Predict_CHD

FINISHED

```
%md

## Predicting Heart Coronary Disease using Spark Mlib

* This tutorial gives steps of loading a dataset, doing some basic exploratory analysis and building a classificaiton model to predict heart coroı
```

Predicting Heart Coronary Disease using Spark Mlib

- This tutorial gives steps of loading a dataset, doing some basic exploratory analysis and building a classificaiton model to predict heart coronary disease

Took 1 seconds

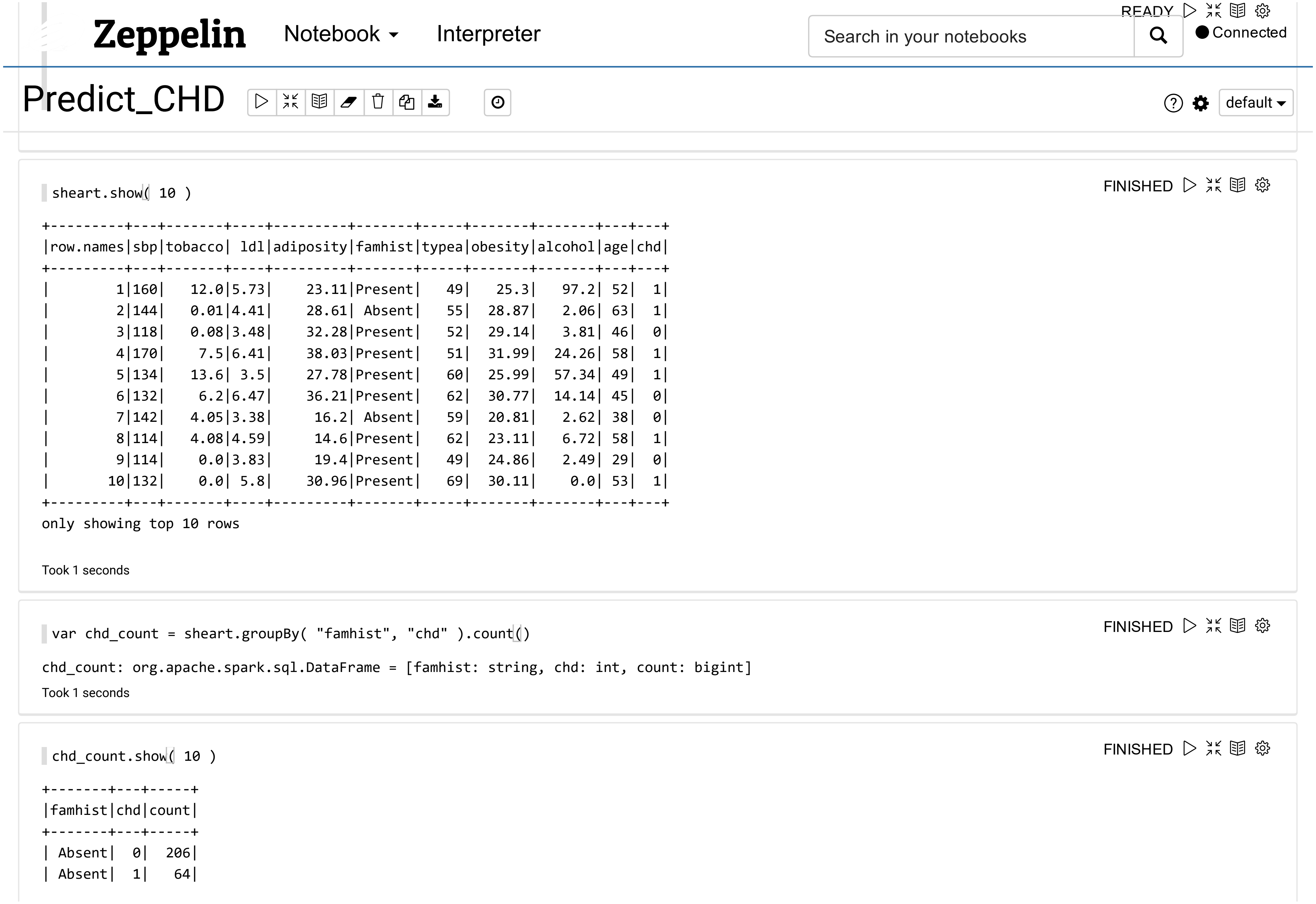
FINISHED

```
var sheart = sqlContext.read.format("com.databricks.spark.csv")
                            .option("delimiter", ",")
                            .option( "header", "true")
                            .option( "inferSchema", "true")
                            .load("file:///home/hadoop/lab/data/SAheart.data")
```

```
sheart: org.apache.spark.sql.DataFrame = [row.names: int, sbp: int, tobacco: double, ldl: double, adiposity: double, famhist: string, typea: int, obe
sity: double, alcohol: double, age: int, chd: int]
```

Took 1 seconds

READY



Predict_CHD

```
sheart.show( 10 )
```

FINISHED

```
+---------+---+-------+----+---------+-------+-----+-------+-------+---+---+
|row.names|sbp|tobacco| ldl|adiposity|famhist|typea|obesity|alcohol|age|chd|
+---------+---+-------+----+---------+-------+-----+-------+-------+---+---+
|        1|160|   12.0|5.73|    23.11|Present|   49|   25.3|   97.2| 52|  1|
|        2|144|   0.01|4.41|    28.61| Absent|   55|  28.87|   2.06| 63|  1|
|        3|118|   0.08|3.48|    32.28|Present|   52|  29.14|   3.81| 46|  0|
|        4|170|    7.5|6.41|    38.03|Present|   51|  31.99|  24.26| 58|  1|
|        5|134|   13.6| 3.5|    27.78|Present|   60|  25.99|  57.34| 49|  1|
|        6|132|    6.2|6.47|    36.21|Present|   62|  30.77|  14.14| 45|  0|
|        7|142|   4.05|3.38|     16.2| Absent|   59|  20.81|   2.62| 38|  0|
|        8|114|   4.08|4.59|     14.6|Present|   62|  23.11|   6.72| 58|  1|
|        9|114|    0.0|3.83|     19.4|Present|   49|  24.86|   2.49| 29|  0|
|       10|132|    0.0| 5.8|    30.96|Present|   69|  30.11|    0.0| 53|  1|
+---------+---+-------+----+---------+-------+-----+-------+-------+---+---+
only showing top 10 rows
```

Took 1 seconds

```
var chd_count = sheart.groupBy( "famhist", "chd" ).count()
```

FINISHED

```
chd_count: org.apache.spark.sql.DataFrame = [famhist: string, chd: int, count: bigint]
```

Took 1 seconds

```
chd_count.show( 10 )
```

FINISHED

```
+-------+---+-----+
|famhist|chd|count|
+-------+---+-----+
| Absent|  0|  206|
| Absent|  1|   64|
|Present|  0|   96|
|Present|  1|   96|
+-------+---+-----+
```

Took 2 seconds

FINISHED

```
import sqlContext.implicits._
chd_count.cache()
chd_count.registerTempTable( "chd_count" )
```

```
import sqlContext.implicits._
res43: org.apache.spark.sql.DataFrame = [famhist: string, chd: int, count: bigint]
```

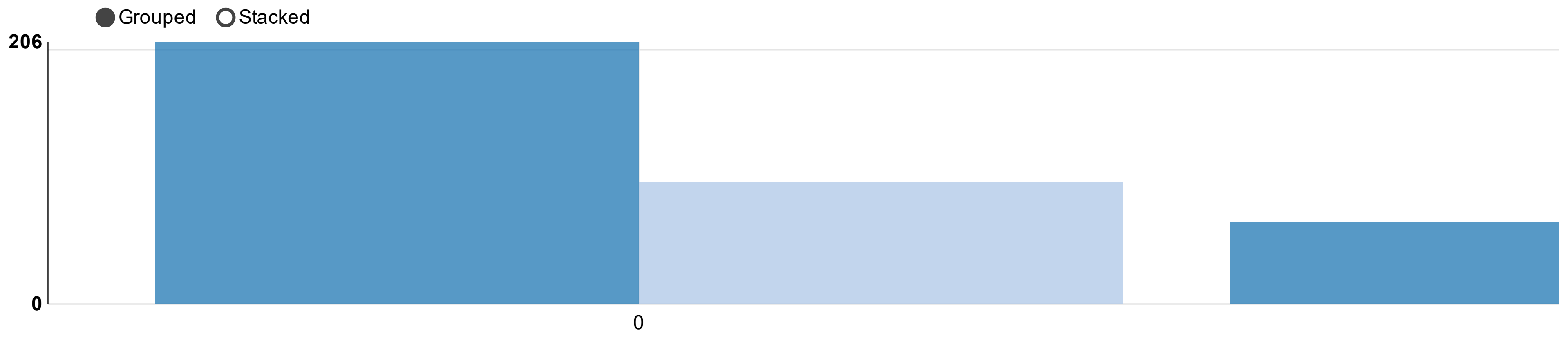
Took 2 seconds

FINISHED

```
%sql

select * from chd_count
```

settings



Took 1 seconds

```
sheart.cache()
sheart.registerTempTable( "sheart")
```

FINISHED

```
res46: org.apache.spark.sql.DataFrame = [row.names: int, sbp: int, tobacco: double, ldl: double, adiposity: double, famhist: string, typea: int, obes
ity: double, alcohol: double, age: int, chd: int]
```

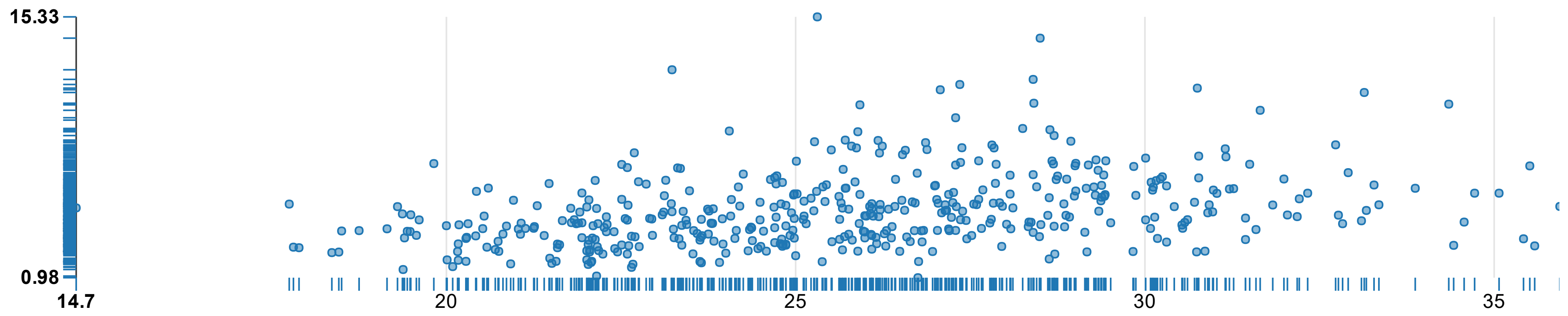
Took 1 seconds

```
%sql

select ldl, obesity from sheart
```

FINISHED

settings



Took 0 seconds

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
```

FINISHED

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
```

Took 1 seconds

FINISHED

```
var sheart_subset = sheart.select( "sbp","ldl","alcohol","tobacco","age" )
val assembler = new VectorAssembler()
  .setInputCols(Array("sbp","ldl","alcohol","tobacco","age"))
  .setOutputCol("features")

val sheart_vecs = assembler.transform(sheart_subset).select( "features" )
```

```
sheart_subset: org.apache.spark.sql.DataFrame = [sbp: int, ldl: double, alcohol: double, tobacco: double, age: int]
assembler: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_5fd4f13f8d6e
sheart_vecs: org.apache.spark.sql.DataFrame = [features: vector]
```

Took 1 seconds

FINISHED

```
sheart_vecs
```

```
res69: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[131] at rdd at <console>:56
```

Took 0 seconds (outdated)

FINISHED

```
import  java.lang.Double

def getVector( rec: Row ) = {

      Vectors.dense(rec.getAs("alcohol"),
                    rec.getAs("tobacco"),
                    rec.getAs[Int]("age").toDouble,
                    rec.getAs("obesity"),
                    rec.getAs("ldl"))
}
```

```
import java.lang.Double
getVector: (rec: org.apache.spark.sql.Row)org.apache.spark.mllib.linalg.Vector
```

Took 1 seconds

FINISHED

```
var sheart_vec = sheart.map( rec => getVector( rec) )
```

```
sheart_vec: org.apache.spark.rdd.RDD[org.apache.spark.mllib.linalg.Vector] = MapPartitionsRDD[139] at map at <console>:76
```

Took 0 seconds

FINISHED

```
sheart_vec.take( 10 )
```

```
res112: Array[org.apache.spark.mllib.linalg.Vector] = Array([97.2,12.0,52.0,25.3,5.73], [2.06,0.01,63.0,28.87,4.41], [3.81,0.08,46.0,29.14,3.48], [2
4.26,7.5,58.0,31.99,6.41], [57.34,13.6,49.0,25.99,3.5], [14.14,6.2,45.0,30.77,6.47], [2.62,4.05,38.0,20.81,3.38], [6.72,4.08,58.0,23.11,4.59], [2.49,
0.0,29.0,24.86,3.83], [0.0,0.0,53.0,30.11,5.8])
```

Took 0 seconds

FINISHED

```
val summary: MultivariateStatisticalSummary = Statistics.colStats( sheart_vec )
```

```
summary: org.apache.spark.mllib.stat.MultivariateStatisticalSummary = org.apache.spark.mllib.stat.MultivariateOnlineSummarizer@18bd5004
```

Took 0 seconds (outdated)

FINISHED

```
summary.mean
```

```
res137: org.apache.spark.mllib.linalg.Vector = [17.044393939393935,3.6356493506493526,42.8160173160173,26.044112554112544,4.7403246753246755]
```

Took 1 seconds

FINISHED

```
summary.variance
```

```
res126: org.apache.spark.mllib.linalg.Vector = [599.3222346644318,21.095870184804358,213.4216083988319,17.755101054549215,4.288664753359437]
```

Took 1 seconds

FINISHED

```
Statistics.corr(sheart_vec, method="pearson")
```

```
res156: org.apache.spark.mllib.linalg.Matrix =
1.0                   0.20081339040839782  0.10112464597373327  ... (5 total)
0.20081339040839782   1.0                  0.45033015960690737  ...
0.10112464597373327   0.45033015960690737  1.0                  ...
0.05161956861191215   0.12452941236866158  0.2917771263718622   ...
-0.03340339827374904  0.15890545800595926  0.31179923367413986  ...
```

Took 0 seconds

FINISHED

```
correlation
```

```
res153: org.apache.spark.mllib.linalg.Matrix =
1.0                   0.20081339040839782  0.10112464597373327  ... (5 total)
0.20081339040839782   1.0                  0.45033015960690737  ...
0.10112464597373327   0.45033015960690737  1.0                  ...
0.05161956861191215   0.12452941236866158  0.2917771263718622   ...
-0.03340339827374904  0.15890545800595926  0.31179923367413986  ...
```

Took 0 seconds

FINISHED

```
import org.apache.spark.mllib.regression.LabeledPoint

def parsePoint( rec: Row ) = {
  LabeledPoint( rec.getAs[Int]("chd"),
                Vectors.dense(rec.getAs("alcohol"),
                              rec.getAs("tobacco"),
                              rec.getAs[Int]("age").toDouble,
                              rec.getAs("obesity"),
                              rec.getAs("ldl") ) )
}
```

```
import org.apache.spark.mllib.regression.LabeledPoint
parsePoint: (rec: org.apache.spark.sql.Row)org.apache.spark.mllib.regression.LabeledPoint
```

Took 1 seconds

FINISHED

```
var sheart_lp = sheart.map( rec => parsePoint( rec ) )
```

```
sheart_lp: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[149] at map at <console>:79
```

Took 1 seconds

FINISHED

```
sheart_lp.take( 10 )
```

```
res165: Array[org.apache.spark.mllib.regression.LabeledPoint] = Array((1.0,[97.2,12.0,52.0,25.3,5.73]), (1.0,[2.06,0.01,63.0,28.87,4.41]), (0.0,[3.8
1,0.08,46.0,29.14,3.48]), (1.0,[24.26,7.5,58.0,31.99,6.41]), (1.0,[57.34,13.6,49.0,25.99,3.5]), (0.0,[14.14,6.2,45.0,30.77,6.47]), (0.0,[2.62,4.05,3
8.0,20.81,3.38]), (1.0,[6.72,4.08,58.0,23.11,4.59]), (0.0,[2.49,0.0,29.0,24.86,3.83]), (1.0,[0.0,0.0,53.0,30.11,5.8]))
```

Took 0 seconds

FINISHED

```
val splits = sheart_lp.randomSplit(Array(0.7, 0.3), seed = 11L)
val training = splits(0).cache()
val test = splits(1)
```

```
splits: Array[org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint]] = Array(MapPartitionsRDD[150] at randomSplit at <console>:81,
 MapPartitionsRDD[151] at randomSplit at <console>:81)
training: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[150] at randomSplit at <console>:81
test: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[151] at randomSplit at <console>:81
```

Took 1 seconds

FINISHED

```
import org.apache.spark.mllib.classification.{LogisticRegressionModel, LogisticRegressionWithLBFGS}
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.util.MLUtils

val model = new LogisticRegressionWithLBFGS().run(training)
```

```
import org.apache.spark.mllib.classification.{LogisticRegressionModel, LogisticRegressionWithLBFGS}
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.util.MLUtils
model: org.apache.spark.mllib.classification.LogisticRegressionModel = org.apache.spark.mllib.classification.LogisticRegressionModel: intercept = 0.
0, numFeatures = 5, numClasses = 2, threshold = 0.5
```

Took 3 seconds

FINISHED

```
// Compute raw scores on the test set.
val predictionAndLabels = test.map { case LabeledPoint(label, features) =>
  val prediction = model.predict(features)
  (prediction, label)
}
```

```
predictionAndLabels: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsRDD[217] at map at <console>:97
```

Took 1 seconds

FINISHED

```
// Get evaluation metrics.
val metrics = new BinaryClassificationMetrics(predictionAndLabels)
val auROC = metrics.areaUnderROC()

println("Area under ROC = " + auROC)
```

```
metrics: org.apache.spark.mllib.evaluation.BinaryClassificationMetrics = org.apache.spark.mllib.evaluation.BinaryClassificationMetrics@8b02b9f
auROC: Double = 0.617271505376344
Area under ROC = 0.617271505376344
```

Took 2 seconds

FINISHED

```
model.save(sc, "file:///home/hadoop/lab/programs/results/sheartmodel")
val sameModel = LogisticRegressionModel.load(sc,
  "file:///home/hadoop/lab/programs/results/sheartmodel")
```

```
sameModel: org.apache.spark.mllib.classification.LogisticRegressionModel = org.apache.spark.mllib.classification.LogisticRegressionModel: intercept =
 0.0, numFeatures = 5, numClasses = 2, threshold = 0.5
```

Took 3 seconds

READY