In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-26-ML-classification")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [2]:
```
bInput = spark.read.format("parquet")\
    .load(SPARK_BOOK_DATA_PATH + "/data/binary-classification")
```

In [3]:
```
bInput.cache()
bInput.count()
```

Out[3]:
```
5
```

In [5]:
```
bInput.printSchema()
```

```
root
 |-- label: integer (nullable = true)
 |-- features: vector (nullable = true)
```

In [6]:
```
bInput = bInput.selectExpr("features", "cast(label as double) as label")
```

In [7]:
```
bInput.show(5)
```

```
+--------------+-----+
|      features|label|
+--------------+-----+
|[3.0,10.1,3.0]|  1.0|
|[1.0,0.1,-1.0]|  0.0|
|[1.0,0.1,-1.0]|  0.0|
| [2.0,1.1,1.0]|  1.0|
| [2.0,1.1,1.0]|  1.0|
+--------------+-----+
```

In [8]:
```
# COMMAND ----------

from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression()
print (lr.explainParams()) # see all parameters
lrModel = lr.fit(bInput)


# COMMAND ----------

print (lrModel.coefficients)
print (lrModel.intercept)
```

```
aggregationDepth: suggested depth for treeAggregate (>= 2). (default: 2)
elasticNetParam: the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an
L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0)
family: The name of family which is a description of the label distribution to be used in the mode
l. Supported options: auto, binomial, multinomial (default: auto)
featuresCol: features column name. (default: features)
fitIntercept: whether to fit an intercept term. (default: True)
labelCol: label column name. (default: label)
lowerBoundsOnCoefficients: The lower bounds on coefficients if fitting under bound constrained opti
mization. The bound matrix must be compatible with the shape (1, number of features) for binomial r
egression, or (number of classes, number of features) for multinomial regression. (undefined)
lowerBoundsOnIntercepts: The lower bounds on intercepts if fitting under bound constrained optimiza
tion. The bounds vector size must beequal with 1 for binomial regression, or the number oflasses fo
r multinomial regression. (undefined)
maxIter: max number of iterations (>= 0). (default: 100)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models out
put well-calibrated probability estimates! These probabilities should be treated as confidences, no
t precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)
regParam: regularization parameter (>= 0). (default: 0.0)
standardization: whether to standardize the training features before fitting the model. (default: T
rue)
threshold: Threshold in binary classification prediction, in range [0, 1]. If threshold and thresho
lds are both set, they must match.e.g. if threshold is p, then thresholds must be equal to [1-p,
p]. (default: 0.5)
thresholds: Thresholds in multi-class classification to adjust the probability of predicting each c
lass. Array must have length equal to the number of classes, with values > 0, excepting that at mos
t one value may be 0. The class with largest value p/t is predicted, where p is the original probab
ility of that class and t is the class's threshold. (undefined)
```

```
tol: the convergence tolerance for iterative algorithms (>= 0). (default: 1e-06)
upperBoundsOnCoefficients: The upper bounds on coefficients if fitting under bound constrained opti
mization. The bound matrix must be compatible with the shape (1, number of features) for binomial r
egression, or (number of classes, number of features) for multinomial regression. (undefined)
upperBoundsOnIntercepts: The upper bounds on intercepts if fitting under bound constrained optimiza
tion. The bound vector size must be equal with 1 for binomial regression, or the number of classes
for multinomial regression. (undefined)
weightCol: weight column name. If this is not set or empty, we treat all instance weights as 1.0.
(undefined)
[6.848741326855039,0.3535658901019763,14.814900276915905]
-10.225695864480985
```

In [9]:

```
# COMMAND ----------

summary = lrModel.summary
print (summary.areaUnderROC)
```

```
1.0
```

In [10]:

```
summary.roc.show()
```

```
+---+------------------+
|FPR|               TPR|
+---+------------------+
|0.0|               0.0|
|0.0|0.3333333333333333|
|0.0|               1.0|
|1.0|               1.0|
|1.0|               1.0|
+---+------------------+
```

In [11]:
```
summary.pr.show()
```

```
+------------------+---------+
|            recall|precision|
+------------------+---------+
|               0.0|      1.0|
|0.3333333333333333|      1.0|
|               1.0|      1.0|
|               1.0|      0.6|
+------------------+---------+
```

In [12]:
```
# COMMAND ----------

summary.objectiveHistory
```

Out[12]:
```
[0.6730116670092565,
 0.5042829330409728,
 0.36356862066874396,
 0.1252407018038338,
 0.08532556611276212,
 0.03550487641573043,
 0.01819649450857124,
 0.008817369922959128,
 0.004413673785392138,
 0.002194038351234705,
 0.0010965641148080834,
 0.0005476575519853134,
 0.00027376237951490067,
 0.0001368465223657472,
 6.841809037070565e-05,
 3.420707791038474e-05,
 1.710317666423187e-05,
 8.551470106426866e-06,
 4.275703677941408e-06,
 2.137824011778135e-06,
 1.0688564054651695e-06,
 5.342600202575246e-07,
 2.668135105897072e-07,
 1.3204627865313503e-07,
 6.768401481681744e-08,
 3.3145477184834275e-08,
 1.615143883748833e-08,
 8.309350118268315e-09]
```

In [27]:
```
# COMMAND ----------

from pyspark.mllib.evaluation import BinaryClassificationMetrics
out = lrModel.transform(bInput)\
  .select("prediction", "label")\
  .rdd.map(lambda x: (float(x[0]), float(x[1])))
metrics = BinaryClassificationMetrics(out)

# COMMAND ----------

print (metrics.areaUnderPR)
print (metrics.areaUnderROC)
print ("Receiver Operating Characteristic")
# metrics.roc.toDF().show()


# COMMAND ----------
```

```
1.0
1.0
Receiver Operating Characteristic
```

In [14]:
```python
# COMMAND ----------

from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier()
print (dt.explainParams())
dtModel = dt.fit(bInput)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. I
f true, the algorithm will cache node IDs for each instance. Caching can speed up training of deepe
r trees. Users can set how often should the cache be checkpointed or disable it by setting checkpoi
ntInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that t
he cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the check
point directory is not set in the SparkContext. (default: 10)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options: en
tropy, gini (default: gini)
labelCol: label column name. (default: label)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of cat
egories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 intern
al node + 2 leaf nodes. (default: 5)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node w
ill be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cause
s the left or right child to have fewer than minInstancesPerNode, the split will be discarded as in
valid. Should be >= 1. (default: 1)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models out
put well-calibrated probability estimates! These probabilities should be treated as confidences, no
t precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)
seed: random seed. (default: 956191873026065186)
```

In [15]:
```
# COMMAND ----------

from pyspark.ml.classification import RandomForestClassifier
rfClassifier = RandomForestClassifier()
print (rfClassifier.explainParams())
trainedModel = rfClassifier.fit(bInput)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes.
If true, the algorithm will cache node IDs for each instance. Caching can speed up training of de
eper trees. Users can set how often should the cache be checkpointed or disable it by setting che
ckpointInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that
the cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the ch
eckpoint directory is not set in the SparkContext. (default: 10)
featureSubsetStrategy: The number of features to consider for splits at each tree node. Supported
options: 'auto' (choose automatically for task: If numTrees == 1, set to 'all'. If numTrees > 1
(forest), set to 'sqrt' for classification and to 'onethird' for regression), 'all' (use all feat
ures), 'onethird' (use 1/3 of the features), 'sqrt' (use sqrt(number of features)), 'log2' (use l
og2(number of features)), 'n' (when n is in the range (0, 1.0], use n * number of features. When
n is in the range (1, number of features), use n features). default = 'auto' (default: auto)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options:
entropy, gini (default: gini)
labelCol: label column name. (default: label)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of c
ategories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 inte
rnal node + 2 leaf nodes. (default: 5)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node
will be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cau
ses the left or right child to have fewer than minInstancesPerNode, the split will be discarded a
s invalid. Should be >= 1. (default: 1)
numTrees: Number of trees to train (>= 1). (default: 20)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models o
utput well-calibrated probability estimates! These probabilities should be treated as confidence
s, not precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)
seed: random seed. (default: -5387697053847413545)
```

```
subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0,
1]. (default: 1.0)
```

In [16]:
```
# COMMAND ----------

from pyspark.ml.classification import GBTClassifier
gbtClassifier = GBTClassifier()
print (gbtClassifier.explainParams())
trainedModel = gbtClassifier.fit(bInput)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. I
f true, the algorithm will cache node IDs for each instance. Caching can speed up training of deepe
r trees. Users can set how often should the cache be checkpointed or disable it by setting checkpoi
ntInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that t
he cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the check
point directory is not set in the SparkContext. (default: 10)
featureSubsetStrategy: The number of features to consider for splits at each tree node. Supported o
ptions: 'auto' (choose automatically for task: If numTrees == 1, set to 'all'. If numTrees > 1 (for
est), set to 'sqrt' for classification and to 'onethird' for regression), 'all' (use all features),
'onethird' (use 1/3 of the features), 'sqrt' (use sqrt(number of features)), 'log2' (use log2(numbe
r of features)), 'n' (when n is in the range (0, 1.0], use n * number of features. When n is in the
range (1, number of features), use n features). default = 'auto' (default: all)
featuresCol: features column name. (default: features)
labelCol: label column name. (default: label)
lossType: Loss function which GBT tries to minimize (case-insensitive). Supported options: logistic
(default: logistic)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of cat
egories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 intern
al node + 2 leaf nodes. (default: 5)
maxIter: max number of iterations (>= 0). (default: 20)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node w
ill be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cause
s the left or right child to have fewer than minInstancesPerNode, the split will be discarded as in
valid. Should be >= 1. (default: 1)
predictionCol: prediction column name. (default: prediction)
seed: random seed. (default: 3504127614838123891)
stepSize: Step size (a.k.a. learning rate) in interval (0, 1] for shrinking the contribution of eac
h estimator. (default: 0.1)
subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0,
1]. (default: 1.0)
```

In [17]:
```
# COMMAND ----------

from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes()
print (nb.explainParams())
trainedModel = nb.fit(bInput.where("label != 0"))
```

```
featuresCol: features column name. (default: features)
labelCol: label column name. (default: label)
modelType: The model type which is a string (case-sensitive). Supported options: multinomial (defau
lt) and bernoulli. (default: multinomial)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models out
put well-calibrated probability estimates! These probabilities should be treated as confidences, no
t precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)
smoothing: The smoothing parameter, should be >= 0, default is 1.0 (default: 1.0)
thresholds: Thresholds in multi-class classification to adjust the probability of predicting each c
lass. Array must have length equal to the number of classes, with values > 0, excepting that at mos
t one value may be 0. The class with largest value p/t is predicted, where p is the original probab
ility of that class and t is the class's threshold. (undefined)
weightCol: weight column name. If this is not set or empty, we treat all instance weights as 1.0.
(undefined)
```

In [26]:
```
# COMMAND ----------

from pyspark.mllib.evaluation import BinaryClassificationMetrics
out = trainedModel.transform(bInput)\
  .select("prediction", "label")\
  .rdd.map(lambda x: (float(x[0]), float(x[1])))
metrics = BinaryClassificationMetrics(out)

# COMMAND ----------

print (metrics.areaUnderPR)
print (metrics.areaUnderROC)
print ("Receiver Operating Characteristic")
# metrics.roc.toDF().show()


# COMMAND ----------
```

```
0.6
0.5
Receiver Operating Characteristic
```

In [ ]: