Zeppelin Notebook

Search your Notebooks

anonymous

spark-ml-randomforest-creditrisk

default

```
import org.apache.spark._
import org.apache.spark.rdd.RDD

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql._
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:08 PM.

```
case class Credit(
    creditability: Double,
    balance: Double, duration: Double, history: Double, purpose: Double, amount: Double,
    savings: Double, employment: Double, instPercent: Double, sexMarried: Double, guarantors: Double,
    residenceDuration: Double, assets: Double, age: Double, concCredit: Double, apartment: Double,
    credits: Double, occupation: Double, dependents: Double, hasPhone: Double, foreign: Double
  )

defined class Credit
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:08 PM.

```
def parseCredit(str: String): Credit = {
    val fields: Array[Double] = str.split(",").map(s => s.toDouble)
    Credit(
      fields(0),
      fields(1) - 1, fields(2), fields(3), fields(4), fields(5),
      fields(6) - 1, fields(7) - 1, fields(8), fields(9) - 1, fields(10) - 1,
      fields(11) - 1, fields(12) - 1, fields(13), fields(14) - 1, fields(15) - 1,
      fields(16) - 1, fields(17) - 1, fields(18) - 1, fields(19) - 1, fields(20) - 1
    )
  }
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:08 PM.

```
val creditText: RDD[String] = sc.textFile("/home/anindya/zeppelin-spark-projects/spark-ml-randomforest-creditrisk/data/germancredit.csv")
val creditDF: DataFrame = creditText.map(parseCredit).toDF().cache()
creditDF.registerTempTable("credit")
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:10 PM.

```
creditDF.show(10)
```

```
+-------------+-------+--------+-------+-------+-------+-------+-----------+-------+-------+-----------+-------+-------+-------+-------+-----------+-----------+-------+-------+
|creditability|balance|duration|history|purpose|amount|savings|employment|instPercent|sexMarried|guarantors|residenceDuration|assets| age|concCredit|apartment|credits|occupation|dependents|hasPhone|foreign|
+-------------+-------+--------+-------+-------+-------+-------+-----------+-------+-------+-----------+-------+-------+-------+-------+-----------+-----------+-------+-------+
|          1.0|    0.0|    18.0|    4.0|    2.0|1049.0|    0.0|        1.0|    4.0|    1.0|        0.0|        3.0|  1.0|21.0|    0.0|    0.0|    0.0|       2.0|       0.0|    0.0|    0.0|
|          1.0|    0.0|     9.0|    4.0|    0.0|2799.0|    0.0|        2.0|    2.0|    2.0|        0.0|        1.0|  0.0|36.0|    0.0|    0.0|    1.0|       2.0|       1.0|    0.0|    0.0|
|          1.0|    1.0|    12.0|    2.0|    9.0| 841.0|    1.0|        3.0|    2.0|    1.0|        0.0|        3.0|  0.0|23.0|    0.0|    0.0|    0.0|       1.0|       0.0|    0.0|    0.0|
|          1.0|    0.0|    12.0|    4.0|    0.0|2122.0|    0.0|        2.0|    3.0|    2.0|        0.0|        1.0|  0.0|39.0|    0.0|    0.0|    1.0|       1.0|       0.0|    1.0|    0.0|
|          1.0|    0.0|    12.0|    4.0|    0.0|2171.0|    0.0|        2.0|    4.0|    2.0|        0.0|        3.0|  1.0|30.0|    0.0|    1.0|    1.0|       1.0|       1.0|    0.0|    1.0|
|          1.0|    0.0|    10.0|    4.0|    0.0|2241.0|    0.0|        1.0|    1.0|    2.0|        0.0|        2.0|  0.0|48.0|    0.0|    0.0|    1.0|       1.0|       1.0|    0.0|    1.0|
|          1.0|    0.0|     8.0|    4.0|    0.0|1804.0|    0.0|        3.0|    1.0|    2.0|        0.0|        3.0|  0.0|39.0|    0.0|    1.0|    1.0|       1.0|       1.0|    0.0|    1.0|
|          1.0|    0.0|    12.0|    4.0|    0.0|1361.0|    0.0|        1.0|    2.0|    2.0|        0.0|        3.0|  0.0|40.0|    0.0|    1.0|    0.0|       1.0|       1.0|    0.0|    1.0|
|          1.0|    3.0|    18.0|    4.0|    3.0|3098.0|    0.0|        4.0|    1.0|    0.0|        0.0|        3.0|  0.0|65.0|    0.0|    1.0|    1.0|       0.0|       0.0|    0.0|    0.0|
|          1.0|    1.0|    24.0|    2.0|    3.0|3758.0|    2.0|        0.0|    1.0|    3.0|        0.0|        3.0|  0.0|23.0|    2.0|    0.0|    0.0|       0.0|       0.0|    0.0|    0.0|
+-------------+-------+--------+-------+-------+-------+-------+-----------+-------+-------+-----------+-------+-------+-------+-------+-----------+-----------+-------+-------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:10 PM.

```
creditDF.printSchema
```

```
root
 |-- history: double (nullable = false)
 |-- purpose: double (nullable = false)
 |-- amount: double (nullable = false)
 |-- savings: double (nullable = false)
 |-- employment: double (nullable = false)
 |-- instPercent: double (nullable = false)
 |-- sexMarried: double (nullable = false)
 |-- guarantors: double (nullable = false)
 |-- residenceDuration: double (nullable = false)
 |-- assets: double (nullable = false)
 |-- age: double (nullable = false)
 |-- concCredit: double (nullable = false)
 |-- apartment: double (nullable = false)
 |-- credits: double (nullable = false)
 |-- occupation: double (nullable = false)
 |-- dependents: double (nullable = false)
 |-- hasPhone: double (nullable = false)
 |-- foreign: double (nullable = false)
```

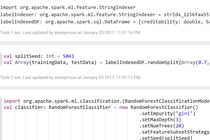
Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:11 PM.

```
sqlContext.sql("SELECT creditability, AVG(balance) as avgbalance, AVG(amount) as avgamt, AVG(duration) as avgdur FROM credit GROUP BY creditability ").show
```

```
+-------------+------------------+------------------+------------------+
|creditability|       avgbalance|          avgamt|          avgdur|
+-------------+------------------+------------------+------------------+
|          0.0|0.9833333333333333|3938.126666666666|           24.86|
|          1.0|1.0657142857142857|2905.4428571428571|19.207142857142856|
+-------------+------------------+------------------+------------------+
```

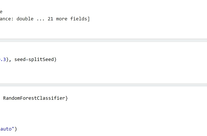
Took 0 sec. Last updated by anonymous at January 25 2017, 11:07:11 PM.

```
z.show(creditDF.groupBy("creditability").avg("balance"))
```



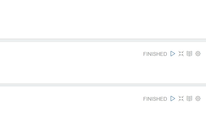
Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:12 PM.

```
z.show(creditDF.groupBy("creditability").avg("amount"))
```



Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:13 PM.

```
z.show(creditDF.groupBy("creditability").avg("duration"))
```



Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:14 PM.

```
creditDF.describe("balance").show
```

```
+-------+------------------+
|summary|           balance|
+-------+------------------+
|  count|              1000|
|   mean|             1.577|
| stddev|1.2576377271108931|
|    min|               0.0|
|    max|               3.0|
+-------+------------------+
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:14 PM.

```
val featureCols: Array[String] = Array("balance", "duration", "history", "purpose", "amount",
                                       "savings", "employment", "instPercent", "sexMarried", "guarantors",
                                       "residenceDuration", "assets", "age", "concCredit", "apartment",
                                       "credits", "occupation", "dependents", "hasPhone", "foreign")
val assembler: VectorAssembler = new VectorAssembler().setInputCols(featureCols).setOutputCol("features")
val assembleDF: DataFrame = assembler.transform(creditDF)

import org.apache.spark.ml.feature.StringIndexer
val labelIndexer: StringIndexer = new StringIndexer().setInputCol("creditability").setOutputCol("label")
val labelIndexerDF: DataFrame = labelIndexer.fit(assembleDF).transform(assembleDF)

import org.apache.spark.ml.feature.StringIndexer
labelIndexer: org.apache.spark.ml.feature.StringIndexer = strIdx_3156faa53c9e
labelIndexerDF: org.apache.spark.sql.DataFrame = [creditability: double, balance: double ... 23 more fields]
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:16 PM.

```
val splitSeed: Int = 5043
val Array(trainingData, testData) = labelIndexerDF.randomSplit(Array(0.7, 0.3), seed-splitSeed)
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:17 PM.

```
import org.apache.spark.ml.classification.{RandomForestClassificationModel, RandomForestClassifier}
val classifier: RandomForestClassifier = new RandomForestClassifier()
                                             .setInputCol("label")
                                             .setNumTrees(5)
                                             .setMaxDepth(5)
                                             .setMaxBins(20)
                                             .setFeatureSubsetStrategy("auto")
                                             .setSeed(splitSeed)
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:17 PM.

```
import org.apache.spark.ml.classification.RandomForestClassificationModel
val model: RandomForestClassificationModel = classifier.fit(trainingData)
model.toDebugString

import org.apache.spark.ml.classification.RandomForestClassificationModel
model: org.apache.spark.ml.classification.RandomForestClassificationModel = RandomForestClassificationModel (uid=rfc_22762cdbdd1b) with 20 trees
res215: String =
"RandomForestClassificationModel (uid=rfc_22762cdbdd1b) with 20 trees
  Tree 0 (weight 1.0):
    If (feature 0 <= 1.0)
     If (feature 10 <= 0.0)
      If (feature 3 <= 6.0)
       Predict: 0.0
      Else (feature 3 > 6.0)
       Predict: 0.0
     Else (feature 10 > 0.0)
      If (feature 12 <= 63.0)
       Predict: 0.0
      Else (feature 12 > 63.0)
       Predict: 0.0
    Else (feature 0 > 1.0)
     If (feature 13 <= 1.0)
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:18 PM.

```
val evaluator: BinaryClassificationEvaluator = new BinaryClassificationEvaluator().setMetricName("areaUnderROC").setLabelCol("label")
val predictions: DataFrame = model.transform(testData)
val accuracy: Double = evaluator.evaluate(predictions)
println("Accuracy Before Pipeline Fitting: " + accuracy)

evaluator: org.apache.spark.ml.evaluation.BinaryClassificationEvaluator = binEval_75ea7787fc5d
predictions: org.apache.spark.sql.DataFrame = [creditability: double, balance: double ... 24 more fields]
accuracy: Double = 0.7824900081835715
Accuracy Before Pipeline Fitting: 0.7824900081835715
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:20 PM.

```
predictions.sample(withReplacement=false,fraction=0.05).select("creditability", "label", "rawPrediction", "probability", "prediction").show(10)
```

```
+-------------+-----+--------------------+--------------------+----------+
|creditability|label|       rawPrediction|         probability|prediction|
+-------------+-----+--------------------+--------------------+----------+
|          1.0|  0.0|[13.9077885856573...|[0.6993894292820...|       0.0|
|          1.0|  0.0|[14.3026062478632...|[0.7451202123926...|       0.0|
|          1.0|  0.0|[12.5854981864153...|[0.6292745893076...|       0.0|
|          1.0|  0.0|[14.3739624784427...|[0.7186908737417E...|       0.0|
|          1.0|  0.0|[16.0445695564182...|[0.8022294778209...|       0.0|
|          1.0|  0.0|[13.6852987996856...|[0.7044669399325...|       0.0|
|          1.0|  0.0|[16.8142230261005...|[0.8407311513058...|       0.0|
|          1.0|  0.0|[16.8187640009022...|[0.8409138245091...|       0.0|
|          1.0|  0.0|[15.3712774137781...|[0.7685638708890...|       0.0|
|          1.0|  0.0|[13.8535930085255...|[0.6926760654267...|       0.0|
+-------------+-----+--------------------+--------------------+----------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:07:20 PM.

```
import org.apache.spark.mllib.evaluation.RegressionMetrics
val rm: RegressionMetrics = new RegressionMetrics(
    predictions.select("prediction", "label").rdd.map(x =>
      (x(0).asInstanceOf[Double], x(1).asInstanceOf[Double]))
)

println("MSE: " + rm.meanSquaredError)
println("MAE: " + rm.meanAbsoluteError)
println("RMSE Squared: " + rm.rootMeanSquaredError)
println("R Squared: " + rm.r2)
println("Explained Variance: " + rm.explainedVariance)

import org.apache.spark.mllib.evaluation.RegressionMetrics
rm: org.apache.spark.mllib.evaluation.RegressionMetrics = org.apache.spark.mllib.evaluation.RegressionMetrics@291ia10e
MSE: 0.31836788297689
MAE: 0.31836788297689
RMSE Squared: 0.5577062233749022
R Squared: -0.4116661590091386
Explained Variance: 0.1318665339381387
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:07:22 PM.

```
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.ml.{ Pipeline, PipelineStage }
import org.apache.spark.ml.tuning.{ ParamGridBuilder, CrossValidator }

val paramGrid: Array[ParamMap] = new ParamGridBuilder()
    .addGrid(classifier.maxBins, Array(25, 31))
    .addGrid(classifier.maxDepth, Array(5, 10))
    .addGrid(classifier.numTrees, Array(20, 60))
    .addGrid(classifier.impurity, Array("entropy", "gini"))
    .build()

val steps: Array[PipelineStage] = Array(classifier)
val pipeline: Pipeline = new Pipeline().setStages(steps)

val cv: CrossValidator = new CrossValidator()
                         .setEstimator(pipeline)
                         .setEvaluator(evaluator)
                         .setEstimatorParamMaps(paramGrid)
                         .setNumFolds(10)
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:09:23 PM.

```
import org.apache.spark.ml.tuning.CrossValidatorModel
val pipelineFittedModel: CrossValidatorModel = cv.fit(trainingData)

import org.apache.spark.ml.tuning.CrossValidatorModel
pipelineFittedModel: org.apache.spark.ml.tuning.CrossValidatorModel = cv_97412dc66a3e
```

Took 2 min 0 sec. Last updated by anonymous at January 25 2017, 11:09:22 PM.

```
val cvpreditions: DataFrame = pipelineFittedModel.transform(testData)
val cvaccuracy: Double = evaluator.evaluate(cvpreditions)
println("Accuracy After Pipeline Fitting: " + cvaccuracy)

cvpreditions: org.apache.spark.sql.DataFrame = [creditability: double, balance: double ... 24 more fields]
cvaccuracy: Double = 0.8089048511727079
Accuracy After Pipeline Fitting: 0.8089048511727079
```

Took 1 min 59 sec. Last updated by anonymous at January 25 2017, 11:09:23 PM.

```
val bestModel = pipelineFittedModel.bestModel.asInstanceOf[org.apache.spark.ml.PipelineModel].stages(0).asInstanceOf[RandomForestClassificationModel]
val pipelineFittedBestModel = pipelineFittedModel.asInstanceOf[org.apache.spark.ml.PipelineModel].stages(0)

bestModel: org.apache.spark.ml.classification.RandomForestClassificationModel = RandomForestClassificationModel (uid=rfc_09e099dd2477) with 60 trees
RandomForestClassificationModel (uid=rfc_09e099dd2477) with 60 trees
```

Took 1 min. Last updated by anonymous at January 25 2017, 11:09:23 PM.

```
bestModel.extractParamMap
```

```
{
    rfc_09e099dd2477-cacheNodeIds: false,
    rfc_09e099dd2477-checkpointInterval: 10,
    rfc_09e099dd2477-featureSubsetStrategy: auto,
    rfc_09e099dd2477-featuresCol: features,
    rfc_09e099dd2477-impurity: gini,
    rfc_09e099dd2477-labelCol: label,
    rfc_09e099dd2477-maxBins: 25,
    rfc_09e099dd2477-maxDepth: 10,
    rfc_09e099dd2477-maxMemoryInMB: 256,
    rfc_09e099dd2477-minInfoGain: 0.0,
    rfc_09e099dd2477-minInstancesPerNode: 1,
    rfc_09e099dd2477-predictionCol: prediction,
    rfc_09e099dd2477-probabilityCol: probability,
    rfc_09e099dd2477-rawPredictionCol: rawPrediction,
    rfc_09e099dd2477-seed: 5043,
    rfc_09e099dd2477-subsamplingRate: 1.0
}
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:09:24 PM.

```
val rmcv: RegressionMetrics = new RegressionMetrics(cvpreditions.select("prediction", "label")
                                        .rdd.map(x => (x(0).asInstanceOf[Double], x(1).asInstanceOf[Double])))

println("MSE: " + rmcv.meanSquaredError)
println("MAE: " + rmcv.meanAbsoluteError)
println("RMSE Squared: " + rmcv.rootMeanSquaredError)
println("R Squared: " + rmcv.r2)
println("Explained Variance: " + rmcv.explainedVariance)

rmcv: org.apache.spark.mllib.evaluation.RegressionMetrics = org.apache.spark.mllib.evaluation.RegressionMetrics@ee708a
MSE: 0.26755852842809363
MAE: 0.26755852842809363
RMSE Squared: 0.5172606803118717
R Squared: -0.2143364809601801E12
Explained Variance: 0.15129619234994053
```

Took 2 sec. Last updated by anonymous at January 25 2017, 11:09:25 PM.

```
bestModel.write.overwrite().save("/home/anindya/zeppelin-spark-projects/spark-ml-randomforest-creditrisk/model")
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:09:26 PM.

```
|
```

Took 1 sec. Last updated by anonymous at January 25 2017, 11:09:26 PM.