>>> sc = SparkContext.getOrCreate()

>>> spark = SparkSession(sc)

>>> csv = spark.read.option("sep", "\t").csv('azureData3.csv', header=True, inferSchema=True)

>>> csv.createTempView('mytable')

>>> spark.sql('select count(\*) from mytable').show()

+--------+

|count(1)|

+--------+

| 33183|

+--------+

>>> csv.columns

['processed\_text', 'y']

>>> from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer

>>> from pyspark.ml.classification import LogisticRegression

>>>

>>> # regular expression tokenizer

... regexTokenizer = RegexTokenizer(inputCol="processed\_text", outputCol="words", pattern="\\s+")

>>>

>>> # stop words

... add\_stopwords = ["http","https","amp","rt","t","c","can", # standard stop words

... "#keithlamontscott","#charlotteprotest","#charlotteriots","#keithscott"] # keywords used to pull data)

>>> stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setStopWords(add\_stopwords)

>>>

>>> # bag of words count

... countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocabSize=10000, minDF=5)

>>> from pyspark.ml import Pipeline

>>>

>>> pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors])

>>>

>>> # Fit the pipeline to training documents.

... pipelineFit = pipeline.fit(csv)

>>> dataset = pipelineFit.transform(csv)

>>> dataset.dtypes

[('processed\_text', 'string'), ('y', 'int'), ('words', 'array<string>'), ('filtered', 'array<string>'), ('features', 'vector')]

>>> dataset.show()

+--------------------+---+--------------------+--------------------+--------------------+

| processed\_text| y| words| filtered| features|

+--------------------+---+--------------------+--------------------+--------------------+

|porcine epidemic ...| 1|[porcine, epidemi...|[porcine, epidemi...|(10000,[0,1,2,5,6...|

|generation rndvs ...| 1|[generation, rndv...|[generation, rndv...|(10000,[0,1,2,5,6...|

|incidence complet...| 0|[incidence, compl...|[incidence, compl...|(10000,[1,2,4,5,8...|

|arenaviruses enve...| 1|[arenaviruses, en...|[arenaviruses, en...|(10000,[0,1,2,5,6...|

|study anti dengue...| 1|[study, anti, den...|[study, anti, den...|(10000,[0,1,2,3,4...|

|microglia special...| 1|[microglia, speci...|[microglia, speci...|(10000,[0,1,2,5,6...|

|a1111111111 a1111...| 1|[a1111111111, a11...|[a1111111111, a11...|(10000,[0,1,2,5,6...|

|pcr technique wid...| 1|[pcr, technique, ...|[pcr, technique, ...|(10000,[2,3,4,6,8...|

|abstract ebola vi...| 1|[abstract, ebola,...|[abstract, ebola,...|(10000,[0,1,2,5,6...|

|decade considerab...| 1|[decade, consider...|[decade, consider...|(10000,[0,1,2,5,6...|

|nuclear factor ka...| 0|[nuclear, factor,...|[nuclear, factor,...|(10000,[0,1,5,7,8...|

|a1111111111 a1111...| 1|[a1111111111, a11...|[a1111111111, a11...|(10000,[0,1,2,3,4...|

|electron microsco...| 1|[electron, micros...|[electron, micros...|(10000,[0,1,2,3,4...|

|avian influenza a...| 1|[avian, influenza...|[avian, influenza...|(10000,[0,1,2,5,6...|

|swine origin infl...| 1|[swine, origin, i...|[swine, origin, i...|(10000,[0,1,2,3,4...|

|overall structure...| 1|[overall, structu...|[overall, structu...|(10000,[0,2,6,7,8...|

|igg concentration...| 1|[igg, concentrati...|[igg, concentrati...|(10000,[0,1,2,6,7...|

|effective communi...| 0|[effective, commu...|[effective, commu...|(10000,[2,3,4,5,7...|

|intestinal parasi...| 1|[intestinal, para...|[intestinal, para...|(10000,[0,1,2,3,4...|

|viruses small sel...| 1|[viruses, small, ...|[viruses, small, ...|(10000,[0,1,2,3,4...|

+--------------------+---+--------------------+--------------------+--------------------+

only showing top 20 rows

>>> dataset.createTempView('newtable')

>>> spark.sql('select distinct y from newtable limit 5').show()

+---+

| y|

+---+

| 1|

| 0|

+---+

>>> from pyspark.ml.classification import LogisticRegression

>>> # Build the model

... lr = LogisticRegression(labelCol='y', maxIter=20, regParam=0.3, elasticNetParam=0, family = "binomial")

>>>

>>> # Train model with Training Data

... lrModel = lr.fit(trainingData)

[Stage 25:=> (1 + 31) / 32]20/05/11 16:17:31 WARN BlockManager: Putting block rdd\_57\_20 failed due to an exception

20/05/11 16:17:31 WARN BlockManager: Putting block rdd\_57\_14 failed due to an exception

20/05/11 16:17:31 WARN BlockManager: Block rdd\_57\_14 could not be removed as it was not found on disk or in memory

20/05/11 16:17:31 WARN BlockManager: Block rdd\_57\_20 could not be removed as it was not found on disk or in memory

20/05/11 16:17:32 ERROR Executor: Exception in task 14.0 in stage 25.0 (TID 451)

java.lang.OutOfMemoryError: Java heap space

>>> pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors])

>>>

>>> # Fit the pipeline to training documents.

... pipelineFit = pipeline.fit(csv)

>>> # Transform dataset with new pipelined features.

... dataset = pipelineFit.transform(csv)

>>> dataset.dtypes

[('processed\_text', 'string'), ('y', 'int'), ('words', 'array<string>'), ('filtered', 'array<string>'), ('features', 'vector')]

>>> (trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed = 100)

>>> print("Training Dataset Count: " + str(trainingData.count()))

Training Dataset Count: 23287

>>> print("Test Dataset Count: " + str(testData.count()))

Test Dataset Count: 9896

>>> from pyspark.ml.classification import DecisionTreeClassifier

>>>

>>> # Create initial Decision Tree Model

... dt = DecisionTreeClassifier(labelCol="y", featuresCol="features", maxDepth=3)

>>> # Only for cli

... # dt = DecisionTreeClassifier(labelCol="y", featuresCol="features")

... # Train model with Training Data

... dtModel = dt.fit(trainingData)

[Stage 17:> (0 + 32) / 32]20/05/11 16:25:30 ERROR Executor: Exception in task 2.0 in stage 17.0 (TID 271)

java.lang.OutOfMemoryError: GC overhead limit exceeded

>>> from pyspark.ml.classification import RandomForestClassifier

>>>

>>> # Create an initial RandomForest model.

... rf = RandomForestClassifier(labelCol="y", \

... featuresCol="features")

>>>

>>> # Train model with Training Data

... rfModel = rf.fit(trainingData)

[Stage 15:> (0 + 32) / 32]20/05/11 16:46:20 ERROR Executor: Exception in task 29.0 in stage 15.0 (TID 265)

java.lang.OutOfMemoryError: Java heap space

>>> from pyspark.ml.classification import NaiveBayes

>>>

>>> # create the trainer and set its parameters

... nb = NaiveBayes(labelCol="y", smoothing=1, modelType="multinomial")

>>>

>>> # train the model

... model = nb.fit(trainingData)

[Stage 14:======================================================> (31 + 1) / 32]20/05/11 16:52:23 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS

20/05/11 16:52:23 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

>>> predictions = model.transform(testData)

>>> from pyspark.sql.functions import col

>>> see = predictions.select('y', col('y').alias('label'), 'rawPrediction')

>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator

>>> evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")

>>> print("Test: Area Under ROC: " + str(evaluator.evaluate(see, {evaluator.metricName: "areaUnderROC"})))

Test: Area Under ROC: 0.383833417424