spark-ml-randomforest-creditrisk

```
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 2 sec. Last updated by anonymous at January 24 2017, 11:39:31 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
    )
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:39:44 PM.

```
def parseCredit(str: String): Credit = {
    val fields: Array[Double] = str.split(",").map(x => x.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 24 2017, 11:39:50 PM.

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

Took 2 sec. Last updated by anonymous at January 24 2017, 11:41:03 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|    6.0|    4.0|    4.0|2.0|1049.0|    0.0|    1.0|    4.0|    1.0|    0.0|    3.0| 1.0|21.0|    0.0|    2.0|    0.0|    0.0|    2.0|    0.0|    0.0|
|          1.0|    1.0|   48.0|    2.0|    4.0|0.0|2795.0|    0.0|    2.0|    2.0|    2.0|    0.0|    1.0|0.0|36.0|    2.0|    0.0|    1.0|    0.0|    1.0|    0.0|    0.0|
|          1.0|    3.0|   12.0|    2.0|    9.0|0.0|841.0|    1.0|    3.0|    2.0|    1.0|    0.0|    3.0|0.0|23.0|    2.0|    0.0|    1.0|    0.0|    0.0|    0.0|    0.0|
|          1.0|    0.0|   42.0|    4.0|    2.0|0.0|8122.0|    0.0|    2.0|    3.0|    2.0|    0.0|    3.0|0.0|39.0|    2.0|    0.0|    1.0|    1.0|    1.0|    0.0|    1.0|
|          1.0|    0.0|   12.0|    4.0|    0.0|0.0|1275.0|    0.0|    2.0|    4.0|    2.0|    0.0|    3.0|1.0|38.0|    0.0|    0.0|    1.0|    0.0|    1.0|    0.0|    1.0|
|          1.0|    0.0|   60.0|    4.0|    3.0|0.0|8229.0|    0.0|    1.0|    2.0|    2.0|    0.0|    3.0|0.0|65.0|    2.0|    0.0|    1.0|    0.0|    1.0|    0.0|    1.0|
|          1.0|    0.0|    8.0|    4.0|    0.0|0.0|1908.0|    0.0|    1.0|    2.0|    2.0|    0.0|    3.0|0.0|30.0|    2.0|    1.0|    1.0|    0.0|    0.0|    1.0|    1.0|
|          1.0|    0.0|   18.0|    4.0|    3.0|0.0|3059.0|    0.0|    2.0|    4.0|    2.0|    0.0|    3.0|0.0|23.0|    2.0|    1.0|    1.0|    0.0|    0.0|    0.0|    1.0|
|          1.0|    1.0|   36.0|    4.0|    2.0|0.0|6948.0|    0.0|    2.0|    4.0|    2.0|    0.0|    3.0|0.0|35.0|    2.0|    1.0|    1.0|    0.0|    0.0|    0.0|    1.0|
+-------------+-------+--------+-------+-------+-------+-------+-----------+-----------+-----------+-------+-----------+-----+-----------+-----------+-----------+-----------+-----------+-----------+-------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:21 PM.

```
creditDF.printSchema
```

```
|-- 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)
```

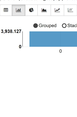
Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:23 PM.

```
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|0.9033333333333333|3938.126666666666|           24.86|
|          1.0|1.8637410957142857| 2985.442857142857|19.207142857142856|
+-------------+------------------+------------------+------------------+
```

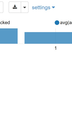
Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:35 PM.

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



Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:40 PM.

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



Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:45 PM.

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

Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:49 PM.

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

```
+-------+------------+
|summary|     balance|
+-------+------------+
|  count|        1000|
|   mean|       1.577|
| stddev|1.2576377271809931|
|    min|         0.0|
|    max|         3.0|
+-------+------------+
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:41:55 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)
z.show(assembleDF.sample(withReplacement=false,fraction=0.01))
```

creditability	balance	duration	history	purpose	amount	savings	employment	instPercent	sexMarried	guarantors	residenceDuration	assets	age	concCredit	apartment	credits	occupation	dependents	hasPh
1	3	18	4	3	1,098	0	0	4	1	0	3	2	65	2	1	1	0	0	0
1	2	30	2	3	3,017	0	4	4	2	0	3	1	47	2	1	0	2	0	0
1	3	21	2	3	3,160	4	4	4	2	0	2	1	41	2	1	0	2	0	1
1	3	54	0	1	9,436	4	2	2	2	0	1	1	39	2	1	0	1	1	0
1	3	6	4	3	1,554	0	3	1	2	0	1	2	24	2	0	1	2	0	1
1	3	18	4	2	3,780	0	1	3	0	0	0	1	2	35	2	1	3	0	0
1	0	36	2	2	3,620	0	2	1	2	2	1	1	37	2	1	0	2	1	0
1	1	8	2	3	760	0	3	4	1	2	1	0	44	2	1	0	1	0	0

Took 1 sec. Last updated by anonymous at January 24 2017, 11:45:00 PM.

```
import org.apache.spark.ml.feature.StringIndexer
val labelIndexer: StringIndexer = new StringIndexer().setInputCol("creditability").setOutputCol("label")
val labelIndexedDF: DataFrame = labelIndexer.fit(df).transform(assembleDF)
z.show(labelIndexedDF.sample(withReplacement=false,fraction=0.01))
```

creditability	balance	duration	history	purpose	amount	savings	employment	instPercent	sexMarried	guarantors	residenceDuration	assets	age	concCredit	apartment	credits	occupation	dependents	hasPh
1	0	9	4	0	2,799	0	2	2	2	0	1	0	36	2	0	1	2	1	0
1	3	30	4	3	5,771	0	3	4	1	0	0	2	25	2	1	1	2	0	0
1	1	36	2	6	3,711	4	2	2	3	0	1	2	27	2	1	0	2	0	0
1	3	6	4	0	250	3	2	2	1	0	1	0	41	0	1	1	0	1	0
1	3	12	0	2	2,759	0	4	2	2	0	3	1	34	2	1	1	2	0	0
1	3	21	3	2	1,591	1	3	4	2	0	2	0	34	2	1	1	3	0	0
1	0	12	2	6	1,200	4	2	4	1	0	3	1	23	0	0	0	2	0	1
1	3	24	3	9	3,883	0	2	1	1	0	1	3	32	2	2	0	2	0	0

Took 1 sec. Last updated by anonymous at January 24 2017, 11:50:14 PM.

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

Took 1 sec. Last updated by anonymous at January 24 2017, 11:50:57 PM.

```
val classifier: RandomForestClassifier = new RandomForestClassifier()
                                        .setImpurity("gini")
                                        .setMaxDepth(3)
                                        .setNumTrees(20)
                                        .setFeatureSubsetStrategy("auto")
                                        .setSeed(splitSeed)

classifier: org.apache.spark.ml.classification.RandomForestClassifier = rfc_55ace430f7d
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:51:50 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_00f60351373f) with 20 trees
res120: String =
"RandomForestClassificationModel (uid=rfc_00f60351373f) with 20 trees
  Tree 0 (weight 1.0):
    If (feature 10 <= 1.0)
     If (feature 10 <= 0.0)
      Predict: 0.0
     Else (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 24 2017, 11:53:43 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_207005b0b066
predictions: org.apache.spark.sql.DataFrame = [creditability: double, balance: double ... 24 more fields]
accuracy: Double = 0.7826900058193716
Accuracy Before Pipeline Fitting: 0.7826900058193716
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:56:23 PM.

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

creditability	label	rawPrediction	probability	prediction
0	1	[12.4561659319822,7.541834060817796]	[0.6229082965991101,0.3770917034009]	0
0	1	[14.29889647815461,5.701103521845379]	[0.7149448829907731,0.2850551760692269]	0
0	1	[11.57053289969721,8.429467110030279]	[0.5785269446984800[5,0.4214730550115955]	0
0	1	[14.25966991572607,5.740030089275093]	[0.7129984955302454,0.2870015044697546]	0
0	1	[11.58046283035153,8.419537319684869]	[0.5790231340157567,0.4209768659842434]	0
0	1	[8.274200872704299,11.7257991272957]	[0.4137100436352149,0.5862899563647851]	1
0	1	[14.37899219557198[7,5.621007804428015]	[0.7189496097785993,0.2810503902214007]	0
1	0	[13.87360077153609,6.126393226468365]	[0.6931903857658[0,0.3068096142341917]	0

Took 1 sec. Last updated by anonymous at January 24 2017, 11:58:54 PM.

```
val rm: RegressionMetrics = new RegressionMetrics(
    predictions.select("prediction", "label").rdd.map(x =>
        (x(0).asInstanceOf[Double], x(1).asInstanceOf[Double]))
print("MSE: " + rm.meanSquaredError)
print("MAE: " + rm.meanAbsoluteError)
print("RMSE Squared: " + rm.rootMeanSquaredError)
print("R Squared: " + rm.r2)
print("Explained Variance: " + rm.explainedVariance)
```

```
rm: org.apache.spark.mllib.evaluation.RegressionMetrics = org.apache.spark.mllib.evaluation.RegressionMetrics@1bfdeeb
MSE: 0.3128267892976589
MAE: 0.3128267892976589
RMSE Squared: 0.5577867237748022
R Squared: -0.6166613980091386
Explained Variance: 0.1331886339313387
```

Took 1 sec. Last updated by anonymous at January 24 2017, 11:59:19 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.numBins, Array(25, 31))
    .addGrid(classifier.maxDepth, Array(5, 10))
    .addGrid(classifier.numTrees, Array(20, 30))
    .addGrid(classifier.impurity, Array("entropy", "gini"))
    .build()

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

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

```
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_d114fc51f7f
```

Took 1hr 15 min. Last updated by anonymous at January 25 2017, 12:02:22 AM.

```
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 = 0.8040040011727075
Accuracy After Pipeline Fitting: 0.8040040011727075
```

Took 1 sec. Last updated by anonymous at January 25 2017, 12:02:56 AM.

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

bestModel: org.apache.spark.ml.classification.RandomForestClassificationModel = RandomForestClassificationModel (uid=rfc_51fd4c3b2b31) with 60 trees
RandomForestClassificationModel (uid=rfc_51fd4c3b2b31) with 60 trees
```

Took 1 sec. Last updated by anonymous at January 25 2017, 12:03:36 AM.

```
bestModel.extractParamMap
```

```
res216: org.apache.spark.ml.param.ParamMap =
{
        rfc_51fd4c3b2b31-cacheNodeIds: false,
        rfc_51fd4c3b2b31-checkpointInterval: 10,
        rfc_51fd4c3b2b31-featureSubsetStrategy: auto,
        rfc_51fd4c3b2b31-featuresCol: features,
        rfc_51fd4c3b2b31-impurity: gini,
        rfc_51fd4c3b2b31-labelCol: label,
        rfc_51fd4c3b2b31-maxBins: 25,
        rfc_51fd4c3b2b31-maxDepth: 10,
        rfc_51fd4c3b2b31-maxMemoryInMB: 256,
        rfc_51fd4c3b2b31-minInfoGain: 0.0,
        rfc_51fd4c3b2b31-minInstancesPerNode: 1,
        rfc_51fd4c3b2b31-predictionCol: prediction,
        rfc_51fd4c3b2b31-probabilityCol: probability,
        rfc_51fd4c3b2b31-rawPredictionCol: rawPrediction,
        rfc_51fd4c3b2b31-seed: 5043,
        rfc_51fd4c3b2b31-subsamplingRate: 1.0
}
```

Took 1 sec. Last updated by anonymous at January 25 2017, 12:05:32 AM.

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

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

rmcv: org.apache.spark.mllib.evaluation.RegressionMetrics = org.apache.spark.mllib.evaluation.RegressionMetrics@10dfecee
MSE: 0.2675385284280593
MAE: 0.2675385284280593
RMSE Squared: 0.5172360401118717
R Squared: -0.2161568088128532
Explained Variance: 0.1511206234494053
```

Took 1 sec. Last updated by anonymous at January 25 2017, 12:04:45 AM.

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

Took 2 sec. Last updated by anonymous at January 25 2017, 12:13:16 AM.

READY