```
sc
spark
```

FINISHED

...Random Forest Fil

```
res4: org.apache.spark.SparkContext = org.apache.spark.SparkContext@383d786c
res5: 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[1] at textFile a
t <console>:25
```

View in Spark web UI

SPARK JOBS FINISHED

```
val gender_correct = spark.read.
    option("header", "true").
    option("InferSchema", "true").
    option("maxColumns", 21600).
    csv("s3://fda-proteins/gender-2.csv")

val msi_correct = spark.read.
    option("header", "true").
    option("InferSchema", "true").
    option("maxColumns", 21600).
    csv("s3://fda-proteins/msi-2.csv")
```

```
gender_correct: org.apache.spark.sql.DataFrame = [_c0: int, A1BG1: double ... 21565 more fi
elds]
msi_correct: org.apache.spark.sql.DataFrame = [_c0: int, A1BG1: double ... 21565 more field
s]
```

```
val Array(gender_trainData, gender_testData) = gender_correct.randomSplit(Array(0.7, 0.3
    ))
```

FINISHED

```
gender_trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG1
: double ... 21565 more fields]
gender_testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG1:
double ... 21565 more fields]
msi_trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG1: d
ouble ... 21565 more fields]
msi_testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_c0: int, A1BG1: do
uble ... 21565 more fields]
```

```
gender_trainData.cache()
gender_testData.cache()
msi_trainData.cache()
msi_testData.cache()
```

READY

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.classification.RandomForestClassifier
import scala.util.Random

val inputCols = gender_trainData.columns.filter(_ != "gender") // this is the same for
    msi

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

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

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

val gender_pipeline = new Pipeline().setStages(Array(assembler, gender_classifier))
val msi_pipeline = new Pipeline().setStages(Array(assembler, msi_classifier))
```

FINISHED

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.classification.RandomForestClassifier
import scala.util.Random
inputCols: Array[String] = Array(_c0, A1BG1, A2M2, AAAS3, AACS4, AAGAB5, AAK16, AAMDC7, AAR
S8, AARS29, AASDHPPT10, AATF11, ABAT12, ABCB713, ABCC114, ABCC315, ABCD116, ABCD317, ABCE11
8, ABCF119, ABCF220, ABCF321, ABHD1022, ABHD1123, ABHD1224, ABHD14B25, ABHD16A26, ABI127, A
BLIM128, ABR29, ABRACL30, ACAA131, ACAA232, ACACA33, ACAD1034, ACAD835, ACAD936, ACADM37, A
CADS38, ACADSB39, ACADVL40, ACAP241, ACAT142, ACAT243, ACBD344, ACBD545, ACE46, ACE247, ACI
N148, ACLY49, ACO150, ACO251, ACOT152, ACOT1153, ACOT1354, ACOT755, ACOT856, ACOT957, ACOX1
58, ACOX359, ACP160, ACP261, ACSF262, ACSF363, ACSL164, ACSL365, ACSL466, ACSL567, ACSS168,
ACSS269, ACTA270, ACTB71, ACTBL2, ACTG173, ACTG274, ACTL6A75, ACTN176, ACTN277, ACTN478, AC
TR1079, ACTR1A80, ACTR1B81, ACTR282, ACTR383, ACY184, ACYP185, ADAM1...assembler: org.apach
e.spark.ml.feature.VectorAssembler = vecAssembler_deb8ab765f3d
gender_classifier: org.apache.spark.ml.classification.RandomForestClassifier = rfc_a6ab7afc
2c25
msi_classifier: org.apache.spark.ml.classification.RandomForestClassifier = rfc_2dd0350920b
b
```

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

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

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

FINISHED

```
        rfc_a6ab7afc2c25-minInfoGain: 0.05
}, {
        rfc_a6ab7afc2c25-impurity: gini,
        rfc_a6ab7afc2c25-maxBins: 40,
        rfc_a6ab7afc2c25-maxDepth: 20,
        rfc_a6ab7afc2c25-minInfoGain: 0.0
}, {
        rfc_a6ab7afc2c25-impurity: gini,
        rfc_a6a...msi_paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
        rfc_2dd0350920bb-impurity: gini,
        rfc_2dd0350920bb-maxBins: 40,
        rfc_2dd0350920bb-maxDepth: 1,
        rfc_2dd0350920bb-minInfoGain: 0.0
}, {
        rfc_2dd0350920bb-impurity: gini,
        rfc_2dd0350920bb-maxBins: 300,
        rfc_2dd0350920bb-maxDepth: 1,
```

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

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

val msi_eval = new MulticlassClassificationEvaluator().
  setLabelCol("msi").
  setPredictionCol("prediction").
```

FINISHED

```
  setMetricName("accuracy")
```

```
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
gender_eval: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = mcEval_3b40
1a681aed
msi_eval: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = mcEval_dcbd0e4
78259
```

FINISHED

```
import org.apache.spark.sql.types.{DoubleType, StringType};

gender_correct.schema.filter(_.dataType == StringType).foldLeft(gender_correct) {
    case (acc, col) => acc.withColumn(col.name, gender_correct(col.name).cast(DoubleType
        ))
}
```

```
import org.apache.spark.sql.types.{DoubleType, StringType}
res20: org.apache.spark.sql.DataFrame = [_c0: int, A1BG1: double ... 21565 more fields]
```

READY

ERROR

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

val gender_validator = new TrainValidationSplit().
  setSeed(Random.nextLong()).
  setEstimator(gender_pipeline).
  setEstimatorParamMaps(gender_paramGrid).
  setTrainRatio(0.9)

val gender_validatorModel = gender_validator.fit(gender_trainData)
```

```
Data type StringType of column CT47A5 is not supported.
Data type StringType of column CT47A6 is not supported.
Data type StringType of column CT47A7 is not supported.
Data type StringType of column CT47A8 is not supported.
Data type StringType of column CT47A9 is not supported.
Data type StringType of column DAZ1 is not supported.
Data type StringType of column DAZ3 is not supported.
Data type StringType of column F8A2 is not supported.
Data type StringType of column OPN1MW is not supported.
Data type StringType of column OPN1MW2 is not supported.
Data type StringType of column PAGE3 is not supported.
Data type StringType of column RBMY1A1 is not supported.
Data type StringType of column RBMY1D is not supported.
Data type StringType of column RBMY1E is not supported.
Data type StringType of column RBMY1F is not supported.
Data type StringType of column SPANXN4 is not supported.
Data type StringType of column SSX4B is not supported.
Data type StringType of column TGIF2LY is not supported.
```

ERROR

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

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

```
import org.apache.spark.ml.PipelineModel
<console>:42: error: not found: value validatorModel
       val bestModel = validatorModel.bestModel
                       ^
```

READY

```
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