pileline2

Cmd 1

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.feature.{HashingTF, Tokenizer}
import org.apache.spark.ml.linalg.Vector
import org.apache.spark.ml.tuning.{CrossValidator, ParamGridBuilder}
import org.apache.spark.sql.Row

// Prepare training data from a list of (id, text, label) tuples.
val training = spark.createDataFrame(Seq(
  (0L, "a b c d e spark", 1.0),
  (1L, "b d", 0.0),
  (2L, "spark f g h", 1.0),
  (3L, "hadoop mapreduce", 0.0),
  (4L, "b spark who", 1.0),
  (5L, "g d a y", 0.0),
  (6L, "spark fly", 1.0),
  (7L, "was mapreduce", 0.0),
  (8L, "e spark program", 1.0),
  (9L, "a e c l", 0.0),
  (10L, "spark compile", 1.0),
  (11L, "hadoop software", 0.0)
)).toDF("id", "text", "label")
```

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.feature.{HashingTF, Tokenizer}
import org.apache.spark.ml.linalg.Vector
import org.apache.spark.ml.tuning.{CrossValidator, ParamGridBuilder}
import org.apache.spark.sql.Row
training: org.apache.spark.sql.DataFrame = [id: bigint, text: string ... 1 more
 field]
```

Command took 11.46 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:13:43 on My Cluster

Cmd 2

```
val tokenizer = new Tokenizer()
  .setInputCol("text")
  .setOutputCol("words")
val hashingTF = new HashingTF()
  .setInputCol(tokenizer.getOutputCol)
  .setOutputCol("features")
val lr = new LogisticRegression()
  .setMaxIter(10)
val pipeline = new Pipeline()
  .setStages(Array(tokenizer, hashingTF, lr))
```

```
tokenizer: org.apache.spark.ml.feature.Tokenizer = tok_3ec513bfc626
hashingTF: org.apache.spark.ml.feature.HashingTF = hashingTF_733efc37aa6c
lr: org.apache.spark.ml.classification.LogisticRegression = logreg_86ad4935b1d6
pipeline: org.apache.spark.ml.Pipeline = pipeline_b134e75223e5
```

Command took 0.98 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:15:58 on My Cluster

Cmd 3

```
val paramGrid = new ParamGridBuilder()
  .addGrid(hashingTF.numFeatures, Array(10, 100, 1000))
  .addGrid(lr.regParam, Array(0.1, 0.01))
  .build()
```

```
paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
        hashingTF_733efc37aa6c-numFeatures: 10,
        logreg_86ad4935b1d6-regParam: 0.1
}, {
        hashingTF_733efc37aa6c-numFeatures: 100,
        logreg_86ad4935b1d6-regParam: 0.1
}, {
        hashingTF_733efc37aa6c-numFeatures: 1000,
        logreg_86ad4935b1d6-regParam: 0.1
}, {
        hashingTF_733efc37aa6c-numFeatures: 10,
        logreg_86ad4935b1d6-regParam: 0.01
}, {
        hashingTF_733efc37aa6c-numFeatures: 100,
        logreg_86ad4935b1d6-regParam: 0.01
}, {
        hashingTF_733efc37aa6c-numFeatures: 1000,
        logreg_86ad4935b1d6-regParam: 0.01
})
```

Command took 0.31 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:16:20 on My Cluster

Cmd 4

```
val cv = new CrossValidator()
  .setEstimator(pipeline)
  .setEvaluator(new BinaryClassificationEvaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(2)  // Use 3+ in practice
```

```
cv: org.apache.spark.ml.tuning.CrossValidator = cv_02427d8afe99
```

```
Command took 0.24 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:17:04 on My Cluster
```

Cmd 5

```
val cvModel = cv.fit(training)
```

▸ (41) Spark Jobs

```
cvModel: org.apache.spark.ml.tuning.CrossValidatorModel = cv_02427d8afe99
```

```
Command took 1.37 minutes -- by louhy1128@gmail.com at 2017/7/3 下午3:18:09 on My Cluster
```

Cmd 6

```
cvModel.numFolds
```

```
res0: org.apache.spark.ml.param.IntParam = cv_02427d8afe99__numFolds
```

```
Command took 0.13 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:21:09 on My Cluster
```

Cmd 7

```
val test = spark.createDataFrame(Seq(
  (4L, "spark i j k"),
  (5L, "l m n"),
  (6L, "mapreduce spark"),
  (7L, "apache hadoop")
)).toDF("id", "text")

// Make predictions on test documents. cvModel uses the best model found (lrModel).
cvModel.transform(test)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach { case Row(id: Long, text: String, prob: Vector, prediction: Double) =>
    println(s"($id, $text) --> prob=$prob, prediction=$prediction")
  }
```

```
(4, spark i j k) --> prob=[0.12566260711357224,0.8743373928864279], prediction=1.0
(5, l m n) --> prob=[0.995215441016286,0.004784558983714], prediction=0.0
(6, mapreduce spark) --> prob=[0.30696895232625965,0.6930310476737404], prediction=1.0
(7, apache hadoop) --> prob=[0.8040279442401378,0.19597205575986223], prediction
```

```
=0.0
test: org.apache.spark.sql.DataFrame = [id: bigint, text: string]
```

Command took 1.17 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:21:36 on My Cluster

Cmd 8

```
1  import org.apache.spark.ml.evaluation.RegressionEvaluator
2  import org.apache.spark.ml.recommendation.ALS
3
4  import sys.process._
5   "wget -P /tmp
   http://www.utdallas.edu/~axn112530/cs6350/data/sample_movielens_ratings.txt
   " !!
6
```

```
--2017-07-03 20:24:48--  http://www.utdallas.edu/~axn112530/cs6350/data/sample_m
ovielens_ratings.txt
Resolving www.utdallas.edu (www.utdallas.edu)... 104.16.44.54, 104.16.43.54, 240
0:cb00:2048:1::6810:2c36, ...
Connecting to www.utdallas.edu (www.utdallas.edu)|104.16.44.54|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: ‘/tmp/sample_movielens_ratings.txt’

     0K .......... .......... .......... .                    4.27M=0.007s

2017-07-03 20:24:48 (4.27 MB/s) - ‘/tmp/sample_movielens_ratings.txt’ saved [323
63]

warning: there were 1 feature warning(s); re-run with -feature for details
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.recommendation.ALS
import sys.process._
res2: String = ""
```

Command took 0.26 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:24:48 on My Cluster

Cmd 9

```
1  case class Rating(userId: Int, movieId: Int, rating: Float, timestamp:
   Long)
```

```
defined class Rating
```

Command took 0.47 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:25:57 on My Cluster

Cmd 10

```
1  def parseRating(str: String): Rating = {
2    val fields = str.split("::")
3    assert(fields.size == 4)
4    Rating(fields(0).toInt, fields(1).toInt, fields(2).toFloat,
   fields(3).toLong)
5  }
```

```
parseRating: (str: String)Rating
```

Command took 0.22 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:28:13 on My Cluster

Cmd 11

```
1  val ratings = spark.read.textFile("file:/tmp/sample_movielens_ratings.txt")
2    .map(parseRating)
3    .toDF()
```

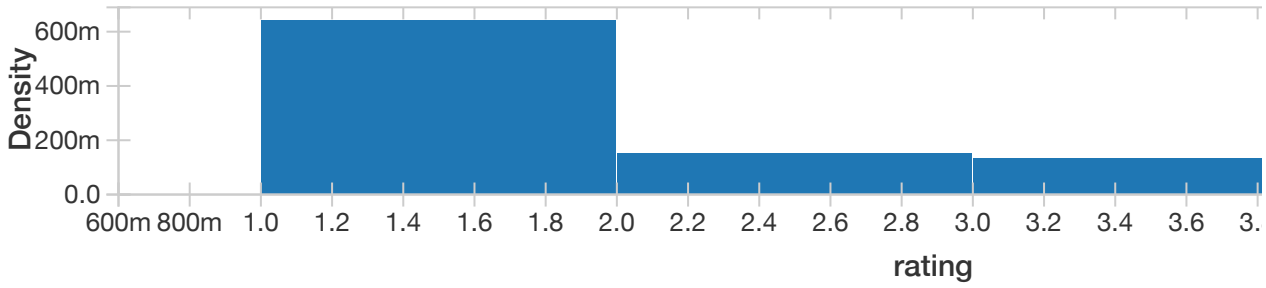
```
ratings: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more
 fields]
```

Command took 0.97 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:28:16 on My Cluster

Cmd 12

```
1  display(ratings)
```

▸ (3) Spark Jobs



Command took 1.59 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:30:30 on My Cluster

Cmd 13

```
1  val Array(training, test) = ratings.randomSplit(Array(0.8, 0.2))
```

```
training: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [userId: int,
movieId: int ... 2 more fields]
test: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [userId: int, mov
ieId: int ... 2 more fields]
```

Command took 0.25 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:32:11 on My Cluster

Cmd 14

```
1  val als = new ALS()
2    .setMaxIter(10)
3    .setRegParam(0.01)
4    .setUserCol("userId")
5    .setItemCol("movieId")
6    .setRatingCol("rating")
```

```
als: org.apache.spark.ml.recommendation.ALS = als_77efe6ede566
```

```
Command took 0.26 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:38:32 on My Cluster
```

Cmd 15

```
1  val model = als.fit(training)
```

▸ (4) Spark Jobs

```
model: org.apache.spark.ml.recommendation.ALSModel = als_77efe6ede566
```

```
Command took 1.71 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:38:33 on My Cluster
```

Cmd 16

```
1  val predictions = model.transform(test)
```

```
predictions: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 3 more fields]
```

```
Command took 0.12 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:38:37 on My Cluster
```

Cmd 17

```
1  display(predictions)
```

▸ (4) Spark Jobs

userId	movieId	rating
27	31	1
12	31	4
8	31	3
12	85	1
13	85	1
6	85	3
16	85	5
24	65	1
13	53	3

```
Command took 1.44 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:38:39 on My Cluster
```

Cmd 18

```
1 val evaluator = new RegressionEvaluator()
2   .setMetricName("rmse")
3   .setLabelCol("rating")
4   .setPredictionCol("prediction")
```

```
evaluator: org.apache.spark.ml.evaluation.RegressionEvaluator = regEval_06651e5b
b285
```

```
Command took 0.12 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:39:18 on My
Cluster
```

Cmd 19

```
1 val rmse = evaluator.evaluate(predictions)
2 println(s"Root-mean-square error = $rmse")
```

▸ (1) Spark Jobs

```
Root-mean-square error = 1.9280628704761118
rmse: Double = 1.9280628704761118
```

```
Command took 1.84 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:40:16 on My
Cluster
```

Cmd 20

```
1 import org.apache.spark.mllib.fpm.FPGrowth
2 import org.apache.spark.rdd.RDD
3
4 val data = sc.textFile("/databricks-
  datasets/samples/data/mllib/sample_fpgrowth.txt")
```

```
import org.apache.spark.mllib.fpm.FPGrowth
import org.apache.spark.rdd.RDD
data: org.apache.spark.rdd.RDD[String] = /databricks-datasets/samples/data/mlli
b/sample_fpgrowth.txt MapPartitionsRDD[1975] at textFile at <console>:50
```

```
Command took 0.13 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:42:19 on My
Cluster
```

Cmd 21

```
1 data.collect().foreach(println)
```

▸ (1) Spark Jobs

```
r z h k p
z y x w v u t s
s x o n r
x z y m t s q e
z
x z y r q t p
```

```
Command took 0.43 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:42:35 on My
Cluster
```

Cmd 22

```
1 val transactions: RDD[Array[String]] = data.map(s => s.trim.split(' '))
```

```
transactions: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[1976] at map at <console>:51
```

Command took 0.16 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:43:01 on My Cluster

Cmd 23

```
1 val fpg = new FPGrowth()
2   .setMinSupport(0.2)
3   .setNumPartitions(10)
```

```
fpg: org.apache.spark.mllib.fpm.FPGrowth = org.apache.spark.mllib.fpm.FPGrowth@7607e43
```

Command took 0.08 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:43:12 on My Cluster

Cmd 24

```
1 val model = fpg.run(transactions)
```

▸ (2) Spark Jobs

```
model: org.apache.spark.mllib.fpm.FPGrowthModel[String] = org.apache.spark.mllib.fpm.FPGrowthModel@75bfe740
```

Command took 0.56 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:43:34 on My Cluster

Cmd 25

```
1 model.freqItemsets.collect().foreach { itemset =>
2   println(itemset.items.mkString("[", ",", "]") + ", " + itemset.freq)
3 }
```

▸ (1) Spark Jobs

```
[z], 5
[x], 4
[x,z], 3
[y], 3
[y,x], 3
[y,x,z], 3
[y,z], 3
[r], 3
[r,x], 2
[r,z], 2
[s], 3
[s,y], 2
[s,y,x], 2
[s,y,x,z], 2
[s,y,z], 2
[s,x], 3
[s,x,z], 2
[s,z], 2
[t], 3
```

```
[t,y], 3
[t,y,x], 3
```

Command took 0.43 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:43:47 on My Cluster

Cmd 26

```
1 val minConfidence = 0.8
2 model.generateAssociationRules(minConfidence).collect().foreach { rule =>
3   println(
4     rule.antecedent.mkString("[", ",", "]")
5       + " => " + rule.consequent .mkString("[", ",", "]")
6       + ", " + rule.confidence)
7 }
```

▸ (1) Spark Jobs

```
[q,t] => [x], 1.0
[q,t] => [z], 1.0
[q] => [y], 1.0
[q] => [t], 1.0
[q] => [x], 1.0
[q] => [z], 1.0
[t,s,z] => [y], 1.0
[t,s,z] => [x], 1.0
[t,x] => [y], 1.0
[t,x] => [z], 1.0
[s,z] => [y], 1.0
[s,z] => [x], 1.0
[s,z] => [t], 1.0
[s,y,x,z] => [t], 1.0
[s] => [x], 1.0
[t,s,y,z] => [x], 1.0

[s,y,z] => [x], 1.0
[s,y,z] => [t], 1.0
[q,t,x] => [y], 1.0
[q,t,x] => [z], 1.0
[r,z] => [p], 1.0
```

Command took 0.29 seconds -- by louhy1128@gmail.com at 2017/7/3 下午3:45:08 on My Cluster

Cmd 27