

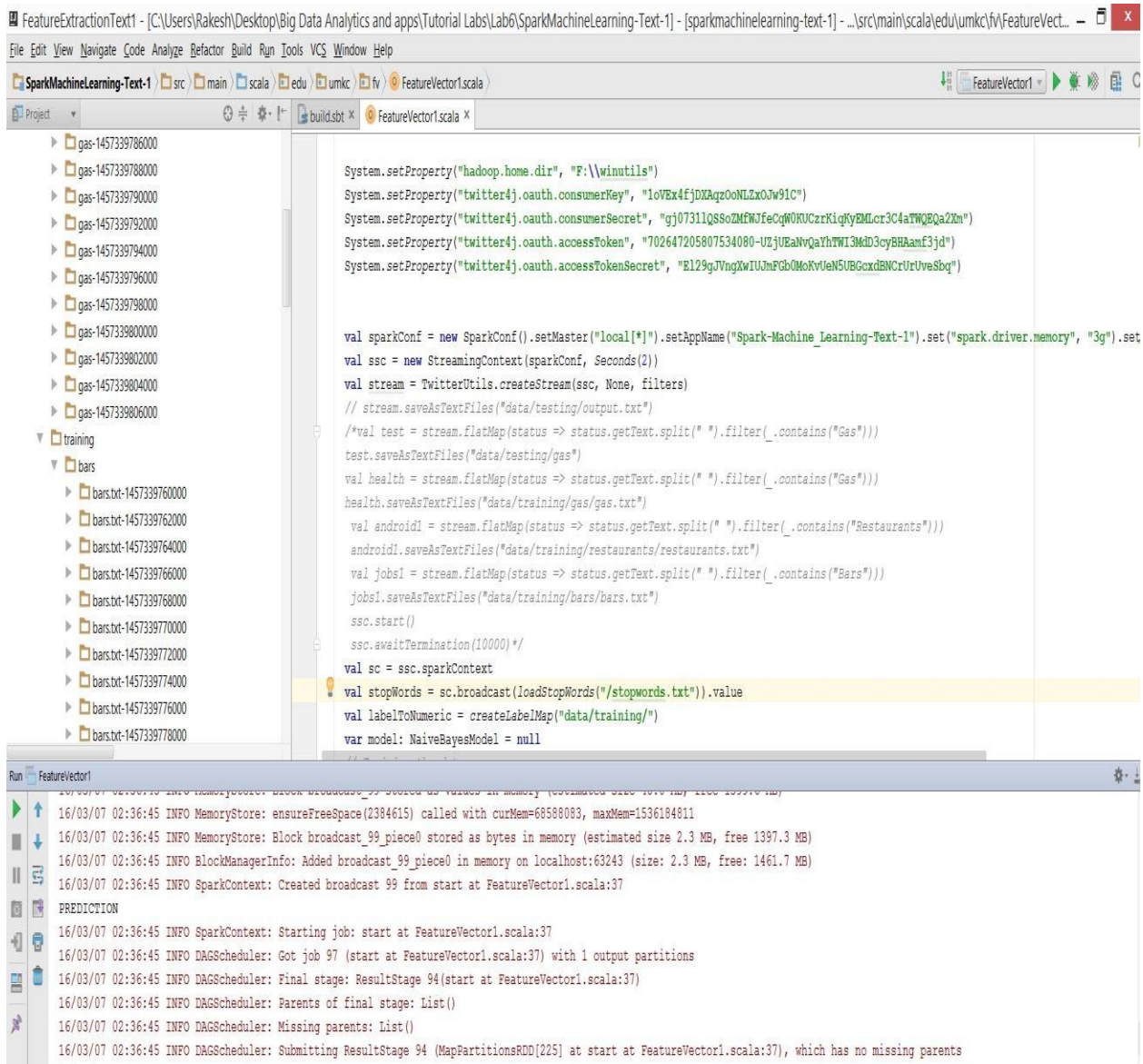
CS 5542 Big Data Applications and Analytics

LAB Assignment #5 & #6

-Sricharan Pamuru (20)

2. Spark ML Lib Application

The below screenshot depicts the training data provided in terms of three constraints, gas, bars and restaurants. Prediction is found in the console and it can be visualized as follows:



The screenshot shows an IDE window titled "FeatureExtractionText1" with a Scala file "FeatureVector1.scala". The code defines a Spark application that sets up properties, creates a streaming context, and processes data from Twitter. It filters data based on "Gas", "Restaurants", and "Bars" and saves them to text files. The code also includes a prediction step using a NaiveBayesModel.

```
System.setProperty("hadoop.home.dir", "F:\\winutils")
System.setProperty("twitter4j.oauth.consumerKey", "1oVEx4fjDXAgz0oNLZxOjw91C")
System.setProperty("twitter4j.oauth.consumerSecret", "gj07311Q8SoZMFwJfeCqWOKUCzrK1qKyEMLcr3C4aTWQBQa2Xm")
System.setProperty("twitter4j.oauth.accessToken", "702647205807534080-UZjUEaNVqYhTWI3MdD3cyBHAamE3jd")
System.setProperty("twitter4j.oauth.accessTokenSecret", "E129gJVngXwTUJmFGb0MoKvUeN5UBGcxDBNCrUtrUveSbq")

val sparkConf = new SparkConf().setMaster("local[*]").setAppName("Spark-Machine_Learning-Text-1").set("spark.driver.memory", "3g").set
val ssc = new StreamingContext(sparkConf, Seconds(2))
val stream = TwitterUtils.createStream(ssc, None, filters)
// stream.saveAsTextFiles("data/testing/output.txt")
/*val test = stream.flatMap(status => status.getText.split(" ").filter(_.contains("Gas")))
test.saveAsTextFiles("data/testing/gas")
val health = stream.flatMap(status => status.getText.split(" ").filter(_.contains("Gas")))
health.saveAsTextFiles("data/training/gas/gas.txt")
val android1 = stream.flatMap(status => status.getText.split(" ").filter(_.contains("Restaurants")))
android1.saveAsTextFiles("data/training/restaurants/restaurants.txt")
val jobs1 = stream.flatMap(status => status.getText.split(" ").filter(_.contains("Bars")))
jobs1.saveAsTextFiles("data/training/bars/bars.txt")
ssc.start()
ssc.awaitTermination(10000)*/
val sc = ssc.sparkContext
val stopWords = sc.broadcast(loadStopWords("/stopwords.txt")).value
val labelToNumeric = createLabelMap("data/training/")
var model: NaiveBayesModel = null
```

The console output shows the execution of the application, including memory management and the prediction results:

```
16/03/07 02:36:45 INFO MemoryStore: ensureFreeSpace(2384615) called with curMem=68588083, maxMem=1536184611
16/03/07 02:36:45 INFO MemoryStore: Block broadcast_99_piece0 stored as bytes in memory (estimated size 2.3 MB, free 1397.3 MB)
16/03/07 02:36:45 INFO BlockManagerInfo: Added broadcast_99_piece0 in memory on localhost:63243 (size: 2.3 MB, free: 1461.7 MB)
16/03/07 02:36:45 INFO SparkContext: Created broadcast 99 from start at FeatureVector1.scala:37
PREDICTION
16/03/07 02:36:45 INFO SparkContext: Starting job: start at FeatureVector1.scala:37
16/03/07 02:36:45 INFO DAGScheduler: Got job 97 (start at FeatureVector1.scala:37) with 1 output partitions
16/03/07 02:36:45 INFO DAGScheduler: Final stage: ResultStage 94(start at FeatureVector1.scala:37)
16/03/07 02:36:45 INFO DAGScheduler: Parents of final stage: List()
16/03/07 02:36:45 INFO DAGScheduler: Missing parents: List()
16/03/07 02:36:45 INFO DAGScheduler: Submitting ResultStage 94 (MapPartitionsRDD[225] at start at FeatureVector1.scala:37), which has no missing parents
```

Prediction output is displayed and the bars are obtained as output based on data collected. Comparison with training data brings a combined result.

The screenshot shows an IDE window titled "FeatureExtractionText1 - [C:\Users\Rakesh\Desktop\Big Data Analytics and apps\Tutorial Labs\Lab6\SparkMachineLearning-Text-1] - [sparkmachinelearning-text-1] - ...". The main editor displays the file "FeatureVector1.scala" with the following code:

```
System.setProperty("hadoop.home.dir", "F:\winutils")
System.setProperty("twitter4j.oauth.consumerKey", "1oVEx4fjdXAgzOoNLzX0Jw91C")
System.setProperty("twitter4j.oauth.consumerSecret", "gj0731lQSSoZMfWJFeCqW0KUCzrKiqKyEMLor3C4aTWQ8Qa2Xm")
System.setProperty("twitter4j.oauth.accessToken", "702647205807534080-UZjUEaVnQaYhTWI3MdD3cyBHAanf3jd")
System.setProperty("twitter4j.oauth.accessTokenSecret", "E129gJVngXwIUJmFGb0MoKVUeN5UBGcxdBNCrUrUveSbq")

val sparkConf = new SparkConf().setMaster("local[*]").setAppName("Spark-Machine_Learning-Text-1").set("spark.driver.memory", "3g").set(...)
val ssc = new StreamingContext(sparkConf, Seconds(2))
val stream = TwitterUtils.createStream(ssc, None, filters)
// stream.saveAsTextFiles("data/testing/output.txt")
/*val test = stream.flatMap(status => status.getText.split(" ").filter(_._contains("Gas"))
test.saveAsTextFiles("data/testing/gas")
val health = stream.flatMap(status => status.getText.split(" ").filter(_._contains("Gas"))
health.saveAsTextFiles("data/training/gas/gas.txt")
val android1 = stream.flatMap(status => status.getText.split(" ").filter(_._contains("Restaurants")))
android1.saveAsTextFiles("data/training/restaurants/restaurants.txt")
val jobs1 = stream.flatMap(status => status.getText.split(" ").filter(_._contains("Bars")))
jobs1.saveAsTextFiles("data/training/bars/bars.txt")
ssc.start()
ssc.awaitTermination(10000)*/
val sc = ssc.sparkContext
val stopWords = sc.broadcast(loadStopWords("/stopwords.txt")).value
val labelToNumeric = createLabelMap("data/training/")
var model: NaiveBayesModel = null
```

The left sidebar shows a project tree with folders like "gas-1457339786000", "bars", and "training". The bottom window shows the output of the program, which is a list of "bars" repeated multiple times.