Big Data Programming

CSEE5590/490

Module 2 Lab 2

Report

Submitted By:

Sneha Mishra:

Class ID: 11

Aditya Soman

Class ID: 19

Team: 12
Professor: Yugyung Lee

Name: Sneha Mishra
Class ID: 11
Email: smccr@mail.umkc.edu
MyGitHub

Technical Partner:
Name: Aditya Soman
Class ID: 19
Email: as9f3@mail.umkc.edu
GitHub

YouTube Link explaining the Lab work can be found here
The report for the Lab work is here
The source code for this lab work can be found here
The available datasets formats can be found here

Objective

Understanding Spark Classification, Spark Streaming and Spark Graphx Task.

Features

1. Use of Classification Algorithms such as Naïve Bayes, Decision Tree, Random Forest for attribute classification.
2. Report the Confusion matrix, Accuracy based on FMeasure, Precision & Recall for all the algorithms.
3. Reason why one of algorithms out performs the rest.
4. Perform Word-Count on Twitter Streaming Data using Spark.
5. Perform Page Rank on given Dataset.
6. State importance of using graphx on the chosen dataset.

Steps:

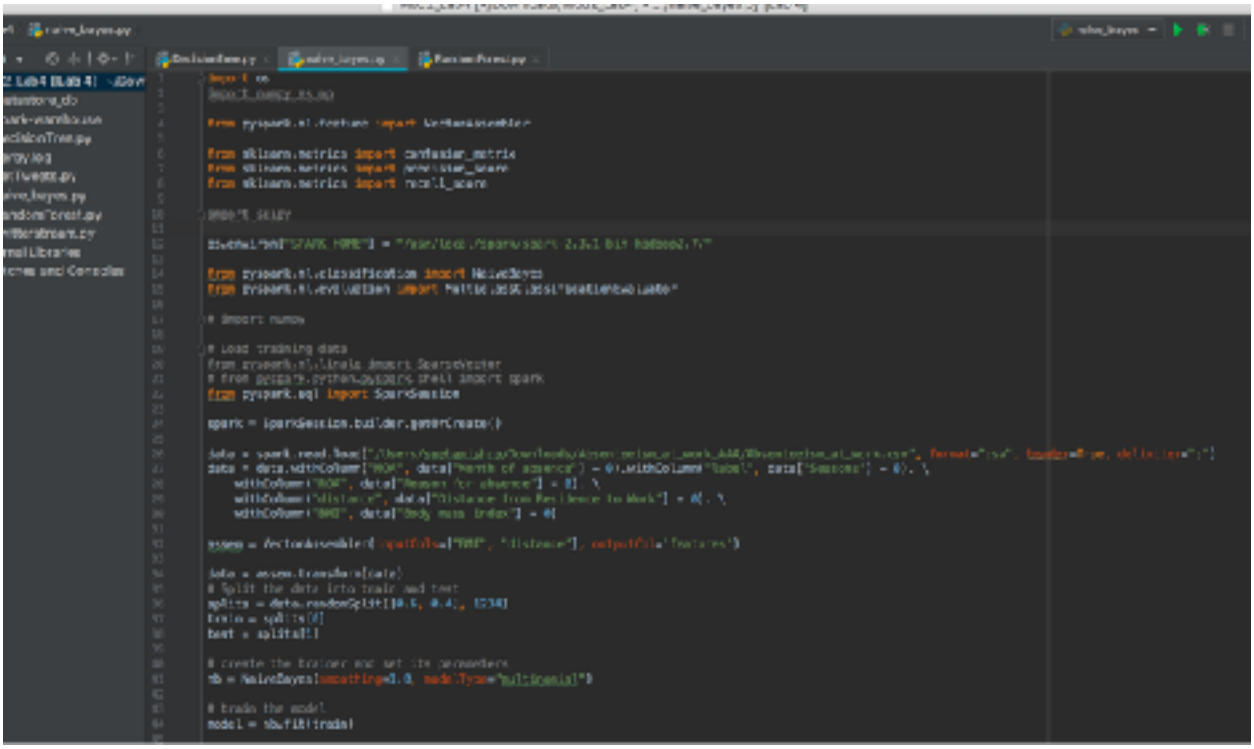
Part 1: Spark Classification Task

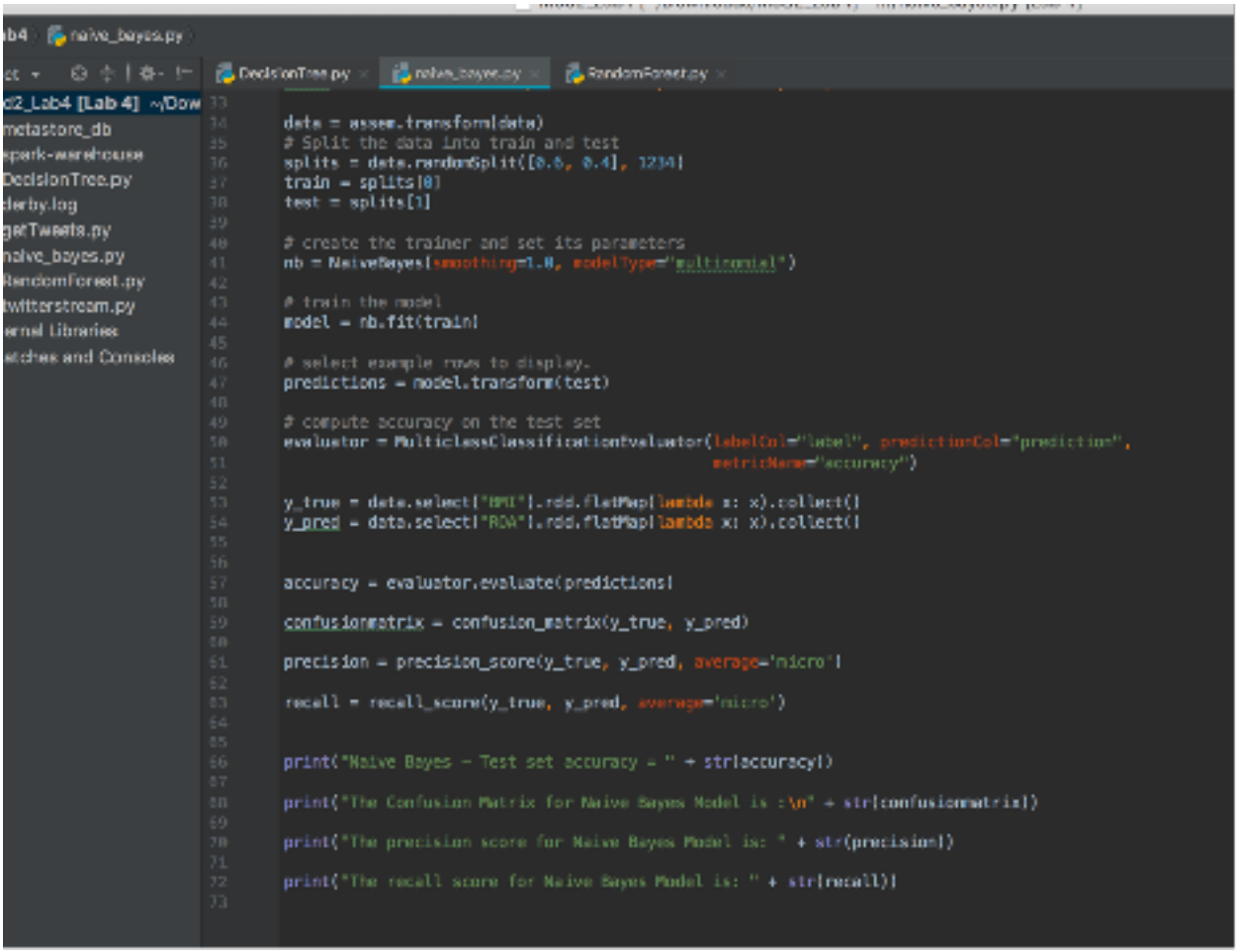
This task contains working on 3 algorithms namely:

1. Naïve Bayes:

It is a classification technique based on Bayes' theorem. Naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

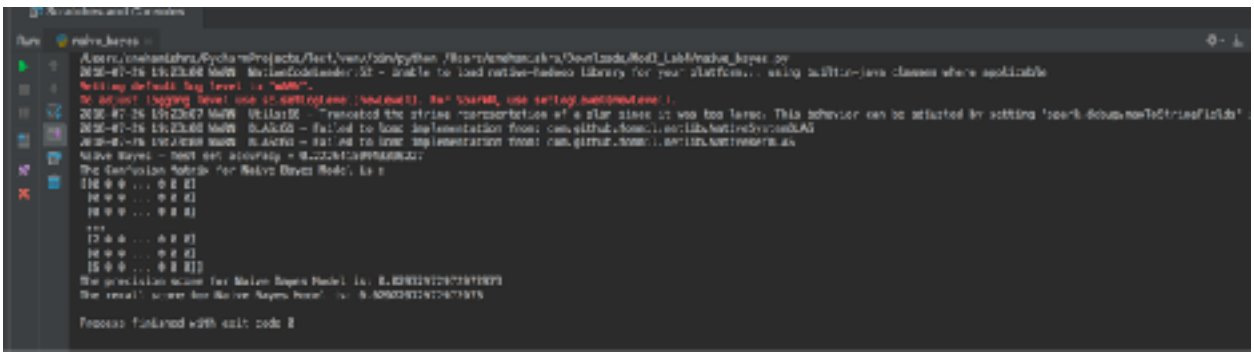
Code for the Algorithm:





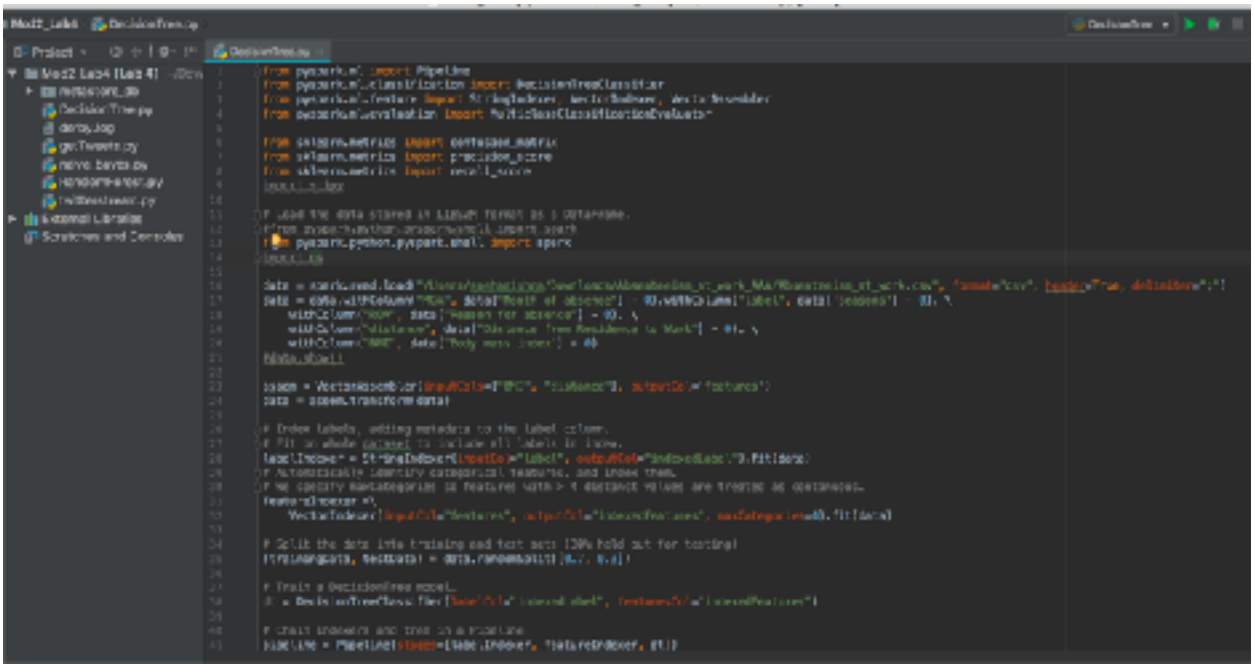
```
data = assem.transform(data)
# Split the data into train and test
splits = data.randomSplit([0.6, 0.4], 1234)
train = splits[0]
test = splits[1]

# 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)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                              metricName="accuracy")

y_true = data.select("MPI").rdd.flatMap(lambda x: x).collect()
y_pred = data.select("RDA").rdd.flatMap(lambda x: x).collect()


accuracy = evaluator.evaluate(predictions)

confusionmatrix = confusion_matrix(y_true, y_pred)

precision = precision_score(y_true, y_pred, average='micro')

recall = recall_score(y_true, y_pred, average='micro')


print("Naive Bayes - Test set accuracy = " + str(accuracy))

print("The Confusion Matrix for Naive Bayes Model is :\n" + str(confusionmatrix))

print("The precision score for Naive Bayes Model is: " + str(precision))

print("The recall score for Naive Bayes Model is: " + str(recall))
```

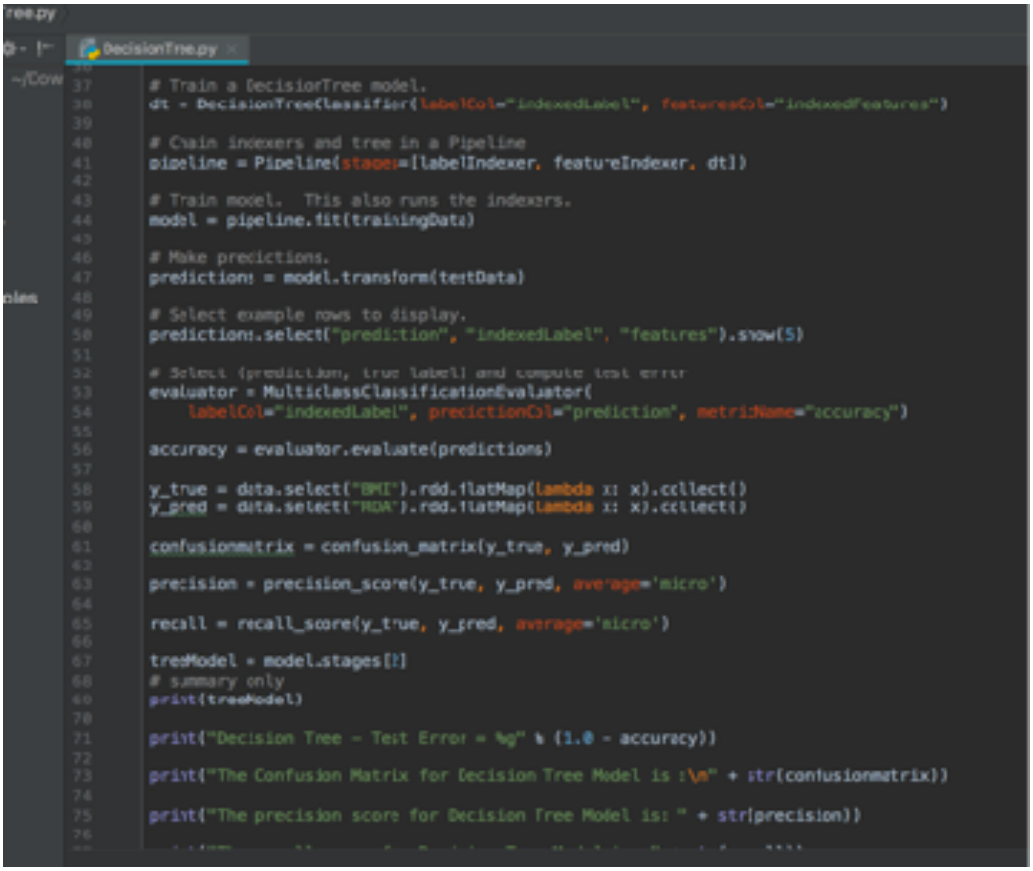
Output after running the Algorithm:



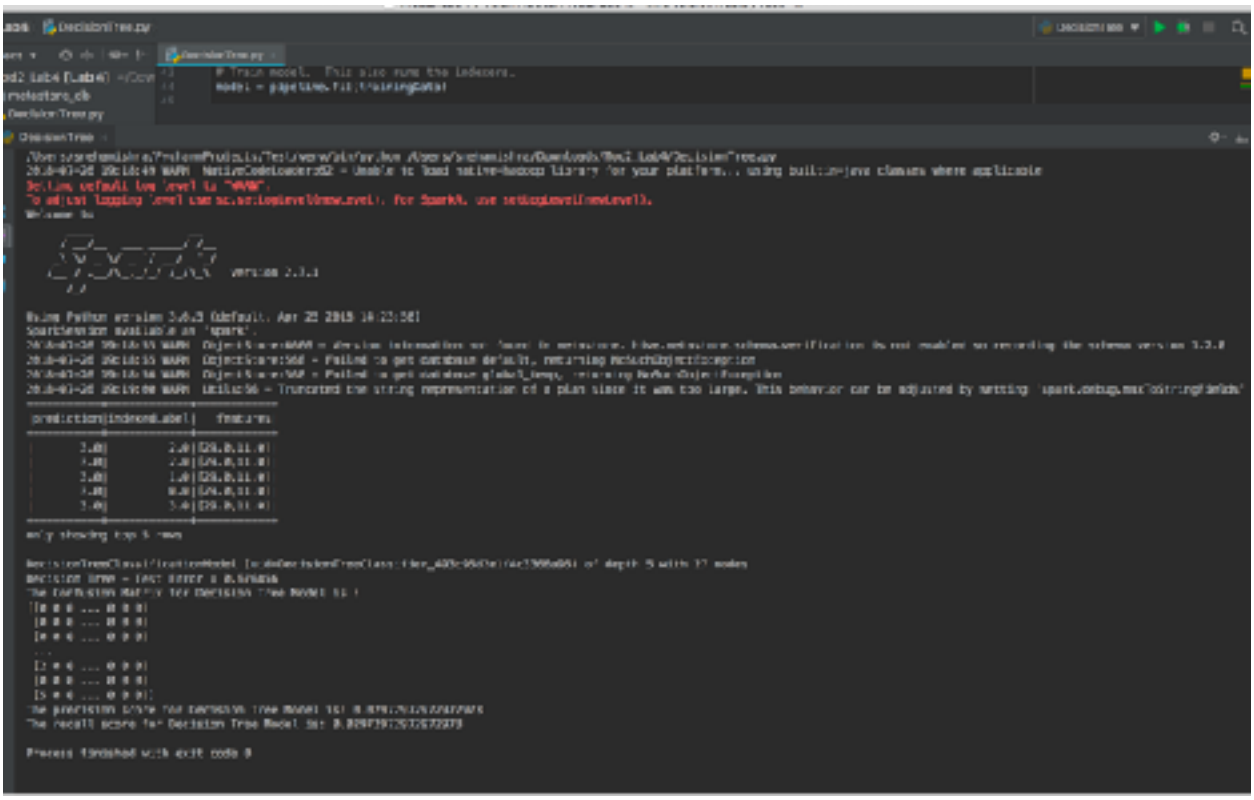
2. Decision Tree:

Code for the Algorithm:



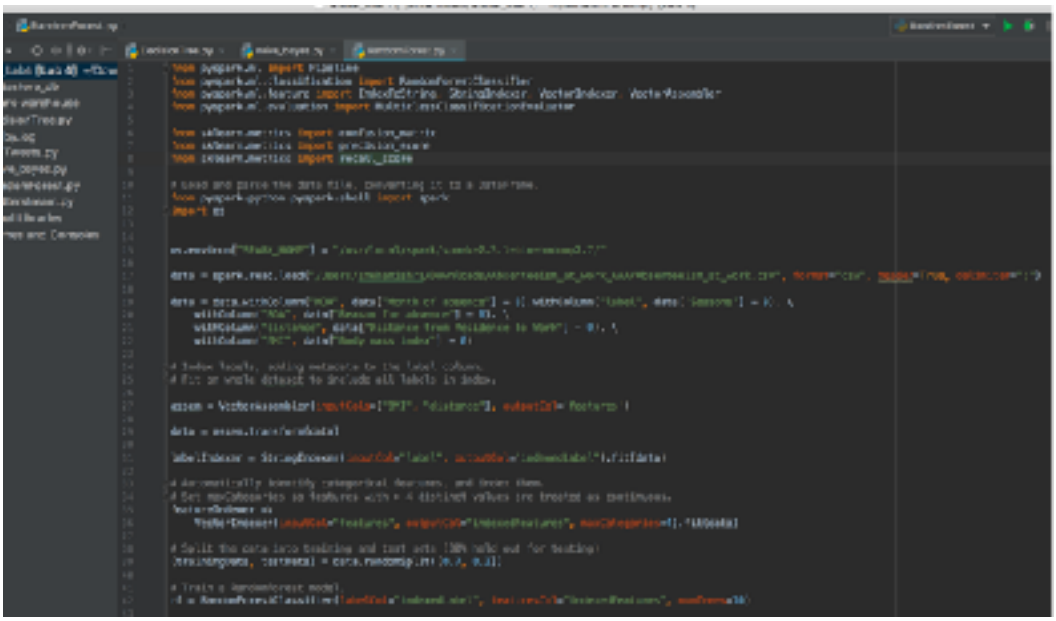


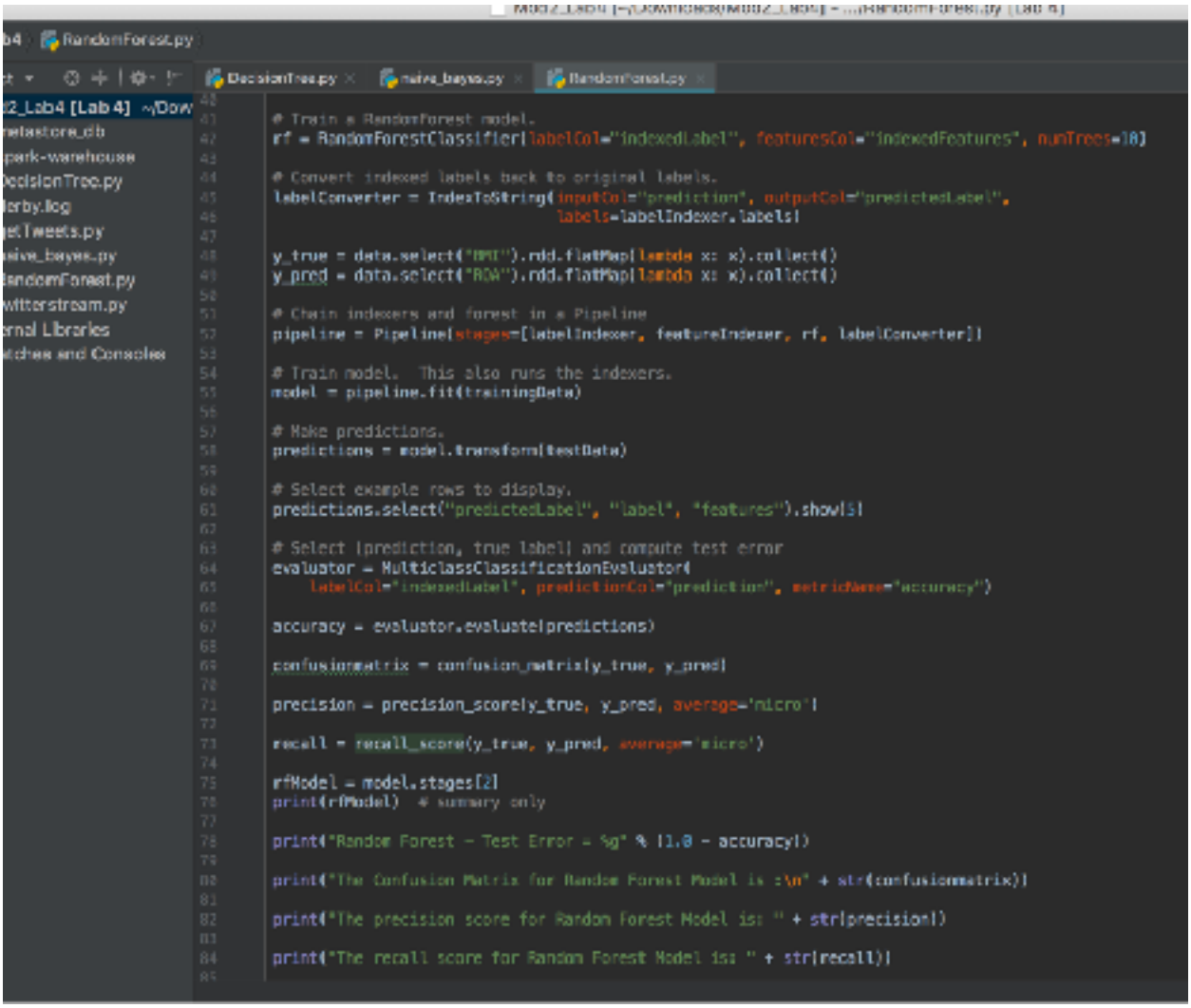
Output after running the Algorithm:



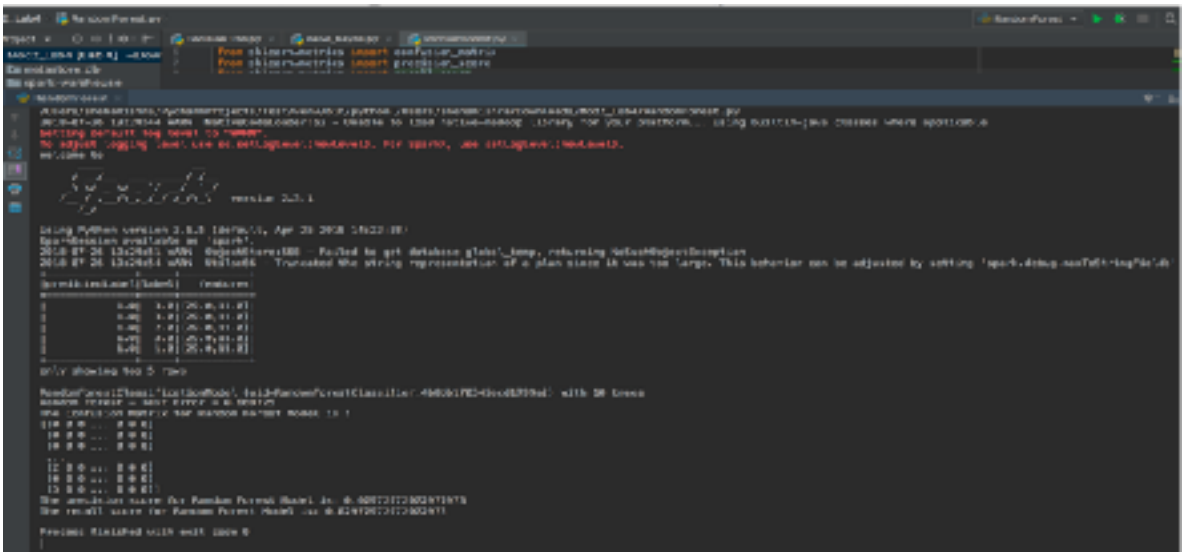
3. Random Tree:

Code for the Algorithm:





Output after running the Algorithm:



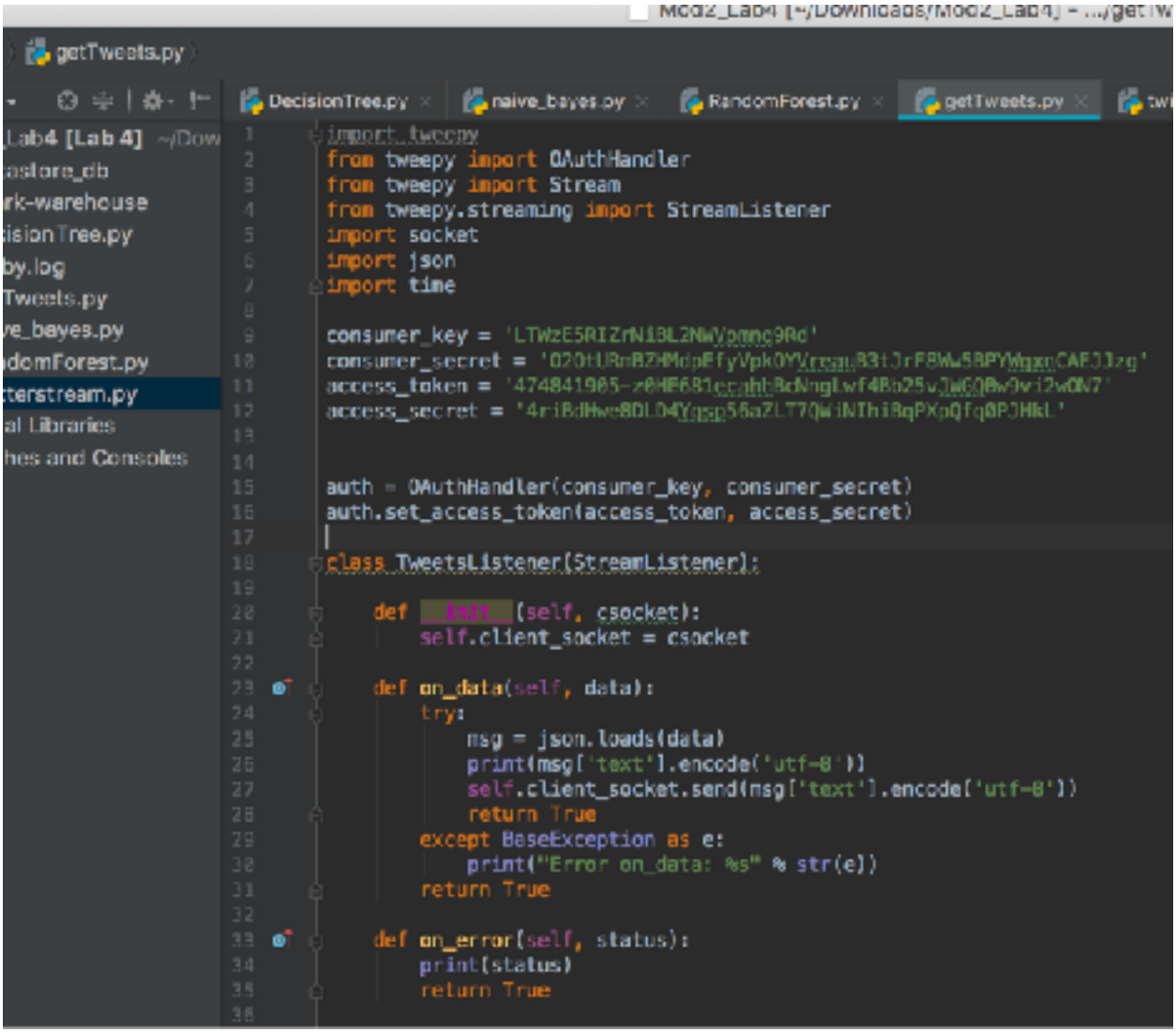
State the reasons on why one of algorithms out performs the rest:

The results indicate that the classification accuracy comparison between Naïve Bayes, Random Forest and Decision Trees that Decision Tree has got the highest average accuracy value than the Naïve Bayes and Random Forest but the difference is not statistically significant.

Part 2: Spark Streaming Task

In this task we perform Word-Count on Twitter Streaming Data using Spark. First we get the Twitter data and then we stream it.

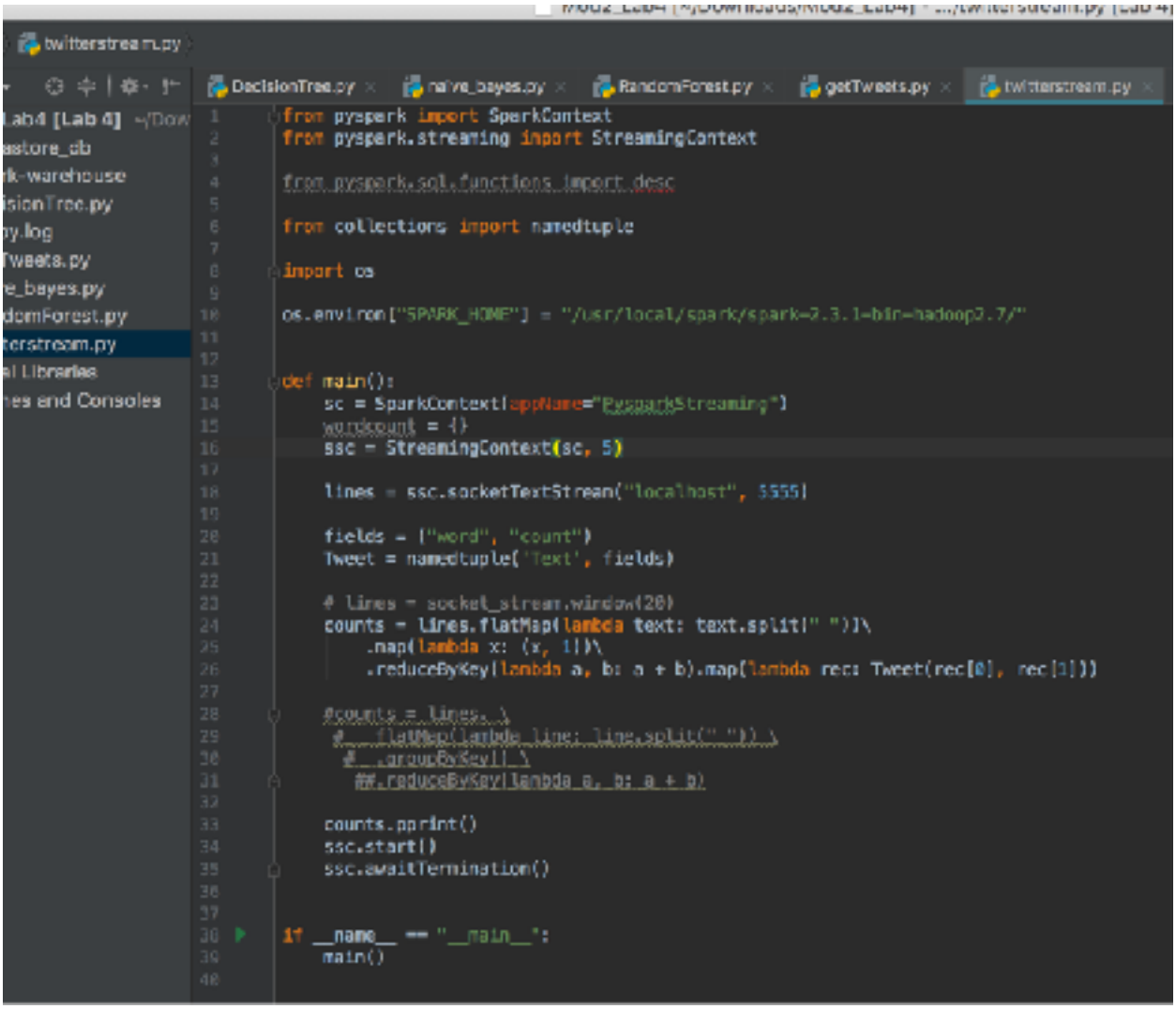
Collecting Tweets code:


getTweets.py
DecisionTree.py
naive_bayes.py
RandomForest.py
getTweets.py
Lab4 [Lab 4]
astore_db
rk-warehouse
isionTree.py
by.log
Tweets.py
ve_bayes.py
domForest.py
terstream.py
al Libraries
hes and Consoles
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import time
consumer_key = 'LTWzE5RIZrN1BL2NWVpmnc9Rc'
consumer_secret = '020tURnBZHMdpEfyVpk0YVrrauB3tJrF8Wu58PYWgxrCAEJJzg'
access_token = '474841905-z0HF681ccqhbRcNngLwf4Bb25vJwGQBwOvi2w0N7'
access_secret = '4riBdHwe8DLO4YgsP56a7LT7QwiNThiRqPXpQfq0PJHkL'
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
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

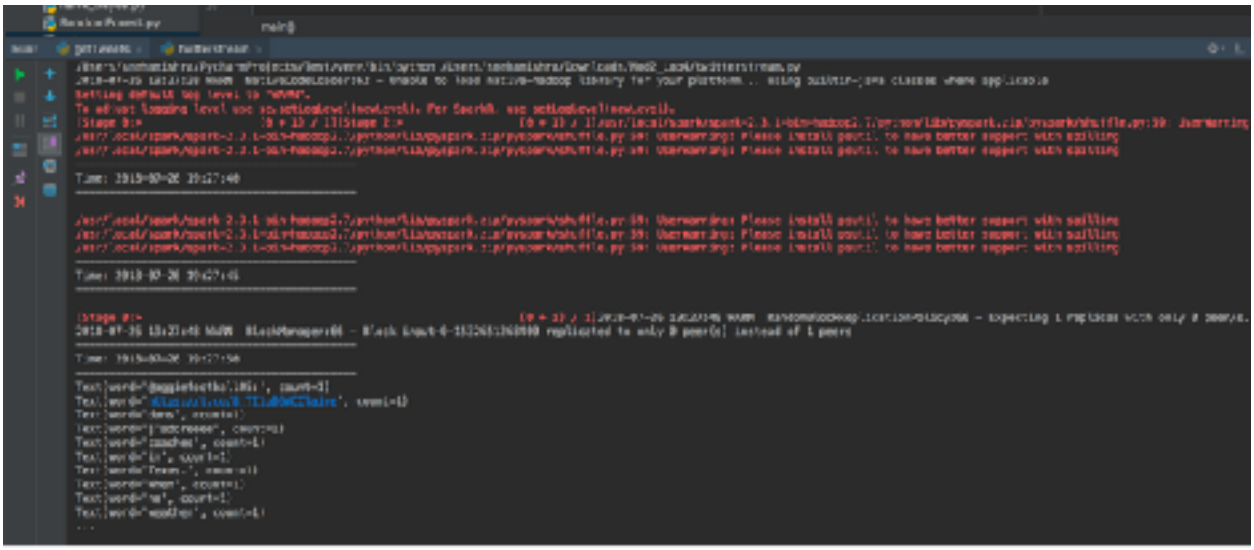
```
        self.client_socket = csocket

    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=['football'])


if __name__ == "__main__":
    s = socket.socket()  # Create a socket object
    host = "localhost"  # Get local machine name
    port = 5555  # Reserve a port for your service.
    s.bind((host, port))  # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)  # Now wait for client connection.
    c, addr = s.accept()  # Establish connection with client.

    print("Received request from: " + str(addr))

    time.sleep(5)

    sendData(c)
```

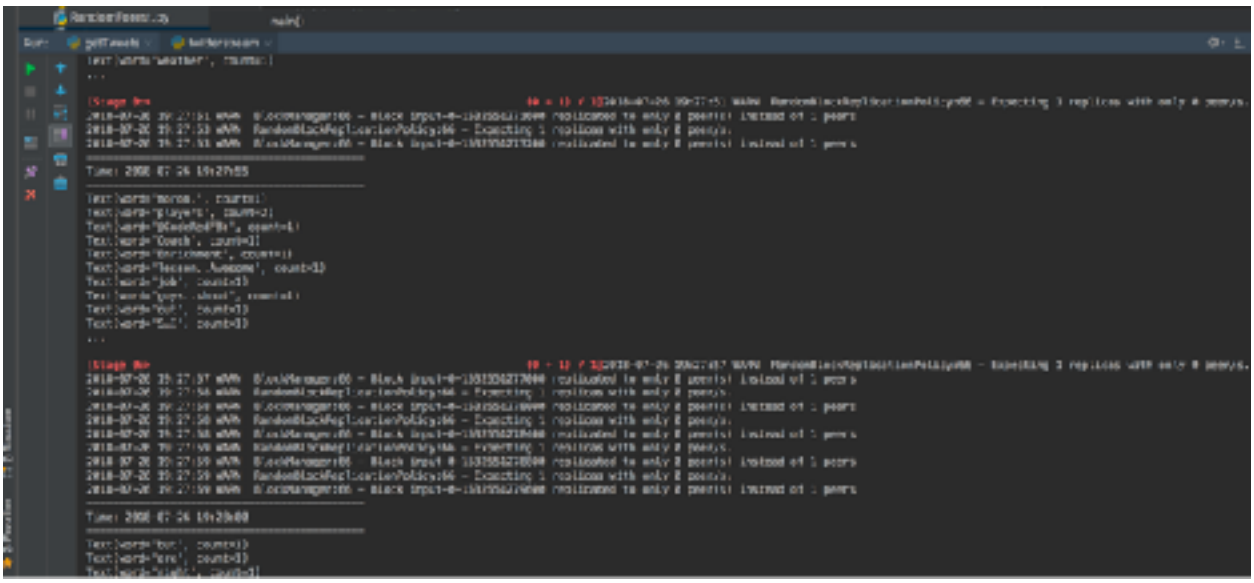
Output of Collecting Tweets code:

Stream Twitter data code:



Output of the Twitter Streaming data:





References:

1. https://www.linkedin.com/pulse/apache-spark-streaming-twitter-python-laurent-weichberger/
2. https://github.com/stefanobaghino/spark-twitter-stream-example

Data-sets provided:

1. Absenteeism at work:
   https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work

2. Immunotherapy Dataset:
   https://archive.ics.uci.edu/ml/datasets/Immunotherapy+Dataset

3. Nashville-meetup Dataset:
   https://www.kaggle.com/stkbailey/nashville-meetup

4. Word Game Dataset:
   https://www.kaggle.com/anneloes/wordgame

5. Cyber Crime Motive:
   https://www.kaggle.com/sunilkumarsv/indiacybercrimestats2013