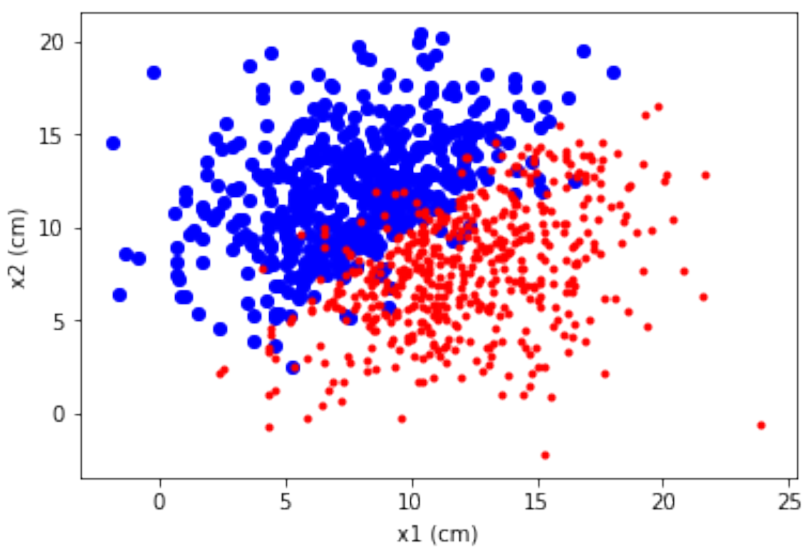
databricksLogisticRegression

Linear Border

```
path = "/FileStore/tables/"
fballs = path + "mixedBallsLarge2.csv"

mixedBalls = spark.read.csv( fballs, header=True, inferSchema=True )
mixedBalls.describe().show()
```

```
+-------+-----------------+-----------------+------+
|summary|               x1|               x2|colour|
+-------+-----------------+-----------------+------+
|  count|             1000|             1000|  1000|
|   mean|         10.01088|9.860119999999991|  null|
| stddev|4.034599710060519|3.876676541395554|  null|
|    min|            -1.83|            -2.14|  blue|
|    max|             23.9|            20.38|   red|
+-------+-----------------+-----------------+------+
```

```
from matplotlib import pyplot as plt
mixpan = mixedBalls.toPandas()
plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( mixpan[ mixpan["colour"]=="blue" ]["x1"], mixpan[
mixpan["colour"]=="blue" ]["x2"], c="b" )
plt.scatter( mixpan[ mixpan["colour"]=="red" ]["x1"], mixpan[
mixpan["colour"]=="red" ]["x2"], c="r", marker='.' )
```




```
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml import Pipeline
sindex = StringIndexer(inputCol="colour", outputCol="colourLabel" )
vecassem = VectorAssembler( inputCols=[ "x1", "x2" ], outputCol="features" )
pipe1 = Pipeline( stages=[ sindex, vecassem ] )
mixedBalls2 = pipe1.fit( mixedBalls ).transform( mixedBalls )
mixedBalls2.show( 5, False )
mixedBalls2.cache()
```

```
+-----+-----+------+-----------+-------------+
|x1   |x2   |colour|colourLabel|features     |
+-----+-----+------+-----------+-------------+
|8.79 |16.3 |blue  |1.0        |[8.79,16.3]  |
|15.07|17.52|blue  |1.0        |[15.07,17.52]|
|6.33 |13.63|blue  |1.0        |[6.33,13.63] |
|9.78 |13.28|blue  |1.0        |[9.78,13.28] |
|8.39 |6.94 |red   |0.0        |[8.39,6.94]  |
+-----+-----+------+-----------+-------------+
only showing top 5 rows

Out[110]: DataFrame[x1: double, x2: double, colour: string, colourLabel: double, fe
atures: vector]
```

```
# Training, test sets split
mix_train, mix_test = mixedBalls2.randomSplit( [.7, .3] )
```

```
from pyspark.ml.classification import LogisticRegression
lr1 = LogisticRegression( featuresCol='features', labelCol='colourLabel',
predictionCol='pred_colour' )
lrFit = lr1.fit( mix_train ) # LogisticRegressionModel
pred_test1 = lrFit.transform( mix_test )
pred_test1.drop("x1", "x2").show( 5, False)
#pred_test1.filter( "colourLabel != pred_colour" ).show( 25, False)
```

```
+------+-----------+-------------+-----------------------------------------+--------
----------------------------------+-----------+
|colour|colourLabel|features     |rawPrediction                            |probabil
ity                               |pred_colour|
+------+-----------+-------------+-----------------------------------------+--------
----------------------------------+-----------+
|blue  |1.0        |[-1.83,14.55]|[-19.353876526813554,19.353876526813554]|[3.93294
813488866E-9,0.9999999960670518]  |1.0        |
|blue  |1.0        |[0.72,8.19]  |[-8.452045474251815,8.452045474251815]  |[2.13417
77787598707E-4,0.999786582222124] |1.0        |
|blue  |1.0        |[0.87,6.37]  |[-6.002103020754881,6.002103020754881]  |[0.00246
74414599757943,0.9975325585400242]|1.0        |
|blue  |1.0        |[1.39,9.85]  |[-9.754473592210049,9.754473592210049]  |[5.80310
91831722965E-5,0.9999419689081682]|1.0        |
|blue  |1.0        |[1.69,9.43]  |[-8.88249567932413,8.88249567932413]    |[1.38778
0772803819E-4,0.9998612219227195] |1.0        |
+------+-----------+-------------+-----------------------------------------+--------
----------------------------------+-----------+
only showing top 5 rows
```

```
# Fit coefficients
print( lrFit.coefficients )
lrFit.intercept
```

```
[-1.052258878360654,1.0490219814931205]
Out[42]: -0.029147290999932646
```

```
pred_test1.collect()[2].probability[1]
```

```
Out[43]: 0.9999890581363439
```

```
from pyspark.sql import functions as fun
from pyspark.sql.types import FloatType
ff = fun.udf( lambda v : float(v.toArray()[1]), FloatType() )
pred_test1.withColumn( "p", ff(pred_test1.probability) ).drop("x1", "x2", "colour",
"rawPrediction").show(2, False)
```

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

```
-------+
|colourLabel|features    |probability                                |pred_colour|p
|
+-----------+------------+-------------------------------------------+-----------+--
-------+
|1.0        |[-1.61,6.41]|[2.272527592385294E-4,0.9997727472407615] |1.0        |0.
9997727|
|1.0        |[-1.37,8.61]|[2.9106305024753247E-5,0.9999708936949752]|1.0        |0.
9999709|
+-----------+------------+-------------------------------------------+-----------+--
-------+
only showing top 2 rows
```

```
# training evaluation
print( lrFit.summary.accuracy )
print( lrFit.summary.precisionByLabel )
print( " TN=", lrFit.summary.truePositiveRateByLabel[1],
       ", FP=", lrFit.summary.falsePositiveRateByLabel[0],
       "\n",
       "FN=", lrFit.summary.falsePositiveRateByLabel[1],
       ", TP=", lrFit.summary.truePositiveRateByLabel[0]
     )
```

```
0.9357142857142857
[0.9428571428571428, 0.9285714285714286]
 TN= 0.9420289855072463 , FP= 0.057971014492753624
 FN= 0.07042253521126761 , TP= 0.9295774647887324
```

```
# test evaluation
test_eval = lrFit.evaluate( mix_test )
print( test_eval.accuracy )
print( test_eval.precisionByLabel )
#
print( "  TN=", test_eval.truePositiveRateByLabel[0],  # TN/( all negatives 0 )
       ", TP=", test_eval.truePositiveRateByLabel[1],  # TP/( all positives 1 )
       "\n",
       " FN=", test_eval.falsePositiveRateByLabel[0], # FN/( all positives 1 )
       ", FP=", test_eval.falsePositiveRateByLabel[1]  # FP/( all negatives 0 )
     )
```

```
0.9733333333333334
[0.9612903225806452, 0.9862068965517241]
  TN= 0.9867549668874173 , TP= 0.959731543624161
```

```
  FN= 0.040268456375838924 , FP= 0.013245033112582781
```

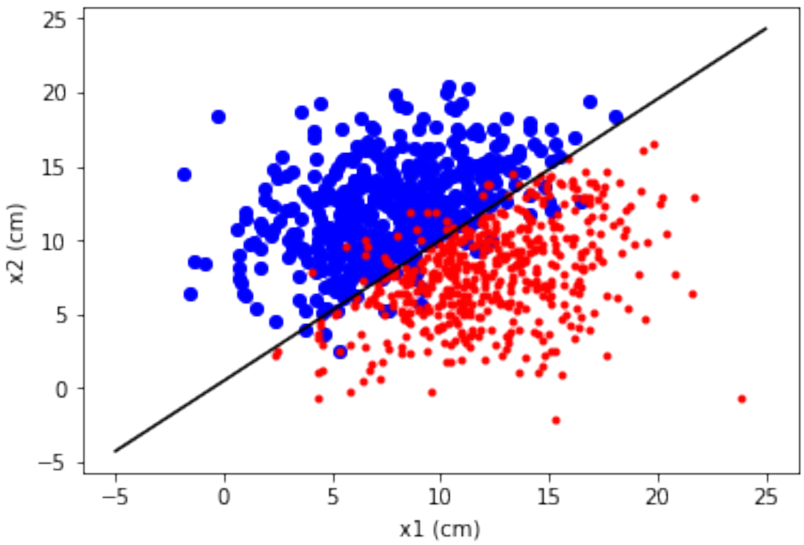
```
# Evaluation with MulticlassClassificationEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
eval1 = MulticlassClassificationEvaluator(predictionCol='pred_colour',
labelCol='colourLabel', metricName='accuracy')
eval1.evaluate( pred_test1 )
```

```
Out[49]: 0.9733333333333334
```

```
# confusion matrix
pred_test1.groupBy("colourLabel","pred_colour").count().show()
```

```
+-----------+-----------+-----+
|colourLabel|pred_colour|count|
+-----------+-----------+-----+
|        1.0|        1.0|  143|
|        0.0|        1.0|    2|
|        1.0|        0.0|    6|
|        0.0|        0.0|  149|
+-----------+-----------+-----+
```

```
# plotting
import numpy as np
x_1 = np.linspace(-5,25,1000)
x_2 = -(lrFit.intercept + lrFit.coefficients[0]*x_1)/lrFit.coefficients[1]
plt.plot( x_1, x_2, c='black' )

plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( mixpan[ mixpan["colour"]=="blue" ]["x1"], mixpan[
mixpan["colour"]=="blue" ]["x2"], c="b" )
plt.scatter( mixpan[ mixpan["colour"]=="red" ]["x1"], mixpan[
mixpan["colour"]=="red" ]["x2"], c="r", marker='.' )
```




```
mixedBalls2.unpersist()
```

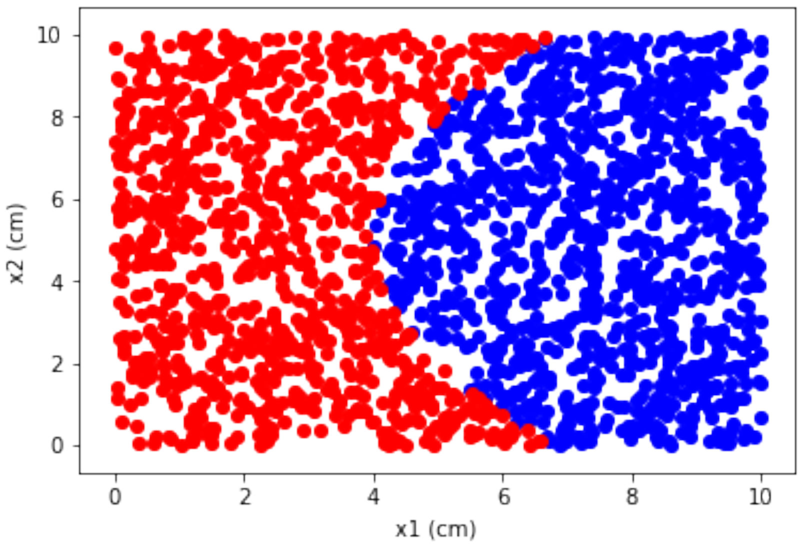
```
Out[81]: DataFrame[x1: double, x2: double, colour: string, colourLabel: double, fea
tures: vector]
```

parabolic border

```
path = "/FileStore/tables/"
fhballs = path + "mixedBallsHomogen2.csv"
hballs = spark.read.csv( fhballs, header=True, inferSchema=True )
hballs.describe().show()
```

```
+-------+-----------------+-----------------+------+
|summary|               x1|               x2|colour|
+-------+-----------------+-----------------+------+
|  count|             2000|             2000|  2000|
|   mean|5.114120000000003|5.044845000000002|  null|
| stddev| 2.91324244036746| 2.92933191508053|  null|
|    min|              0.0|              0.0|  blue|
|    max|             10.0|             10.0|   red|
+-------+-----------------+-----------------+------+
```

```
from matplotlib import pyplot as plt
hpan = hballs.toPandas()
plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( hpan[ hpan["colour"]=="blue" ]["x1"], hpan[ hpan["colour"]=="blue" ]
["x2"], c="b" )
plt.scatter( hpan[ hpan["colour"]=="red" ]["x1"], hpan[ hpan["colour"]=="red" ]
["x2"], c="r" )
```

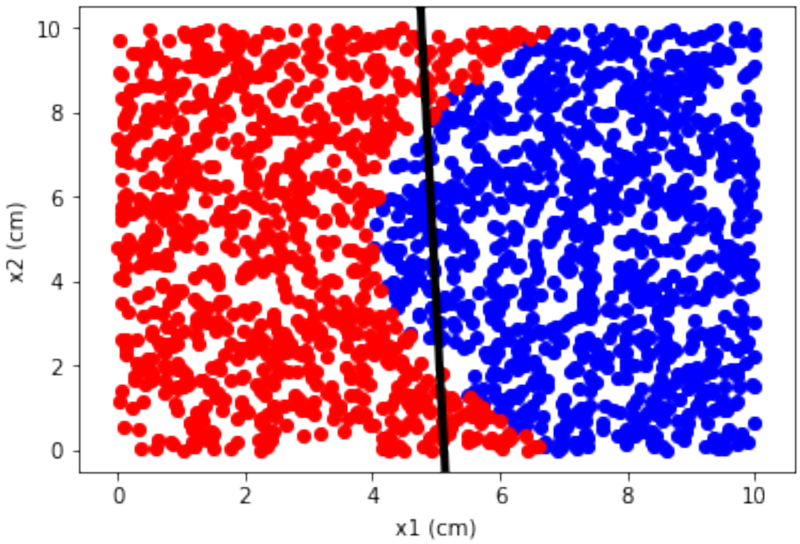


```
# Exercise : Fit a line to hballs2. Compute, the fit accuracy, precision and
confusion matrix for the test sample and plot the data and the fited line.

from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml import Pipeline
sindex2 = StringIndexer(inputCol="colour", outputCol="colourLabel" )
vecassem2 = VectorAssembler( inputCols=[ "x1", "x2" ], outputCol="features" )
pipe2 = Pipeline( stages=[ sindex2, vecassem2 ] )
hballs2 = pipe2.fit( hballs ).transform( hballs )
print( hballs2.show( 5, False ) )
hballs2.cache()

h_train, h_test = hballs2.randomSplit( [.7, .3] )

from pyspark.ml.classification import LogisticRegression
lr2 = LogisticRegression( featuresCol='features', labelCol='colourLabel',
predictionCol='pred_colour' )
lrFit2 = lr2.fit( h_train ) # LogisticRegressionModel
pred_test2 = lrFit2.transform( h_test )
print( pred_test2.drop("x1", "x2").show( 5, False) )

# test evaluation
test_eval2 = lrFit2.evaluate( h_test )
print( "Test sample accuracy : ", round( test_eval2.accuracy, 2 ) )
print( "Test sample precision : ", test_eval2.precisionByLabel )
print( "  TN=", test_eval2.truePositiveRateByLabel[0],  # TN/( all negatives 0 )
       ", TP=", test_eval2.truePositiveRateByLabel[1],  # TP/( all positives 1 )
       "\n",
       " FN=", test_eval2.falsePositiveRateByLabel[0], # FN/( all positives 1 )
       ", FP=", test_eval2.falsePositiveRateByLabel[1]  # FP/( all negatives 0 )
      )
```

```
+----+----+------+-----------+-----------+
|x1  |x2  |colour|colourLabel|features   |
+----+----+------+-----------+-----------+
|4.06|8.04|red   |1.0        |[4.06,8.04]|
|0.98|7.69|red   |1.0        |[0.98,7.69]|
|2.66|2.67|red   |1.0        |[2.66,2.67]|
|8.51|7.26|blue  |0.0        |[8.51,7.26]|
|5.7 |0.66|red   |1.0        |[5.7,0.66] |
+----+----+------+-----------+-----------+
only showing top 5 rows

None
+------+-----------+----------+---------------------------------------+----------
```

```
------------------------------+-----------+
|colour|colourLabel|features    |rawPrediction                             |probability
|pred_colour|
+------+-----------+-----------+-----------------------------------------+-----------
------------------------------+-----------+
|red   |1.0        |[0.02,9.69]|[-9.481134860361284,9.481134860361284]|[7.62715061
194686E-5,0.9999237284938806] |1.0        |
|red   |1.0        |[0.03,1.17]|[-9.99219945067536,9.99219945067536]  |[4.57533655
```

```
import numpy as np
x_1 = np.linspace(0, 10, 1000)
x_2 = -(lrFit2.intercept + lrFit2.coefficients[0]*x_1)/lrFit2.coefficients[1]
plt.plot( x_1, x_2, c='black', linewidth=4 )
plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( hpan[ hpan["colour"]=="blue" ]["x1"], hpan[ hpan["colour"]=="blue" ]
["x2"], c="b" )
plt.scatter( hpan[ hpan["colour"]=="red" ]["x1"], hpan[ hpan["colour"]=="red" ]
["x2"], c="r" )
plt.ylim(bottom=-.5, top=10.5)

hballs2.unpersist()
```



```
# Exercise : Add PolynomialExpansion class to your pipeline and fit a parabola to
the data. Compute, the fit accuracy, precision and confusion matrix for the test
sample and plot the data and the fited curve.

from pyspark.ml.feature import StringIndexer, VectorAssembler, PolynomialExpansion
from pyspark.ml import Pipeline
sindex3 = StringIndexer(inputCol="colour", outputCol="colourLabel" )
vecassem3 = VectorAssembler( inputCols=[ "x1", "x2" ], outputCol="f1" )
polex = PolynomialExpansion(degree=2, inputCol="f1", outputCol="features")
pipe3 = Pipeline( stages=[ sindex3, vecassem3, polex ] )
hballs3 = pipe3.fit( hballs ).transform( hballs )
print( hballs3.show( 5, False ) )
hballs3.cache()

h_train3, h_test3 = hballs3.randomSplit( [.7, .3] )

from pyspark.ml.classification import LogisticRegression
lr3 = LogisticRegression( featuresCol='features', labelCol='colourLabel',
predictionCol='pred_colour' )
lrFit3 = lr3.fit( h_train3 ) # LogisticRegressionModel
pred_test3 = lrFit3.transform( h_test3 )
print( pred_test3.drop("x1", "x2", "rawPrediction").show( 5, False) )

# test evaluation
test_eval3 = lrFit3.evaluate( h_test3 )
print( "Test sample accuracy : ", round( test_eval3.accuracy, 2 ) )
print( "Test sample precision : ", test_eval3.precisionByLabel )
print( "  TN=", test_eval3.truePositiveRateByLabel[0],  # TN/( all negatives 0 )
       ", TP=", test_eval3.truePositiveRateByLabel[1],  # TP/( all positives 1 )
       "\n",
       " FN=", test_eval3.falsePositiveRateByLabel[0], # FN/( all positives 1 )
       ", FP=", test_eval3.falsePositiveRateByLabel[1]  # FP/( all negatives 0 )
     )
```

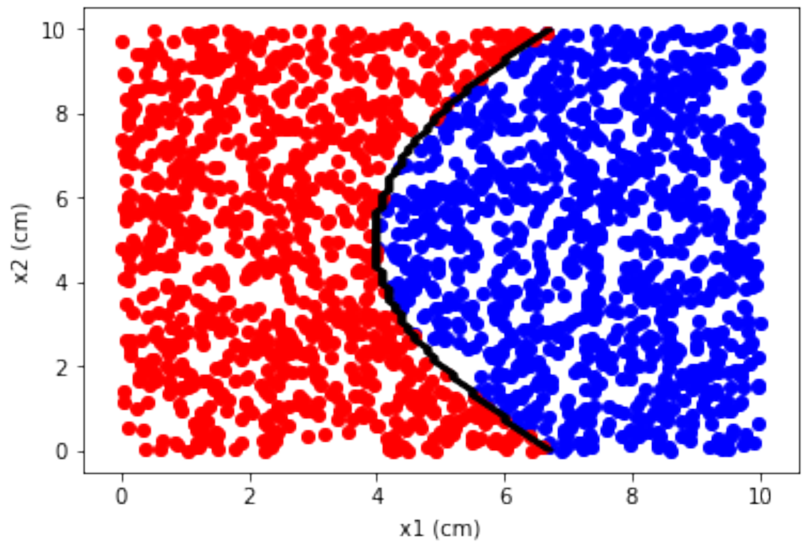
```
+----+----+------+-----------+-----------+---------------------------------------
--------------------------+
|x1  |x2  |colour|colourLabel|f1         |features
|
+----+----+------+-----------+-----------+---------------------------------------
--------------------------+
|4.06|8.04|red   |1.0        |[4.06,8.04]|[4.06,16.483599999999996,8.04,32.6423999
99999995,64.64159999999998]|
|0.98|7.69|red   |1.0        |[0.98,7.69]|[0.98,0.9603999999999999,7.69,7.5362,59.
136100000000006]           |
|2.66|2.67|red   |1.0        |[2.66,2.67]|[2.66,7.0756000000000006,2.67,7.1022,7.1
```

```
289]                        |
|8.51|7.26|blue  |0.0       |[8.51,7.26]|[8.51,72.42009999999999,7.26,61.78259999
9999995,52.7076]            |
|5.7 |0.66|red   |1.0       |[5.7,0.66] |[5.7,32.49,0.66,3.7620000000000005,0.435
60000000000004]             |
+----+----+------+----------+-----------+---------------------------------------
---------------------------+
only showing top 5 rows

None
```

```
import numpy as np
a, b = np.meshgrid( np.linspace(0,10,100), np.linspace(0,10,100) )
zz = list( zip( a.ravel().tolist(), b.ravel().tolist() ) )
Z = spark.createDataFrame( zz, ["x1", "x2"] )
```

```
from pyspark.ml.feature import VectorAssembler, PolynomialExpansion
from pyspark.ml import Pipeline
vecassem4 = VectorAssembler( inputCols=["x1", "x2"], outputCol="f" )
polex4 = PolynomialExpansion( degree=2, inputCol="f", outputCol="features" )
pipe4 = Pipeline( stages=[ vecassem4, polex4 ] )
Z2 = pipe4.fit( Z ).transform( Z )
#Z2.show()
pred_z = lrFit3.transform( Z2 ).select("pred_colour")
```

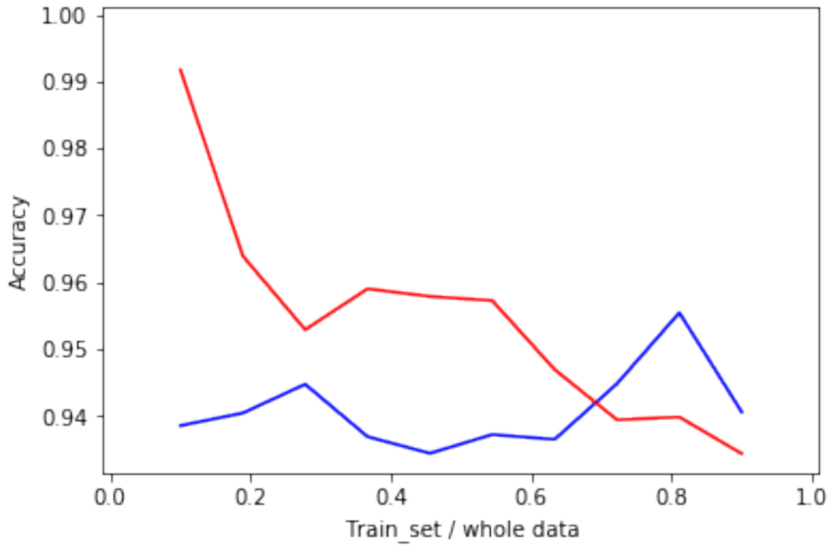
```
from matplotlib import pyplot as plt

plt.contour(a, b, pred_z.toPandas().values.reshape(a.shape), colors='black' )
plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( hpan[ hpan["colour"]=="blue" ]["x1"], hpan[ hpan["colour"]=="blue" ]
["x2"], c="b" )
plt.scatter( hpan[ hpan["colour"]=="red" ]["x1"], hpan[ hpan["colour"]=="red" ]
["x2"], c="r" )
plt.ylim(bottom=-.5, top=10.5)
```



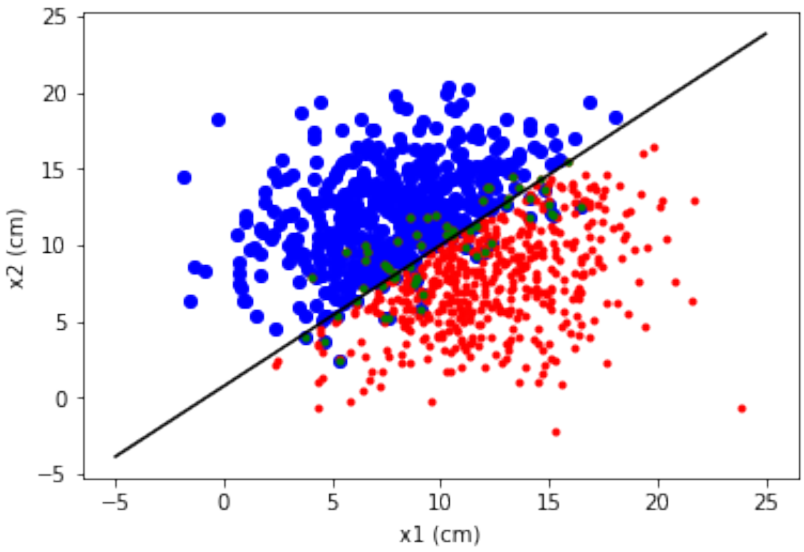
Convergence Diagram

```
# Exercise : Consider mixedBalls2 data. Plot the fit accuracies for training and
test samples vs. the training sample size for training sample size varing from 0.1
to 0.95 whole sample.

def computeAccuracy( df, ratio ) :
  h_train, h_test = df.randomSplit( [ ratio, 1-ratio ], 6 )
  lrFit = lr1.fit( h_train ) # LogisticRegressionModel
  pred_train = lrFit.transform( h_train )
  pred_test = lrFit.transform( h_test )

  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
  eval = MulticlassClassificationEvaluator(predictionCol='pred_colour',
labelCol='colourLabel', metricName='accuracy')

  return eval.evaluate( pred_train ), eval.evaluate( pred_test )

n_p = 10
ratio = np.linspace( .1, .9, n_p )
s_train, s_test= np.zeros(n_p), np.zeros(n_p)
for i in range( n_p ) :
  s_train[i], s_test[i] = computeAccuracy( mixedBalls2, ratio[i] )
```

```
plt.plot( 1-ratio, s_train, c='b' )
plt.plot( 1-ratio, s_test, c='r' )
plt.xlim( left=-.01, right=1.01 )
plt.ylim( top=1.001)
#plt.ylim( top=.98)
plt.xlabel("Train_set / whole data")
plt.ylabel("Accuracy")
```




```
# Execise : Find the balls that are mis-classified for the mixedBallsLarge2.csv.
Plot the data and the border. Mark the misclassified calss with green colour.

from pyspark.sql import functions as fun
pred_missc = lrFit.transform( mixedBalls2 )\
                  .filter( fun.col("colourLabel") != fun.col("pred_colour") )\
                  .toPandas()

pred_mix.show()
import numpy as np
x_1 = np.linspace(-5,25,1000)
x_2 = -(lrFit.intercept + lrFit.coefficients[0]*x_1)/lrFit.coefficients[1]
plt.plot( x_1, x_2, c='black' )

plt.xlabel("x1 (cm)")
plt.ylabel("x2 (cm)")
plt.scatter( mixpan[ mixpan["colour"]=="blue" ]["x1"], mixpan[
mixpan["colour"]=="blue" ]["x2"], c="b" )
plt.scatter( mixpan[ mixpan["colour"]=="red" ]["x1"], mixpan[
mixpan["colour"]=="red" ]["x2"], c="r", marker='.' )
plt.scatter( pred_missc["x1"], pred_missc["x2"], c="g", marker='.' )
```




```
sna.describe().show()
sna = sna.drop("User ID")
```

```
+-------+------+------------------+-----------------+------------------+
|summary|Gender|               Age|  EstimatedSalary|         Purchased|
+-------+------+------------------+-----------------+------------------+
|  count|   400|               400|              400|               400|
|   mean|  null|            37.655|          69742.5|            0.3575|
| stddev|  null|10.482876597307927|34096.9602824248|0.4798639635968691|
|    min|Female|              18.0|          15000.0|                 0|
|    max|  Male|              60.0|         150000.0|                 1|
+-------+------+------------------+-----------------+------------------+
```

```
# Exercise : Consider the file Social_Network_Ads.csv. It contains the online
shopping data of some internet users. The Purchased column is either 0 or 1 that
shows if a user has bought the article or not. Take the Age and the EstimatedSalary
as the features and classify the data by using simple linear logistic regression.
# Attention : the Age and the EstimatedSalary do not span the same value ranges.
You should rescale them.

path = "/FileStore/tables/"
fsname = path + "Social_Network_Ads.csv"
sna = spark.read.csv( fsname, header=True, inferSchema=True )
sna.describe().show()

# Preprocessing (Transformers pipeline)
stindex = StringIndexer( inputCol="Gender", outputCol="GenderInd" )
hoten = OneHotEncoderEstimator( inputCols=["GenderInd"], outputCols=["GenderR"] )
vecassem5 = VectorAssembler( inputCols=[ "Age", "EstimatedSalary", "GenderR" ],
outputCol="f1" )
stansca = StandardScaler( inputCol="f1", outputCol="features" )

pipe5 = Pipeline( stages=[stindex, hoten, vecassem5, stansca ] )
sna2 = pipe5.fit(sna ).transform( sna.drop("User ID") ).drop("GenderInd", "f1")
sna2.show(5, False)
sna2.cache()

# Estimator fitting
from pyspark.ml.classification import LogisticRegression
lr5 = LogisticRegression( featuresCol="features", labelCol="Purchased",
predictionCol="pred_purchased" )
sna_train, sna_test = sna2.randomSplit( [.7, .3] )
lr5Fit = lr5.fit( sna_train )
pred_test5 = lr5Fit.transform( sna_test )

# Fit evaluation
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
eval5 = MulticlassClassificationEvaluator( labelCol="Purchased",
predictionCol="pred_purchased", metricName="accuracy" )
print( eval5.evaluate( pred_test5 ),  lr5Fit.evaluate( sna_test ).accuracy )
pred_test5.groupby( "Purchased", "pred_purchased" ).count().show()

sna2.unpersist()
```

```
+-------+-----------------+------+------------------+-----------------+------------
------+
|summary|          User ID|Gender|               Age|  EstimatedSalary|         Pur
chased|
```

```
+-------+------------------+------+------------------+-----------------+------------
------+
|  count|               400|   400|               400|              400|
400|
|   mean|  1.56915397575E7|  null|            37.655|          69742.5|
0.3575|
| stddev|71658.32158119006|  null|10.482876597307927|34096.9602824248|0.4798639635
968691|
|    min|          15566689|Female|              18.0|          15000.0|
0|
|    max|          15815236|  Male|              60.0|         150000.0|
1|
+-------+------------------+------+------------------+-----------------+------------
------+

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

```
+-------+------------------+------+------------------+-----------------+-------------
-----+
|summary|           User ID|Gender|               Age| EstimatedSalary|          Purc
hased|
+-------+------------------+------+------------------+-----------------+-------------
-----+
|  count|               400|   400|               400|              400|
400|
|   mean|  1.56915397575E7|  null|            37.655|          69742.5|             0
.3575|
| stddev|71658.32158119006|  null|10.482876597307927|34096.9602824248|0.47986396359
68691|
|    min|          15566689|Female|              18.0|          15000.0|
0|
|    max|          15815236|  Male|              60.0|         150000.0|
1|
+-------+------------------+------+------------------+-----------------+-------------
-----+

+--------------------+---------------------------------+-------------------------+
|corr(Age, Purchased)|corr(EstimatedSalary, Purchased)|corr(GenderInd, Purchased)|
+--------------------+---------------------------------+-------------------------+
|  0.6224541988845287|              0.36208302580467777|     -0.04246945626450938|
+--------------------+---------------------------------+-------------------------+
```