```
sc
```
FINISHED

```
res24: org.apache.spark.SparkContext = org.apache.spark.SparkContext@383d786c
res24: org.apache.spark.SparkContext = org.apache.spark.SparkContext@383d786c
```

```
spark
```
FINISHED

```
res26: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@441c4458
res26: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@441c4458
```

```
val rdd = sc.textFile("s3://fda-proteins")
```
FINISHED

```
rdd: org.apache.spark.rdd.RDD[String] = s3://fda-proteins MapPartitionsRDD[21] at textFile
at <console>:44
rdd: org.apache.spark.rdd.RDD[String] = s3://fda-proteins MapPartitionsRDD[21] at textFile
at <console>:44
```

```
val gender = spark.read.
    option("header", "true").
    option("InferSchema", "true").
    csv("s3://fda-proteins/gender.csv")

val msi = spark.read.
    option("header", "true").
    option("InferSchema", "true").
    csv("s3://fda-proteins/msi.csv")
```
READY

```
gender: org.apache.spark.sql.DataFrame = [_c0: int, A1BG: double ... 4117 more fields]
msi: org.apache.spark.sql.DataFrame = [_c0: int, A1BG: double ... 4117 more fields]
gender: org.apache.spark.sql.DataFrame = [_c0: int, A1BG: double ... 4117 more fields]
msi: org.apache.spark.sql.DataFrame = [_c0: int, A1BG: double ... 4117 more fields]
```

```
val Array(trainData, testData) = gender.randomSplit(Array(0.8, 0.2))
trainData.cache()
testData.cache()
```
READY

```
trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG: double
... 4117 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG: double
... 4117 more fields]
res122: trainData.type = [_c0: int, A1BG: double ... 4117 more fields]
res123: testData.type = [_c0: int, A1BG: double ... 4117 more fields]
trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG: double
... 4117 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG: double
... 4117 more fields]
res122: trainData.type = [_c0: int, A1BG: double ... 4117 more fields]
res123: testData.type = [_c0: int, A1BG: double ... 4117 more fields]
```

READY

```
import org.apache.spark.ml.feature.VectorAssembler

val inputCols = trainData.columns.filter(_ != "gender")

val assembler = new VectorAssembler().
  setInputCols(inputCols).
  setOutputCol("featureVector")

val assembledTrainData = assembler.transform(trainData)
assembledTrainData.select("featureVector").show(truncate = false)
```

```
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
```

Output is truncated to 102400 bytes. Learn more about **ZEPPELIN_INTERPRETER_OUTPUT_LIMIT**

```
import org.apache.spark.ml.classification.DecisionTreeClassifier
import scala.util.Random

val classifier = new DecisionTreeClassifier().
  setSeed(Random.nextLong()).
  setLabelCol("gender").
  setFeaturesCol("featureVector").
  setPredictionCol("prediction")

val model = classifier.fit(assembledTrainData)
println(model.toDebugString)
```

READY

```
import scala.util.Random
classifier: org.apache.spark.ml.classification.DecisionTreeClassifier = dtc_674734a7e087
model: org.apache.spark.ml.classification.DecisionTreeClassificationModel = DecisionTreeCla
ssificationModel (uid=dtc_674734a7e087) of depth 3 with 9 nodes
DecisionTreeClassificationModel (uid=dtc_674734a7e087) of depth 3 with 9 nodes
  If (feature 824 <= 2.654789224848015)
   If (feature 3866 <= 6.22272802278904)
    If (feature 3 <= 1.030940732348935)
     Predict: 1.0
    Else (feature 3 > 1.030940732348935)
     Predict: 0.0
   Else (feature 3866 > 6.22272802278904)
    If (feature 81 <= 1.274504593012105)
     Predict: 0.0
    Else (feature 81 > 1.274504593012105)
     Predict: 1.0
  Else (feature 824 > 2.654789224848015)
   Predict: 0.0
```

```
model.featureImportances.toArray.zip(inputCols).
  sorted.reverse.foreach(println)
```

READY

```
(0.38431453976939645,UBA1)
(0.3720930232558137,CUL4A)
(0.1663865546218487,ACTR1B)
(0.07720588235294118,AAAS)
(0.0,_c0)
(0.0,ZZEF1)
(0.0,ZYX)
(0.0,ZW10)
(0.0,ZPR1)
(0.0,ZNF706)
(0.0,ZNF638)
(0.0,ZNF326)
(0.0,ZNF280C)
(0.0,ZNF207)
(0.0,ZNF185)
(0.0,ZMYM3)
(0.0,ZMPSTE24)
(0.0,ZG16)
```

```
val predictions = model.transform(assembledTrainData)
predictions.select("gender", "prediction", "probability").
  show(truncate = false)
```

READY

```
|1     |1.0       |[0.0,1.0]  |
|0     |0.0       |[1.0,0.0]  |
|1     |1.0       |[0.0,1.0]  |
|1     |1.0       |[0.0,1.0]  |
|1     |1.0       |[0.0,1.0]  |
|0     |0.0       |[1.0,0.0]  |
|1     |1.0       |[0.0,1.0]  |
|1     |1.0       |[0.0,1.0]  |
|1     |1.0       |[0.0,1.0]  |
|0     |0.0       |[1.0,0.0]  |
|1     |1.0       |[0.0,1.0]  |
+------+----------+-----------+
only showing top 20 rows

predictions: org.apache.spark.sql.DataFrame = [_c0: int, A1BG: double ... 4121 more fields]
+------+----------+-----------+
|gender|prediction|probability|
```

```
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

val evaluator = new MulticlassClassificationEvaluator().
  setLabelCol("gender").
  setPredictionCol("prediction")
```

READY

```
evaluator.setMetricName("accuracy").evaluate(predictions)
evaluator.setMetricName("f1").evaluate(predictions)
```

```
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = mcEval_767b78
42f63a
res133: Double = 1.0
res134: Double = 1.0
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = mcEval_767b78
42f63a
res133: Double = 1.0
res134: Double = 1.0
```

```
import org.apache.spark.mllib.evaluation.MulticlassMetrics

val predictionRDD = predictions.
  select("prediction", "gender").
  as[(Double,Double)].
  rdd
val multiclassMetrics = new MulticlassMetrics(predictionRDD)
multiclassMetrics.confusionMatrix
```

READY

```
import org.apache.spark.mllib.evaluation.MulticlassMetrics
predictionRDD: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsRDD[388] at rdd at
<console>:56
multiclassMetrics: org.apache.spark.mllib.evaluation.MulticlassMetrics = org.apache.spark.m
llib.evaluation.MulticlassMetrics@256f350b
res136: org.apache.spark.mllib.linalg.Matrix =
17.0  0.0
0.0   34.0
import org.apache.spark.mllib.evaluation.MulticlassMetrics
predictionRDD: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsRDD[388] at rdd at
<console>:56
multiclassMetrics: org.apache.spark.mllib.evaluation.MulticlassMetrics = org.apache.spark.m
llib.evaluation.MulticlassMetrics@256f350b
res136: org.apache.spark.mllib.linalg.Matrix =
17.0  0.0
0.0   34.0
```

```
val confusionMatrix = predictions.
  groupBy("gender").
  pivot("prediction", (1 to 2)).
  count().
```

READY

```
  na.fill(0.0).
  orderBy("gender")

confusionMatrix.show()
```

```
confusionMatrix: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [gender: int, 1:
bigint ... 1 more field]
+------+---+---+
|gender|  1|  2|
+------+---+---+
|     0|  0|  0|
|     1| 34|  0|
+------+---+---+

confusionMatrix: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [gender: int, 1:
bigint ... 1 more field]
+------+---+---+
|gender|  1|  2|
+------+---+---+
|     0|  0|  0|
|     1| 34|  0|
+------+---+---+
```

READY

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.RandomForestClassifier

val inputCols = trainData.columns.filter(_ != "gender")

val assembler = new VectorAssembler().
  setInputCols(inputCols).
  setOutputCol("featureVector")

val classifier = new RandomForestClassifier().
  setSeed(Random.nextLong()).
  setLabelCol("gender").
  setFeaturesCol("featureVector").
  setPredictionCol("prediction")

val pipeline = new Pipeline().setStages(Array(assembler, classifier))
```

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.RandomForestClassifier
inputCols: Array[String] = Array(_c0, A1BG, A2M, AAAS, AACS, AAGAB, AAK1, AAMDC, AARS, AARS
2, AASDHPPT, AATF, ABAT, ABCB7, ABCC1, ABCC3, ABCD1, ABCD3, ABCE1, ABCF1, ABCF2, ABCF3, ABH
D10, ABHD11, ABHD12, ABHD14B, ABHD16A, ABI1, ABLIM1, ABR, ABRACL, ACAA1, ACAA2, ACACA, ACAD
10, ACAD8, ACAD9, ACADM, ACADS, ACADSB, ACADVL, ACAP2, ACAT1, ACAT2, ACBD3, ACBD5, ACE, ACE
2, ACIN1, ACLY, ACO1, ACO2, ACOT1, ACOT11, ACOT13, ACOT7, ACOT8, ACOT9, ACOX1, ACOX3, ACP1,
ACP2, ACSF2, ACSF3, ACSL1, ACSL3, ACSL4, ACSL5, ACSS1, ACSS2, ACTA2, ACTB, ACTBL2, ACTG1, A
CTG2, ACTL6A, ACTN1, ACTN2, ACTN4, ACTR10, ACTR1A, ACTR1B, ACTR2, ACTR3, ACY1, ACYP1, ADAM1
0, ADAR, ADD1, ADD3, ADGRE5, ADH1B, ADH1C, ADH5, ADK, ADNP, ADPGK, ADPRHL2, ADRM1, ADSL, AD
SS, AEBP1, AFAP1, AFDN, AFG3L2, AFM, AGFG1, AGK, AGL, AGMAT, AGO2, A...assembler: org.apach
e.spark.ml.feature.VectorAssembler = vecAssembler_360d6c1eadba
classifier: org.apache.spark.ml.classification.RandomForestClassifier = rfc_6cb159416dee
pipeline: org.apache.spark.ml.Pipeline = pipeline_c0b2fa793fc7
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.RandomForestClassifier
inputCols: Array[String] = Array(_c0, A1BG, A2M, AAAS, AACS, AAGAB, AAK1, AAMDC, AARS, AARS
2, AASDHPPT, AATF, ABAT, ABCB7, ABCC1, ABCC3, ABCD1, ABCD3, ABCE1, ABCF1, ABCF2, ABCF3, ABH
```

READY

```
import org.apache.spark.ml.tuning.ParamGridBuilder

val paramGrid = new ParamGridBuilder().
  addGrid(classifier.impurity, Seq("gini", "entropy")).
  addGrid(classifier.maxDepth, Seq(1, 20)).
  addGrid(classifier.maxBins, Seq(40, 300)).
  addGrid(classifier.minInfoGain, Seq(0.0, 0.05)).
  build()

val multiclassEval = new MulticlassClassificationEvaluator().
  setLabelCol("gender").
  setPredictionCol("prediction").
  setMetricName("accuracy")
```

```
import org.apache.spark.ml.tuning.ParamGridBuilder
paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
        rfc_6cb159416dee-impurity: gini,
        rfc_6cb159416dee-maxBins: 40,
        rfc_6cb159416dee-maxDepth: 1,
        rfc_6cb159416dee-minInfoGain: 0.0
}, {
        rfc_6cb159416dee-impurity: gini,
        rfc_6cb159416dee-maxBins: 300,
        rfc_6cb159416dee-maxDepth: 1,
        rfc_6cb159416dee-minInfoGain: 0.0
}, {
        rfc_6cb159416dee-impurity: entropy,
        rfc_6cb159416dee-maxBins: 40,
        rfc_6cb159416dee-maxDepth: 1,
        rfc_6cb159416dee-minInfoGain: 0.0
}, {
```

Proteomics

```
import org.apache.spark.ml.tuning.TrainValidationSplit

val validator = new TrainValidationSplit().
  setSeed(Random.nextLong()).
  setEstimator(pipeline).
  setEvaluator(multiclassEval).
  setEstimatorParamMaps(paramGrid).
  setTrainRatio(0.9)

val validatorModel = validator.fit(trainData)
```

READY

```
import org.apache.spark.ml.tuning.TrainValidationSplit
validator: org.apache.spark.ml.tuning.TrainValidationSplit = tvs_03982fa18f10
validatorModel: org.apache.spark.ml.tuning.TrainValidationSplitModel = tvs_03982fa18f10
import org.apache.spark.ml.tuning.TrainValidationSplit
validator: org.apache.spark.ml.tuning.TrainValidationSplit = tvs_03982fa18f10
validatorModel: org.apache.spark.ml.tuning.TrainValidationSplitModel = tvs_03982fa18f10
```

```
import org.apache.spark.ml.PipelineModel

val bestModel = validatorModel.bestModel
bestModel.asInstanceOf[PipelineModel].stages.last.extractParamMap
```

READY

```
        rfc_6cb159416dee-cacheNodeIds: false,
        rfc_6cb159416dee-checkpointInterval: 10,
        rfc_6cb159416dee-featureSubsetStrategy: auto,
        rfc_6cb159416dee-featuresCol: featureVector,
        rfc_6cb159416dee-impurity: gini,
        rfc_6cb159416dee-labelCol: gender,
        rfc_6cb159416dee-maxBins: 40,
        rfc_6cb159416dee-maxDepth: 1,
        rfc_6cb159416dee-maxMemoryInMB: 256,
        rfc_6cb159416dee-minInfoGain: 0.0,
        rfc_6cb159416dee-minInstancesPerNode: 1,
        rfc_6cb159416dee-numTrees: 20,
        rfc_6cb159416dee-predictionCol: prediction,
        rfc_6cb159416dee-probabilityCol: probability,
        rfc_6cb159416dee-rawPredictionCol: rawPrediction,
        rfc_6cb159416dee-seed: 2817032373697688064,
        rfc_6cb159416dee-subsamplingRate: 1.0
}
```

```
val validatorModel = validator.fit(trainData)

val paramsAndMetrics = validatorModel.validationMetrics.
  zip(validatorModel.getEstimatorParamMaps).sortBy(-_._1)

paramsAndMetrics.foreach { case (metric, params) =>
    println(metric)
    println(params)
    println()
}
```

READY

```
import org.apache.spark.ml.PipelineModel
import org.apache.spark.ml.classification.RandomForestClassificationModel

val forestModel = bestModel.asInstanceOf[PipelineModel].
  stages.last.asInstanceOf[RandomForestClassificationModel]

forestModel.featureImportances.toArray.zip(inputCols).
  sorted.reverse.foreach(println)
```

READY

```
import org.apache.spark.ml.PipelineModel
import org.apache.spark.ml.classification.RandomForestClassificationModel
forestModel: org.apache.spark.ml.classification.RandomForestClassificationModel = RandomFor
estClassificationModel (uid=rfc_6cb159416dee) with 20 trees
(0.05,RNF20)
(0.05,PSMG1)
(0.05,PRTN3)
(0.05,PADI4)
(0.05,MRPL9)
(0.05,METTL1)
(0.05,MAP2K6)
(0.05,LYPLAL1)
(0.05,IRAK1)
(0.05,IFI35)
(0.05,HYOU1)
(0.05,HSPA2)
(0.05,HMGB2)
(0.05,GGH)
```

READY

READY

READY

READY

READY

READY

READY