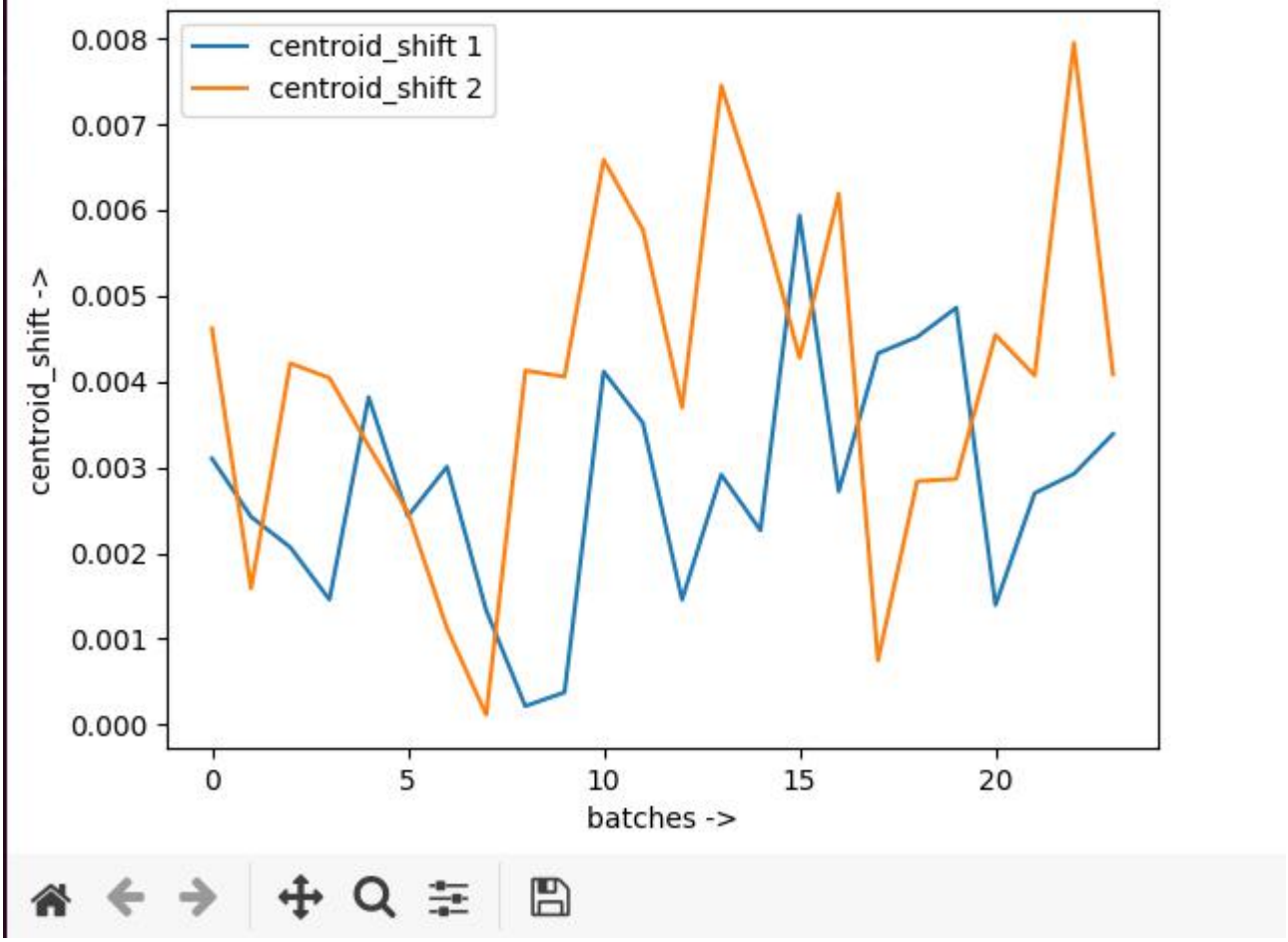
Metrics predicted from test data is here as follows:

```
### report of SVM ###
              precision    recall  f1-score   support

           0       0.79      0.81      0.80       504
           4       0.80      0.78      0.79       496

    accuracy                           0.79      1000
   macro avg       0.79      0.79      0.79      1000
weighted avg       0.79      0.79      0.79      1000


### report of NB_classifier ###
              precision    recall  f1-score   support

           0       0.65      0.79      0.71       504
           4       0.72      0.56      0.63       496

    accuracy                           0.68      1000
   macro avg       0.68      0.68      0.67      1000
weighted avg       0.68      0.68      0.67      1000


### report of RF_classifier ###
              precision    recall  f1-score   support

           0       0.78      0.75      0.76       504
           4       0.75      0.79      0.77       496

    accuracy                           0.77      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.77      0.77      0.77      1000
```

Clustering algorithm:

```
[0.0031033394115905495, 0.0046198653611396 28]
[0.0024248529234458887, 0.0015900011681361857]
[0.0020688964369162508, 0.004212739869395 82]
[0.0014591542824101866, 0.0040425156195552 08]
[0.0038193989274485146, 0.0032509436132060502]
[0.0024303288059974344, 0.0024783730219919108]
[0.003007755959160241, 0.0011264473064729382]
[0.0013347970742040535, 0.00011620201897319556]
############  small value of centroid shift detected ########################
[0.00021744618830572197, 0.0041286015822745836]
############  small value of centroid shift detected ########################
[0.0003778894487325582, 0.004057805931778525]
############  small value of centroid shift detected ########################
[0.004117711677220063, 0.006585207111953961]
[0.0035173463714888845, 0.005769687432170629]
[0.0014584852207433952, 0.0036977200964222762]
[0.002915025595789188, 0.007452728657995193]
[0.002266052142763173, 0.005994253596968403]
[0.00593558868218071, 0.004275434634870899]
[0.002722082704203702, 0.006191288735651952]
```

plots.


centroid_shift 1
centroid_shift 2
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
centroid_shift ->
0
5
10
15
20
batches ->