

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.

Part 1: Spark Classification Task

1. Naïve Bayes:

Code for the Algorithm:

[illegible]

2. Decision Tree:

Code for the Algorithm:

```
1 from pytorch.nn import Pipeline
2 from pytorch.nn.classification import DecisionTreeClassifier
3 from pytorch.nn.functional import Softplus, Softmax, Softmax2d
4 from pytorch.nn.functional import Softplus, Softmax, Softmax2d
5
6 from sklearn.metrics import accuracy_score
7 from sklearn.metrics import precision_score
8 from sklearn.metrics import recall_score
9
10 # Load the data stored in a CSV format as a DataFrame.
11 from pytorch.nn.classification import Softplus, Softmax, Softmax2d
12 from pytorch.nn.functional import Softplus, Softmax, Softmax2d
13
14 # Load the data
15 data = torch.load("data/decision_tree_data.pt")
16 data = data.to(device)
17 data = data.to(device)
18 data = data.to(device)
19 data = data.to(device)
20 data = data.to(device)
21 data = data.to(device)
22 data = data.to(device)
23 data = data.to(device)
24 data = data.to(device)
25 data = data.to(device)
26 data = data.to(device)
27 data = data.to(device)
28 data = data.to(device)
29 data = data.to(device)
30 data = data.to(device)
31 data = data.to(device)
32 data = data.to(device)
33 data = data.to(device)
34 data = data.to(device)
35 data = data.to(device)
36 data = data.to(device)
37 data = data.to(device)
38 data = data.to(device)
39 data = data.to(device)
40 data = data.to(device)
41 data = data.to(device)
42 data = data.to(device)
43 data = data.to(device)
44 data = data.to(device)
45 data = data.to(device)
46 data = data.to(device)
47 data = data.to(device)
48 data = data.to(device)
49 data = data.to(device)
50 data = data.to(device)
51 data = data.to(device)
52 data = data.to(device)
53 data = data.to(device)
54 data = data.to(device)
55 data = data.to(device)
56 data = data.to(device)
57 data = data.to(device)
58 data = data.to(device)
59 data = data.to(device)
60 data = data.to(device)
61 data = data.to(device)
62 data = data.to(device)
63 data = data.to(device)
64 data = data.to(device)
65 data = data.to(device)
66 data = data.to(device)
67 data = data.to(device)
68 data = data.to(device)
69 data = data.to(device)
70 data = data.to(device)
71 data = data.to(device)
72 data = data.to(device)
73 data = data.to(device)
74 data = data.to(device)
75 data = data.to(device)
76 data = data.to(device)
77 data = data.to(device)
78 data = data.to(device)
79 data = data.to(device)
80 data = data.to(device)
81 data = data.to(device)
82 data = data.to(device)
83 data = data.to(device)
84 data = data.to(device)
85 data = data.to(device)
86 data = data.to(device)
87 data = data.to(device)
88 data = data.to(device)
89 data = data.to(device)
90 data = data.to(device)
91 data = data.to(device)
92 data = data.to(device)
93 data = data.to(device)
94 data = data.to(device)
95 data = data.to(device)
96 data = data.to(device)
97 data = data.to(device)
98 data = data.to(device)
99 data = data.to(device)
100 data = data.to(device)
```

```

tree.py
37 # Train a DecisionTree model.
38 dt = DecisionTreeClassifier(labelCol="indexedLabel", featureCol="indexedFeatures")
39
40 # Chain indexers and tree in a Pipeline
41 pipeline = Pipeline(stages=[labelIndexer, featureIndexer, dt])
42
43 # Train model. This also runs the indexers.
44 model = pipeline.fit(trainingData)
45
46 # Make predictions.
47 predictions = model.transform(testData)
48
49 # Select example rows to display.
50 predictions.select("prediction", "indexedLabel", "features").show(5)
51
52 # Select (prediction, true label) and compute test error
53 evaluator = MulticlassClassificationEvaluator(
54     labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
55
56 accuracy = evaluator.evaluate(predictions)
57
58 y_true = data.select("BMI").rdd.flatMap(Lambda x: x).collect()
59 y_pred = data.select("RDA").rdd.flatMap(Lambda x: x).collect()
60
61 confusionMatrix = confusion_matrix(y_true, y_pred)
62
63 precision = precision_score(y_true, y_pred, average='micro')
64
65 recall = recall_score(y_true, y_pred, average='micro')
66
67 treeModel = model.stages[2]
68 # summary only
69 print(treeModel)
70
71 print("Decision Tree - Test Error = %g" % (1.0 - accuracy))
72
73 print("The Confusion Matrix for DecisionTree Model is :\n" + str(confusionMatrix))
74
75 print("The precision score for Decision Tree Model is: " + str(precision))
76

```

Output after running the Algorithm:


```
MOD2_Lab4 [Lab 4] - JupyterLab (MOD2_Lab4) - .../Hadoop3-OpenJDK (Lab 4)
b4: RandomForest.py
DecisionTree.py x naive_bayes.py x RandomForest.py x
42
43 # Train a RandomForest model.
44 rf = RandomForestClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures", numTrees=18)
45
46 # Convert indexed labels back to original labels.
47 labelConverter = IndexToString(inputCol="prediction", outputCol="predictedLabel",
48                               labels=labelIndexer.labels)
49
50 y_true = data.select("RMI").rdd.flatMap(lambda x: x).collect()
51 y_pred = data.select("ROW").rdd.flatMap(lambda x: x).collect()
52
53 # Chain indexers and forest in a Pipeline
54 pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf, labelConverter])
55
56 # Train model. This also runs the indexers.
57 model = pipeline.fit(trainingData)
58
59 # Make predictions.
60 predictions = model.transform(testData)
61
62 # Select example rows to display.
63 predictions.select("predictedLabel", "label", "features").show(5)
64
65 # Select (prediction, true label) and compute test error
66 evaluator = MulticlassClassificationEvaluator(
67     labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
68
69 accuracy = evaluator.evaluate(predictions)
70
71 confusionMatrix = confusion_matrix(y_true, y_pred)
72
73 precision = precision_score(y_true, y_pred, average='micro')
74
75 recall = recall_score(y_true, y_pred, average='micro')
76
77 rfModel = model.stages[2]
78 print(rfModel) # summary only
79
80 print("Random Forest - Test Error = %g" % (1.0 - accuracy))
81
82 print("The Confusion Matrix for Random Forest Model is :\n" + str(confusionMatrix))
83
84 print("The precision score for Random Forest Model is: " + str(precision))
85
86 print("The recall score for Random Forest Model is: " + str(recall))
```

Output after running the Algorithm:


```

getTweets.py
DecisionTree.py x naive_bayes.py x RandomForest.py x getTweets.py x twi
Lab4 [Lab 4] ~/Dow
astore_db
ark-warehouse
isionTree.py
by.log
Tweets.py
ve_bayes.py
andomForest.py
terstream.py
al Libraries
hes and Consoles
1 import tweepy
2 from tweepy import OAuthHandler
3 from tweepy import Stream
4 from tweepy.streaming import StreamListener
5 import socket
6 import json
7 import time
8
9 consumer_key = 'LTWzE5RIZrN1BL2NwYpmng9Rd'
10 consumer_secret = '020HURnBZHMdpEfyVpk0YVrgauB3tJrF8Ww58PYWggnCAEJ1zg'
11 access_token = '474841905-z0HF681ggahh8cNnglwF4Bb25vJwGQBw9v12wON7'
12 access_secret = '4riBdHwe8DL04Ygsp55aZLT7QWInThi8qPKpQfq0PJHkl'
13
14
15 auth = OAuthHandler(consumer_key, consumer_secret)
16 auth.set_access_token(access_token, access_secret)
17
18 class TweetsListener(StreamListener):
19
20     def __init__(self, csocket):
21         self.client_socket = csocket
22
23     def on_data(self, data):
24         try:
25             msg = json.loads(data)
26             print(msg['text'].encode('utf-8'))
27             self.client_socket.send(msg['text'].encode('utf-8'))
28             return True
29         except BaseException as e:
30             print("Error on_data: %s" % str(e))
31             return True
32
33     def on_error(self, status):
34         print(status)
35         return True
36

```


Stream Twitter data code:

```
1 from pyspark import SparkContext
2 from pyspark.streaming import StreamingContext
3
4 from pyspark.sql.functions import desc
5
6 from collections import namedtuple
7
8 import os
9
10 os.environ["SPARK_HOME"] = "/usr/local/spark/spark-2.3.1-bin-hadoop2.7/"
11
12
13 def main():
14     sc = SparkContext(appName="PySparkStreaming")
15     wconf = {}
16     ssc = StreamingContext(sc, 5)
17
18     lines = ssc.socketTextStream("localhost", 5555)
19
20     fields = ("word", "count")
21     Tweet = namedtuple('Tweet', fields)
22
23     # lines = socket_stream.window(20)
24     counts = lines.flatMap(lambda text: text.split(" "))\
25         .map(lambda x: (x, 1))\
26         .reduceByKey(lambda a, b: a + b).map(lambda rec: Tweet(rec[0], rec[1]))
27
28     # counts = lines.\
29     #     .flatMap(lambda line: line.split(" "))\
30     #     .groupByKey()\
31     #     .reduceByKey(lambda a, b: a + b)
32
33     counts.pprint()
34     ssc.start()
35     ssc.awaitTermination()
36
37
38 if __name__ == "__main__":
39     main()
40
```

[illegible]

```

Run: gptfwc -l /usr/share/...
...
[Stage 0]
2018-10-26 19:27:53 wfh s/cosmanager - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
2018-10-26 19:27:53 wfh RandomBlackReplicationPolicy:06 - Exacting 1 replicas with only 0 peers;
2018-10-26 19:27:53 wfh s/cosmanager:off - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
Time: 2018-10-26 19:27:55
Text[word="score", count=1]
Text[word="play", count=1]
Text[word="BeebleBrox", count=1]
Text[word="Cook", count=1]
Text[word="crisscross", count=1]
Text[word="Harris, Antonio", count=1]
Text[word="Jok", count=1]
Text[word="guy", count=1]
Text[word="out", count=1]
Text[word="Tut", count=1]
...
[Stage 0]
2018-10-26 19:27:57 wfh s/cosmanager:off - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
2018-10-26 19:27:58 wfh RandomBlackReplicationPolicy:06 - Exacting 1 replicas with only 0 peers;
2018-10-26 19:27:58 wfh s/cosmanager - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
2018-10-26 19:27:58 wfh RandomBlackReplicationPolicy:06 - Exacting 1 replicas with only 0 peers;
2018-10-26 19:27:58 wfh s/cosmanager:off - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
2018-10-26 19:27:58 wfh RandomBlackReplicationPolicy:06 - Exacting 1 replicas with only 0 peers;
2018-10-26 19:27:58 wfh s/cosmanager - Black input=0-1037627760 replicated to only 0 peers; limited of 1 peers
Time: 2018-10-26 19:28:00
Text[word="top", count=1]
Text[word="Ten", count=1]
Text[word="", count=1]

```

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>