Team Id : 14

Member 1 : Ruthvic Punyamurtula

Class Id : 16

Member 2 : Shankar Pentyala

Class Id : 15

Source Code :

https://github.com/Ruthvicp/CS5590_BigDataProgramming/tree/mas

Video/Demo : **https://youtu.be/42IJBhnslpk**

Introduction

This lab assignment consists of using Spark MLlib classification alogirthms on the given data set and also run the word count on the twitter streaming data.

Objective

1. Classification Algorithms used are - Decision Tree, Naive Bayes, Random Forest

Approach

Read the data set and convert the column data from string to float/double

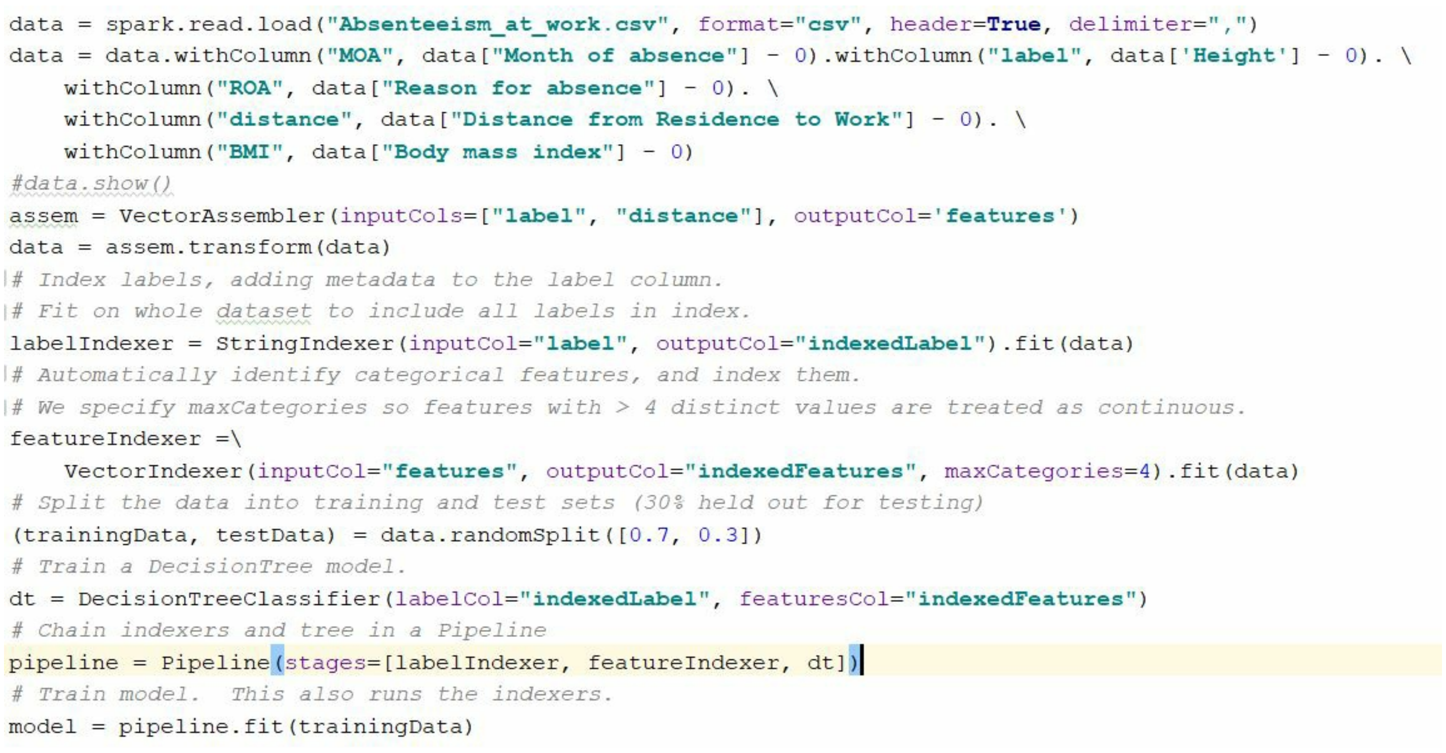
type. We perform the classification based on
the columns “Month of Absence”, “Day of the week”, “Height”, “Travel expenses”, “Distance” ,“Body Mass Index”.

Workflow

We split the data into train-70%, Test-30%. Finally we evaluate the model and predict the results of the test data set.
Then calculate the confusion matrix using the above columns and find the precision and recall values.

1. Decision Tree

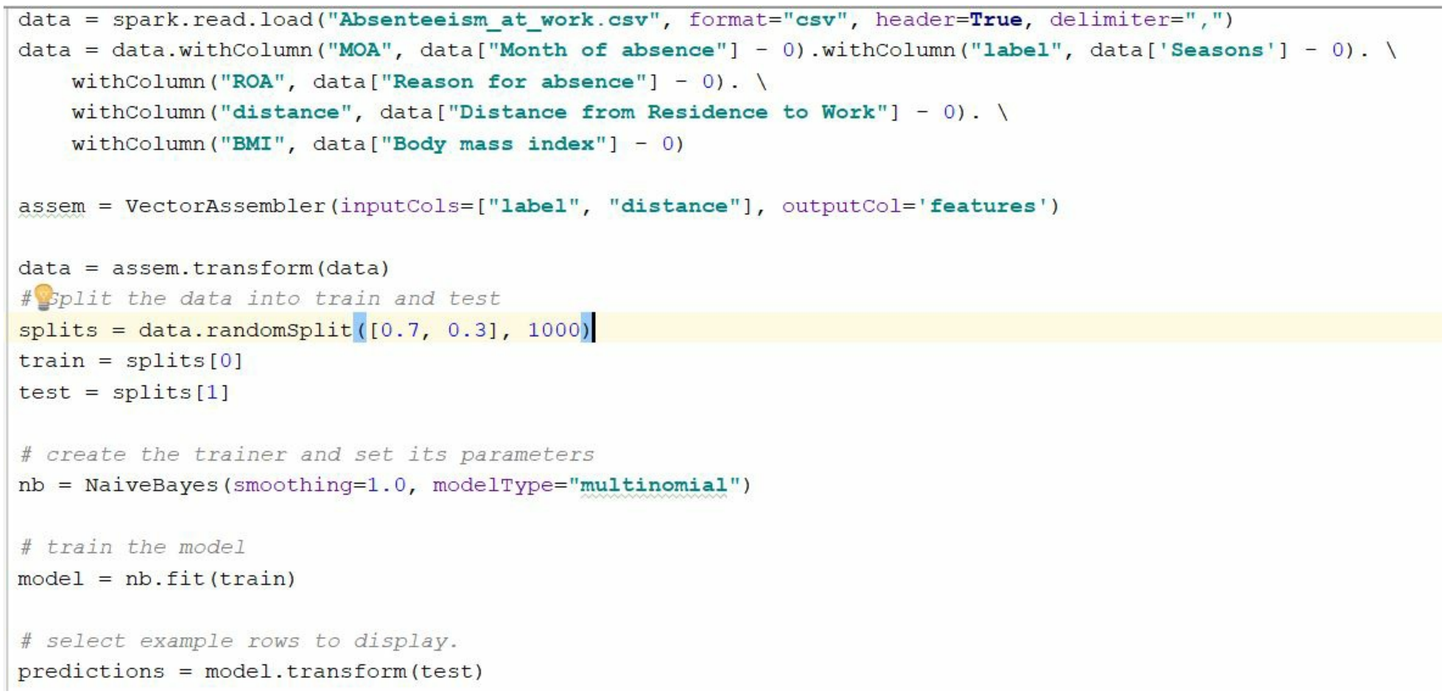
We create a vector assembler on input data columns “label (Height)” and “Distance” and use the DecisionTreeClassifier on indexedlabel and indexed features



```
data = spark.read.load("Absenteeism_at_work.csv", format="csv", header=True, delimiter=",")
data = data.withColumn("MOA", data["Month of absence"] - 0).withColumn("label", data['Height'] - 0). \
    withColumn("ROA", data["Reason for absence"] - 0). \
    withColumn("distance", data["Distance from Residence to Work"] - 0). \
    withColumn("BMI", data["Body mass index"] - 0)
#data.show()
assem = VectorAssembler(inputCols=["label", "distance"], outputCol='features')
data = assem.transform(data)
# Index labels, adding metadata to the label column.
# Fit on whole dataset to include all labels in index.
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(data)
# Automatically identify categorical features, and index them.
# We specify maxCategories so features with > 4 distinct values are treated as continuous.
featureIndexer =\
    VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(data)
# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = data.randomSplit([0.7, 0.3])
# Train a DecisionTree model.
dt = DecisionTreeClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures")
# Chain indexers and tree in a Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, dt])
# Train model.  This also runs the indexers.
model = pipeline.fit(trainingData)
```

2. Naive Bayes

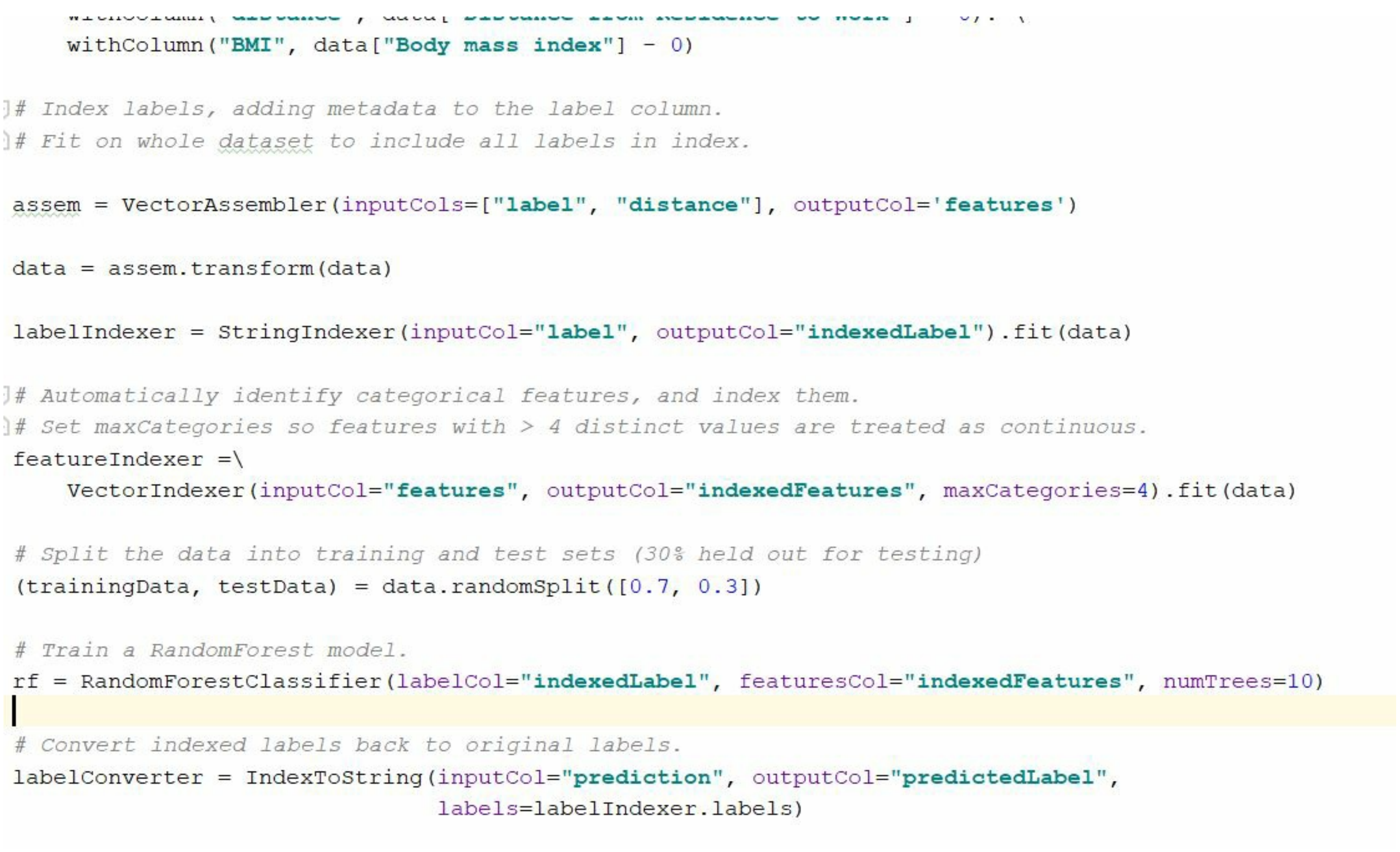
For the same data set and columns we perform naive bayes to find the prediction for absenteeism at work



```
data = spark.read.load("Absenteeism_at_work.csv", format="csv", header=True, delimiter=",")
data = data.withColumn("MOA", data["Month of absence"] - 0).withColumn("label", data['Seasons'] - 0). \
    withColumn("ROA", data["Reason for absence"] - 0). \
    withColumn("distance", data["Distance from Residence to Work"] - 0). \
    withColumn("BMI", data["Body mass index"] - 0)

assem = VectorAssembler(inputCols=["label", "distance"], outputCol='features')

data = assem.transform(data)
#Split the data into train and test
splits = data.randomSplit([0.7, 0.3], 1000)
train = splits[0]
test = splits[1]

# create the trainer and set its parameters
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")

# train the model
model = nb.fit(train)

# select example rows to display.
predictions = model.transform(test)
```

3. Random Forest

We use Random forest Classifier for the input columns height and distance and create a vector assembler on this indexed data.



```
    withColumn("distance", data["Distance from Residence to Work"] - 0). \
    withColumn("BMI", data["Body mass index"] - 0)

# Index labels, adding metadata to the label column.
# Fit on whole dataset to include all labels in index.

assem = VectorAssembler(inputCols=["label", "distance"], outputCol='features')

data = assem.transform(data)

labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(data)

# Automatically identify categorical features, and index them.
# Set maxCategories so features with > 4 distinct values are treated as continuous.
featureIndexer =\
    VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(data)

# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Train a RandomForest model.
rf = RandomForestClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures", numTrees=10)

# Convert indexed labels back to original labels.
labelConverter = IndexToString(inputCol="prediction", outputCol="predictedLabel",
                               labels=labelIndexer.labels)
```

Data set and Parameter

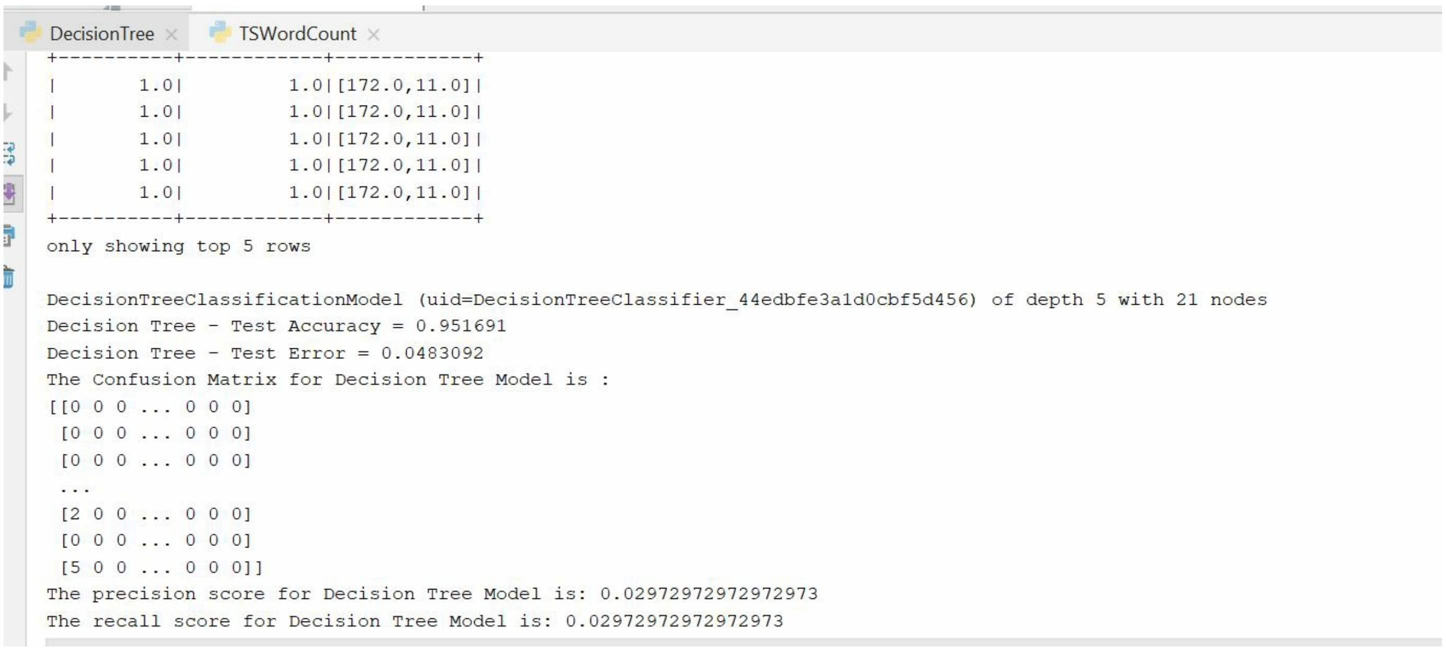
The input file for this can be found at https://github.com/Ruthvicp/CS5590_BigDataProgramming/raw/master/Lab and also at https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work

Evaluation

we compare the results - Accuracy and Error calculated using the above classifier and pick the best possible based on highest accuracy or lowest error

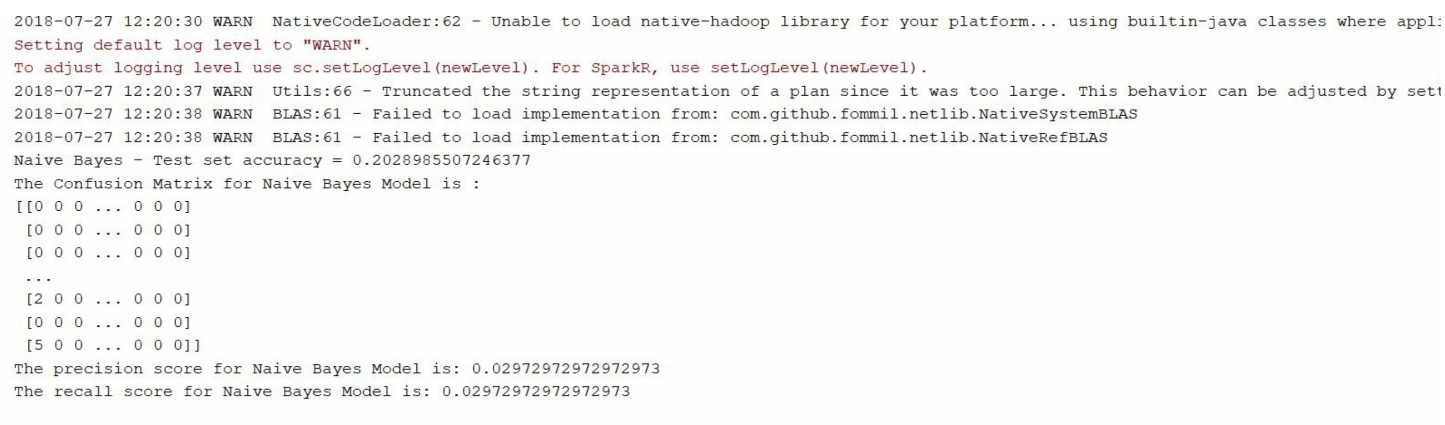
Output

The snapshot for the output for **Decision Tree** is given below



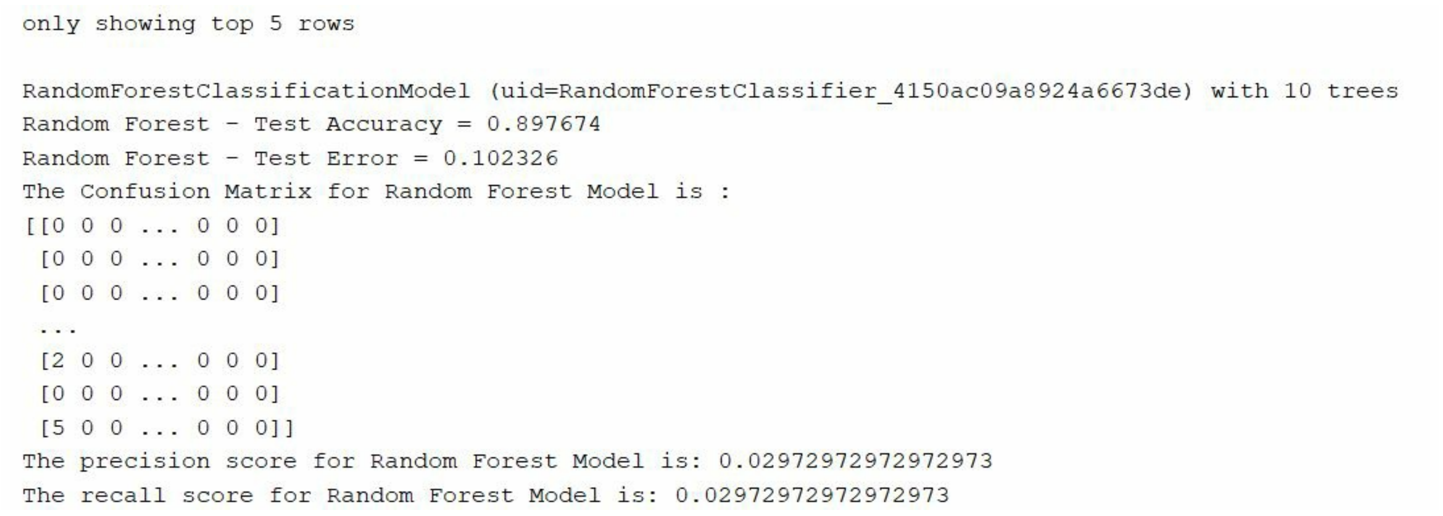
```
+----------+------------+------------+
|       1.0|         1.0|[172.0,11.0]|
|       1.0|         1.0|[172.0,11.0]|
|       1.0|         1.0|[172.0,11.0]|
|       1.0|         1.0|[172.0,11.0]|
|       1.0|         1.0|[172.0,11.0]|
+----------+------------+------------+
only showing top 5 rows

DecisionTreeClassificationModel (uid=DecisionTreeClassifier_44edbfe3a1d0cbf5d456) of depth 5 with 21 nodes
Decision Tree - Test Accuracy = 0.951691
Decision Tree - Test Error = 0.0483092
The Confusion Matrix for Decision Tree Model is :
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [2 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [5 0 0 ... 0 0 0]]
The precision score for Decision Tree Model is: 0.02972972972972973
The recall score for Decision Tree Model is: 0.02972972972972973
```

The snapshot for the output for **Naive Bayes** is given below



```
2018-07-27 12:20:30 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where appl
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2018-07-27 12:20:37 WARN  Utils:66 - Truncated the string representation of a plan since it was too large. This behavior can be adjusted by sett
2018-07-27 12:20:38 WARN  BLAS:61 - Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
2018-07-27 12:20:38 WARN  BLAS:61 - Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Naive Bayes - Test set accuracy = 0.2028985507246377
The Confusion Matrix for Naive Bayes Model is :
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [2 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [5 0 0 ... 0 0 0]]
The precision score for Naive Bayes Model is: 0.02972972972972973
The recall score for Naive Bayes Model is: 0.02972972972972973
```

The snapshot for the output for **Random Forest** is given below



```
only showing top 5 rows

RandomForestClassificationModel (uid=RandomForestClassifier_4150ac09a8924a6673de) with 10 trees
Random Forest - Test Accuracy = 0.897674
Random Forest - Test Error = 0.102326
The Confusion Matrix for Random Forest Model is :
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [2 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [5 0 0 ... 0 0 0]]
The precision score for Random Forest Model is: 0.02972972972972973
The recall score for Random Forest Model is: 0.02972972972972973
```

conclusion

Based on the accuracy, for the columns chosen, Decision Tree has the highest accuracy of 95 %
and the lowest error of 5%. Also the confusion matrix is plotted for all the 3 classifications
and is shown in the above output images

---

2. Word count on Twitter streaming data

Introduction

We create a streaming context on a host and bind it to an available port and send the streaming context on it.
Once the receiver starts listening to the same port, then the data is sent across. On this data we perform the
word count to get the results.

Objectives

a) Create a twitter streaming class to connect to twitter using the auth credentials

b) Create a listening class to bind onto same host and port

c) Perform the word count on it

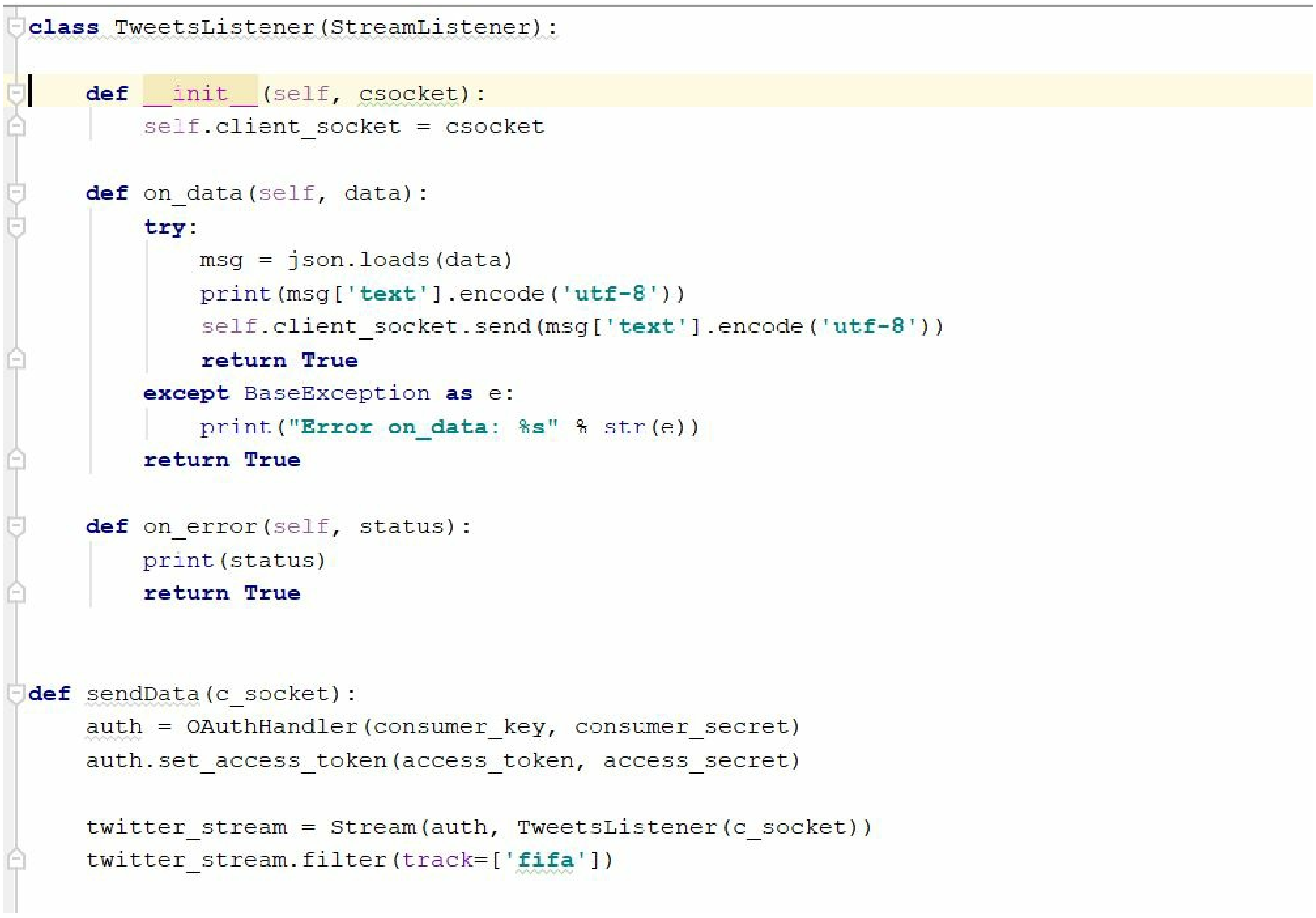
Approach

Binding a stream using " s.bind((host, port))". Establish the connection "c, addr = s.accept() ". Now send the data
as in “sendData©”.

Inside on_Data() : separate the twitter text and encode them before sending - self.client_socket.send(msg[‘text’].encode(‘utf-8’))

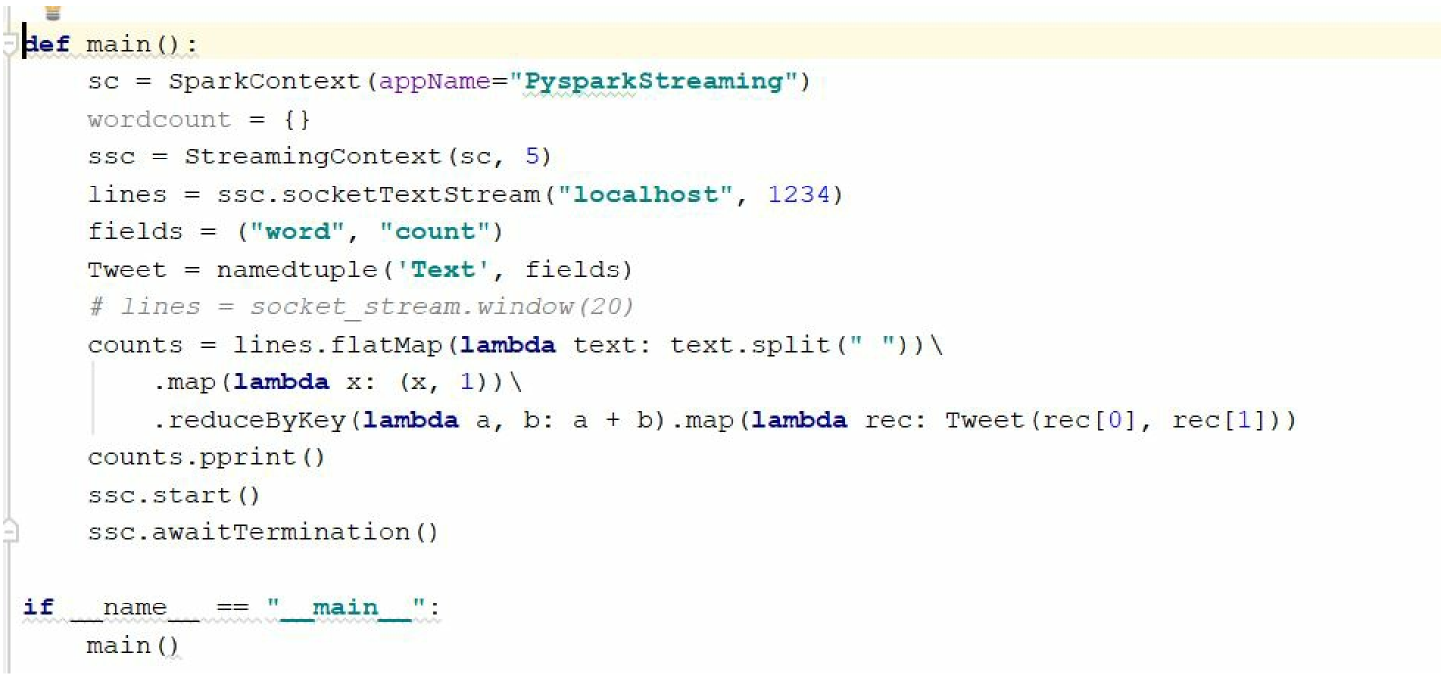
Workflow

Twitter Streaming class



```python
class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            msg = json.loads(data)
            print(msg['text'].encode('utf-8'))
            self.client_socket.send(msg['text'].encode('utf-8'))
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
        return True


def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    twitter_stream.filter(track=['fifa'])
```

Listening class and then the word count



```python
def main():
    sc = SparkContext(appName="PysparkStreaming")
    wordcount = {}
    ssc = StreamingContext(sc, 5)
    lines = ssc.socketTextStream("localhost", 1234)
    fields = ("word", "count")
    Tweet = namedtuple('Text', fields)
    # lines = socket_stream.window(20)
    counts = lines.flatMap(lambda text: text.split(" "))\
        .map(lambda x: (x, 1))\
        .reduceByKey(lambda a, b: a + b).map(lambda rec: Tweet(rec[0], rec[1]))
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()

if __name__ == "__main__":
    main()
```

Datasets & parameters

Set the authentication credentials given below

- consumer_key
- consumer_secret
- access_token
- access_secret

auth = OAuthHandler(consumer_key, consumer_secret)

auth.set_access_token(access_token, access_secret)

in send_data() :

type the below code to get twitter streams. I have filtered the tweets based on 'fifa'

auth = OAuthHandler(consumer_key, consumer_secret)

auth.set_access_token(access_token, access_secret)

```
twitter_stream = Stream(auth, TweetsListener(c_socket))
twitter_stream.filter(track=['fifa'])
```

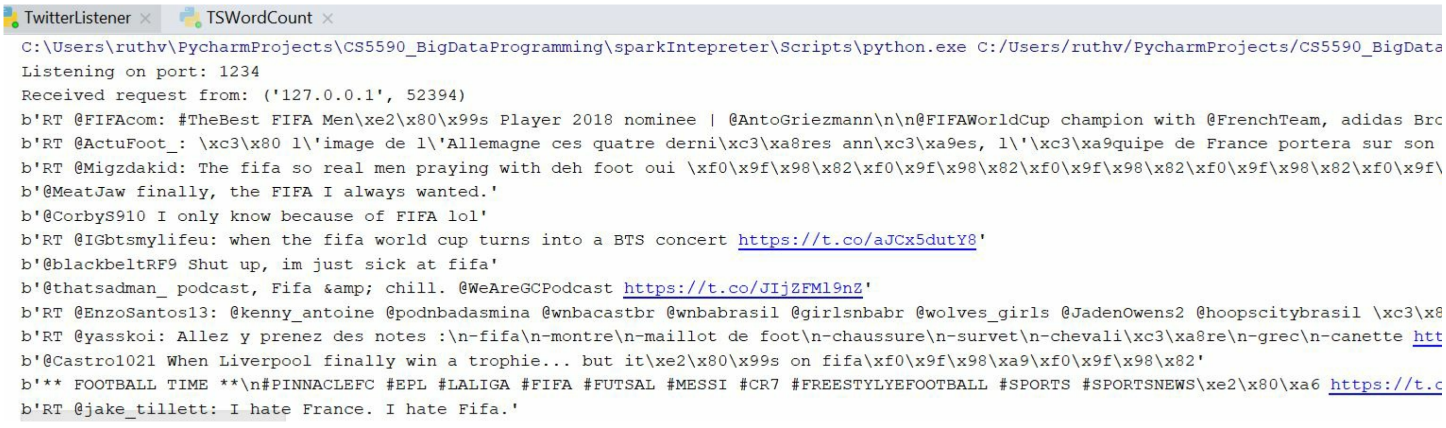
Evaluation

To get proper word count results, enocode on sending side and decode the data on the receiving end.

Set the window size and reduce the streaming context duration to have the word count done as fast as possible

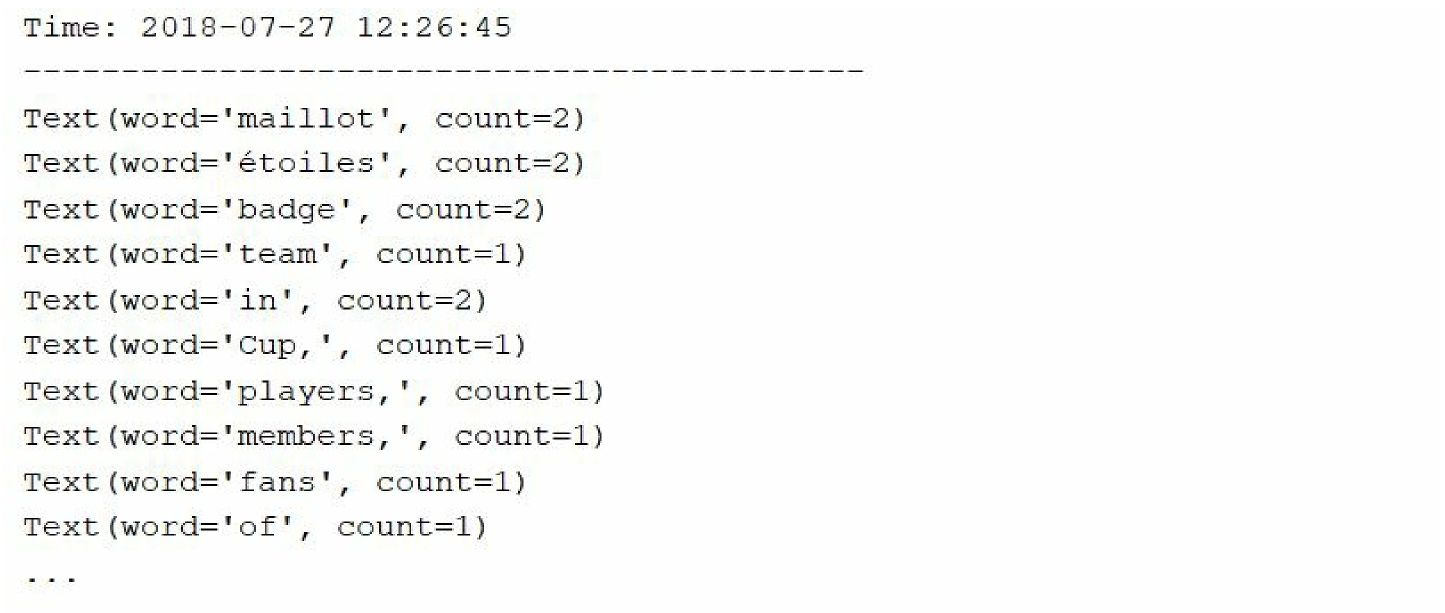
Output

screen shot on running the twitter streaming class :



```
C:\Users\ruthv\PycharmProjects\CS5590_BigDataProgramming\sparkIntepreter\Scripts\python.exe C:/Users/ruthv/PycharmProjects/CS5590_BigData
Listening on port: 1234
Received request from: ('127.0.0.1', 52394)
b'RT @FIFAcom: #TheBest FIFA Men\xe2\x80\x99s Player 2018 nominee | @AntoGriezmann\n\n@FIFAWorldCup champion with @FrenchTeam, adidas Bro
b'RT @ActuFoot_: \xc3\x80 l\'image de l\'Allemagne ces quatre derni\xc3\xa8res ann\xc3\xa9es, l\'\xc3\xa9quipe de France portera sur son
b'RT @Migzdakid: The fifa so real men praying with deh foot oui \xf0\x9f\x98\x82\xf0\x9f\x98\x82\xf0\x9f\x98\x82\xf0\x9f\x98\x82\xf0\x9f\
b'@MeatJaw finally, the FIFA I always wanted.'
b'@CorbyS910 I only know because of FIFA lol'
b'RT @IGbtsmylifeu: when the fifa world cup turns into a BTS concert https://t.co/aJCx5dutY8'
b'@blackbeltRF9 Shut up, im just sick at fifa'
b'@thatsadman_ podcast, Fifa &amp; chill. @WeAreGCPodcast https://t.co/JIjZFMl9nZ'
b'RT @EnzoSantos13: @kenny_antoine @podnbadasmina @wnbacastbr @wnbabrasil @girlsnbabr @wolves_girls @JadenOwens2 @hoopscitybrasil \xc3\x8
b'RT @yasskoi: Allez y prenez des notes :\n-fifa\n-montre\n-maillot de foot\n-chaussure\n-survet\n-chevali\xc3\xa8re\n-grec\n-canette htt
b'@Castro1021 When Liverpool finally win a trophie... but it\xe2\x80\x99s on fifa\xf0\x9f\x98\xa9\xf0\x9f\x98\x82'
b'** FOOTBALL TIME **\n#PINNACLEFC #EPL #LALIGA #FIFA #FUTSAL #MESSI #CR7 #FREESTYLYEFOOTBALL #SPORTS #SPORTSNEWS\xe2\x80\xa6 https://t.c
b'RT @jake_tillett: I hate France. I hate Fifa.'
```

screen shot of the word count output on the tweets is :



```
Time: 2018-07-27 12:26:45
-------------------------------------------
Text(word='maillot', count=2)
Text(word='étoiles', count=2)
Text(word='badge', count=2)
Text(word='team', count=1)
Text(word='in', count=2)
Text(word='Cup,', count=1)
Text(word='players,', count=1)
Text(word='members,', count=1)
Text(word='fans', count=1)
Text(word='of', count=1)
...
```

Conclusion

The word count for the fifa tweets is done using the twitter streams by creating a streaming context in spark.

We have done word count on 7237 tweets in 3 minutes of duration.

References

- https://spark.apache.org/docs/latest/ml-decision-tree.html
- https://spark.apache.org/docs/2.2.0/mllib-naive-bayes.html
- https://weiminwang.blog/2016/06/09/pyspark-tutorial-building-a-random-forest-binary-classifier-on-unbalanced-dataset/
- https://stackoverflow.com/questions/43872281/pyspark-find-number-of-tweets-that-contain-a-word-hashtag