

# 一、环境配置

## 1.1 Hadoop

```
root@hadoop01:~# start-all.sh
Starting namenodes on [101.42.45.176]
Starting datanodes
Starting secondary namenodes [hadoop02]
Starting resourcemanager
Starting nodemanagers
root@hadoop01:~# jps
11522 DataNode
12103 NodeManager
12279 Jps
```

## 1.2 Hive

### metastore

```
root@hadoop01:~# 2023-01-16 19:20:27: Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-3.1.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

### hiveserver2

```
root@hadoop01:~# hive --service hiveserver2 &
[1] 6470
root@hadoop01:~# 2023-01-16 19:20:39: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-3.1.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 986a0d6e-665f-4cca-9879-41d27c93ae9c
Hive Session ID = 17f085be-0bad-4855-bcb9-d57bac42267a
Hive Session ID = 5c2b7ca6-6d35-4a81-86ca-05c8457785b8
Hive Session ID = a07d68d2-1008-4c4f-abfe-eb558dc5b23e
OK
```

# 二、导入数据

## 2.1 建表语句

```
1 CREATE TABLE `human`.`aug_train` (
2     `enrollee__id` INT,
3     `city` DOUBLE,
4     `city_development_index` DOUBLE,
5     `gender` DOUBLE,
6     `relevent_experience` DOUBLE,
7     `enrolled_university` DOUBLE,
8     `education_level` DOUBLE,
9     `major_discipline` DOUBLE,
10    `experience` DOUBLE,
11    `company_size` DOUBLE,
12    `company_type` DOUBLE,
13    `last_new_job` DOUBLE,
14    `training_hours` DOUBLE, `target` DOUBLE)
15 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
16 STORED AS TEXTFILE;
```

```

1 CREATE TABLE `human`.`aug_test` (
2   `enrollee__id` INT,
3   `city` DOUBLE,
4   `city_development_index` DOUBLE,
5   `gender` DOUBLE,
6   `relevent_experience` DOUBLE,
7   `enrolled_university` DOUBLE,
8   `education_level` DOUBLE,
9   `major_discipline` DOUBLE,
10  `experience` DOUBLE,
11  `company_size` DOUBLE,
12  `company_type` DOUBLE,
13  `last_new_job` DOUBLE,
14  `training_hours` DOUBLE)
15  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
16  STORED AS TEXTFILE;

```

## 2.2 导入数据

```

1 load data local inpath '/usr/local/aug_test.csv' overwrite into table
  aug_test
2 load data local inpath '/usr/local/aug_train.csv' overwrite into table
  aug_train

```

aug_test.enrollee_id	aug_test.city	aug_test.city_development_index	aug_test.gender	aug_test.relevent_experience	aug_test.enrolled_university	aug_test.education_level	aug_test.major_discipline	aug_test.experience	aug_test.company_size	aug_test.company_type	aug_test.last_new_job	aug_test.training_hours
27385	5.0	13.0	0.827	1.0	4.0	0.0	1.0	1.0	1.0	1.0	39.0	1.0
9272	5.0	90.0	0.698	1.0	4.0	0.0	2.0	1.0	1.0	1.0	51.0	0.0
14249	5.0	46.0	0.762	1.0	5.0	0.0	0.0	1.0	1.0	1.0	48.0	0.0
7865	5.0	21.0	0.624	1.0	4.0	0.0	1.0	1.0	1.0	1.0	4.0	1.0
7463	5.0	13.0	0.827	1.0	4.0	0.0	1.0	1.0	1.0	1.0	31.0	1.0
25202	5.0	21.0	0.624	1.0	4.0	0.0	3.0	1.0	1.0	1.0	33.0	0.0
17189	5.0	21.0	0.624	1.0	4.0	0.0	1.0	1.0	1.0	1.0	43.0	0.0
25855	0.0	103.0	0.92	0.0	4.0	0.0	1.0	1.0	1.0	1.0	20.0	0.0
32126	5.0	160.0	0.92	1.0	4.0	0.0	1.0	1.0	1.0	1.0	130.0	0.0
29242	5.0	21.0	0.624	1.0	4.0	0.0	1.0	1.0	1.0	1.0	49.0	1.0
	5.0	5.0	500.0	4.0	1.0							

10 rows selected (0.136 seconds)  
0: jdbc:hive2://hadoop01:10000/default>

## 2.3 测试idea连接

enrollee__id	city	city_developmen...	ge
27385	13	0.827	
9272	90	0.698	
14249	46	0.762	

# 三、SparkMllib建模

## 3.1 连接hive，读取数据

```

1 var data_test = spark.read.table("human.aug_test")
2 var data_train_source = spark.read.table("human.aug_train")

```

## 3.2 提取需要的字段

```

1      var df =
2      data_train_source.select("target","city","city_development_index","gender",
3      "relevent_experience","enrolled_university","education_level","major_discipline",
4      "experience","company_size","company_type","last_new_job","training_hours")

```

### 3.3 转成LabeledPoint类型

```

1      var labeledPointRDD = df.rdd.map(row=>{
2      var label = row.getAs[Double]("target")
3      var city = row.getAs[Double]("city")
4      var city_development_index = row.getAs[Double]
5      ("city_development_index")
6      var gender = row.getAs[Double]("gender")
7      var relevent_experience = row.getAs[Double]("relevent_experience")
8      var enrolled_university = row.getAs[Double]("enrolled_university")
9      var education_level = row.getAs[Double]("education_level")
10     var major_discipline = row.getAs[Double]("major_discipline")
11     var experience = row.getAs[Double]("experience")
12     var company_size = row.getAs[Double]("company_size")
13     var company_type = row.getAs[Double]("company_type")
14     var last_new_job = row.getAs[Double]("last_new_job")
15     var training_hours = row.getAs[Double]("training_hours")
16
17     LabeledPoint(label, Vectors.dense(city, city_development_index, gender
18     , relevent_experience, enrolled_university, education_level, major_discipline,
19     experience, company_size, company_type, last_new_job, training_hours))

```

### 3.4 持久化

```

1      labeledPointRDD.cache()

```

## 3.5 建模部分

### 3.5.1 支持向量机

```

1      var svmModel: SVMModel = SVMWithSGD.train(trainRDD, 100)
2
3      var svmPredictAndActualRDD: RDD[(Double, Double)] = testRDD.map{
4      case LabeledPoint(label, features) => (svmModel.predict(features), label)
5      }
6      // 评价roc
7      var svmMetrics = new BinaryClassificationMetrics(svmPredictAndActualRDD)
8      var svmRoc = svmMetrics.areaUnderROC()
9
10     val nbTotalCorrect = testRDD.map { point =>
11     if (svmModel.predict(point.features) == point.label) 1 else 0
12     }.sum

```

```

13 val numData = testRDD.count()
14 val svmPre = nbTotalCorrect/numData

```

### 3.5.2 逻辑回归

```

1  var lr = new LogisticRegressionWithLBFGS().setNumClasses(2)
2
3  var lrModel:LogisticRegressionModel = lr.run(trainRDD)
4  var lrPredictAndActualRDD:RDD[(Double,Double)] = testRDD.map{
5      case LabeledPoint(label,features)=>(lrModel.predict(features),label)
6  }
7  var lrMetrics = new BinaryClassificationMetrics(lrPredictAndActualRDD)
8  var lrRoc = lrMetrics.areaUnderROC()
9
10 val LrnbTotalCorrect = testRDD.map { point =>
11     if (lrModel.predict(point.features) == point.label) 1 else 0
12 }.sum
13 val LrnumData = testRDD.count()
14 val LrAuc = LrnbTotalCorrect/LrnumData

```

### 3.5.3 决策树

```

1  var dtModel:DecisionTreeModel = DecisionTree.trainClassifier(trainRDD,2
2      ,Map[Int,Int](),"gini",6,2)
3
4      var dtPredictAndActualRDD:RDD[(Double,Double)] = testRDD.map{
5          case LabeledPoint(label,features) =>(dtModel.predict(features),label)
6      }
7      val dtMetrics = new BinaryClassificationMetrics(dtPredictAndActualRDD)
8      val dtRoc = dtMetrics.areaUnderROC()
9
10     val DtnbTotalCorrect = testRDD.map { point =>
11         if (dtModel.predict(point.features) == point.label) 1 else 0
12     }.sum
13     val DtnumData = testRDD.count()
14     val Dtauc = DtnbTotalCorrect/DtnumData

```

### 3.5.4 朴素贝叶斯

```

1      var nbModel = NaiveBayes.train(trainRDD,1.0)
2      var nbPredictAndActualRDD:RDD[(Double,Double)] = testRDD.map{
3          case LabeledPoint(label,features)=>(nbModel.predict(features),label)
4      }
5      var nbMetrics = new BinaryClassificationMetrics(nbPredictAndActualRDD)
6      val nbRoc = nbMetrics.areaUnderROC()
7
8      val NbnbTotalCorrect = testRDD.map { point =>
9          if (nbModel.predict(point.features) == point.label) 1 else 0
10     }.sum
11     val NbnumData = testRDD.count()
12     var nbauc = NbnbTotalCorrect/NbnumData

```

## 运行结果

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" "-javaagent:F:\IntelliJ IDEA 20
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/F:/maven_repository/org/slf4j/slf4j-log4j12/1.
SLF4J: Found binding in [jar:file:/F:/maven_repository/org/apache/logging/log4j/l
SLF4J: Found binding in [jar:file:/F:/maven_repository/ch/qos/logback/logback-cla
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
支持向量机 准确度 = 0.8022508038585209
支持向量机 ROC = 0.5140573284206582

逻辑回归 准确度 = 0.8303858520900321
逻辑回归 ROC = 0.6016299256853807

决策树 准确度 = 0.8062700964630225
决策树 ROC = 0.49900497512437814

朴素贝叶斯 准确度 = 0.6254019292604501
朴素贝叶斯 ROC = 0.4954932450717126
```

1 | 我们发现:四个模型的ROC都接近0.5, 其中决策树的准确度和roc都是不错的

## 3.6 模型评估

为了避免模型偶然性造成的损失, 我们对每个模型进行10次训练求每个模型的平均准确度和ROC

### 3.6.1 支持向量机

```
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/slf4j/slf4j-log4j12/1.7.30/slf4j-l
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/apache/logging/log4j/log4j-slf4j-impl/2.11.2/log4j-slf4j-impl-2.11.2.jar]
SLF4J: Found binding in [jar:file:/root/.m2/repository/ch/qos/logback/logback-classic/1.2.3/logback-classic-1.2.3.jar]
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
支持向量机10次平均 ROC = 0.5
支持向量机10次平均准确度 = 0.815970386039133

Process finished with exit code 0
```

### 3.6.2 逻辑回归

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/slf4j/slf4j-log4j12/1.7.30/slf4j-l
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/apache/logging/log4j/log4j-slf4j-impl/2.11.2/log4j-slf4j-impl-2.11.2.jar]
SLF4J: Found binding in [jar:file:/root/.m2/repository/ch/qos/logback/logback-classic/1.2.3/logback-classic-1.2.3.jar]
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
逻辑回归10次平均 ROC = 0.5276959860673766
逻辑回归10次平均准确度 = 0.8125714285714286

Process finished with exit code 0
```

### 3.6.3 决策树

```
/usr/java/jdk1.8.0_281-amd64/bin/java ...  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/slf4j/slf4j-  
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/apache/log  
SLF4J: Found binding in [jar:file:/root/.m2/repository/ch/qos/logback  
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an e  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
决策树10次平均 ROC = 0.6077480009265518  
决策树10次平均准确度 = 0.8080752212389382
```

### 3.6.4 朴素贝叶斯

```
/usr/java/jdk1.8.0_281-amd64/bin/java ...  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/slf4j/s  
SLF4J: Found binding in [jar:file:/root/.m2/repository/org/apache/  
SLF4J: Found binding in [jar:file:/root/.m2/repository/ch/qos/logb  
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for a  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactor  
朴素贝叶斯10次平均 ROC = 0.505493627151857  
朴素贝叶斯10次平均准确度 = 0.6243469174503656  
  
Process finished with exit code 0
```

1 | 我们发现：经过10次模型训练，朴素贝叶斯不适合上述数据集，其他三个均取得不错的效果

## 3.7 模型优化：决策树

- maxDepth：决策树最大深度
- maxBins：决策树每个节点的最大分支数

根据决策树的最大深度和每个节点的最大分支数进行调参，每个参数运行30次求准确度的平均值

```
1 | "C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" "-javaagent:F:\IntelliJ  
IDEA 2020.3.1\lib\idea_rt.jar=56251:F:\IntelliJ IDEA 2020.3.1\bin" -  
Dfile.encoding=UTF-8 -classpath  
C:\Users\DELL\AppData\Local\Temp\classpath61966627.jar  
com.program.DTModelPlus  
2 | SLF4J: Class path contains multiple SLF4J bindings.  
3 | SLF4J: Found binding in [jar:file:/F:/maven_repository/org/slf4j/slf4j-  
log4j12/1.7.30/slf4j-log4j12-  
1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
4 | SLF4J: Found binding in  
[jar:file:/F:/maven_repository/org/apache/logging/log4j/log4j-slf4j-  
impl/2.10.0/log4j-slf4j-impl-  
2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
```

```
5 SLF4J: Found binding in
  [jar:file:/F:/maven_repository/ch/qos/logback/logback-classic/1.2.3/logback-
  classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
6 SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an
  explanation.
7 SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
8 maxDepth is : 3 maxBins is : 2
9 决策树30次平均准确度 = 0.8125530110262933
10
11 maxDepth is : 3 maxBins is : 3
12 决策树30次平均准确度 = 0.8142493638676845
13
14 maxDepth is : 3 maxBins is : 4
15 决策树30次平均准确度 = 0.8320610687022904
16
17 maxDepth is : 3 maxBins is : 5
18 决策树30次平均准确度 = 0.8371501272264628
19
20 maxDepth is : 3 maxBins is : 6
21 决策树30次平均准确度 = 0.8439355385920277
22
23 maxDepth is : 4 maxBins is : 2
24 决策树30次平均准确度 = 0.8125530110262933
25
26 maxDepth is : 4 maxBins is : 3
27 决策树30次平均准确度 = 0.815945716709076
28
29 maxDepth is : 4 maxBins is : 4
30 决策树30次平均准确度 = 0.8227311280746398
31
32 maxDepth is : 4 maxBins is : 5
33 决策树30次平均准确度 = 0.8227311280746398
34
35 maxDepth is : 4 maxBins is : 6
36 决策树30次平均准确度 = 0.8396946564885501
37
38 maxDepth is : 5 maxBins is : 2
39 决策树30次平均准确度 = 0.8125530110262933
40
41 maxDepth is : 5 maxBins is : 3
42 决策树30次平均准确度 = 0.8193384223918575
43
44 maxDepth is : 5 maxBins is : 4
45 决策树30次平均准确度 = 0.8210347752332484
46
47 maxDepth is : 5 maxBins is : 5
48 决策树30次平均准确度 = 0.8269720101781174
49
50 maxDepth is : 5 maxBins is : 6
51 决策树30次平均准确度 = 0.8379983036471581
52
53 maxDepth is : 6 maxBins is : 2
54 决策树30次平均准确度 = 0.8100084817642069
55
56 maxDepth is : 6 maxBins is : 3
```



```
57 决策树30次平均准确度 = 0.8184902459711622
58
59 maxDepth is : 6 maxBins is : 4
60 决策树30次平均准确度 = 0.8320610687022904
61
62 maxDepth is : 6 maxBins is : 5
63 决策树30次平均准确度 = 0.8329092451229851
64
65 maxDepth is : 6 maxBins is : 6
66 决策树30次平均准确度 = 0.8337574215436805
67
68 maxDepth is : 7 maxBins is : 2
69 决策树30次平均准确度 = 0.8057675996607294
70
71 maxDepth is : 7 maxBins is : 3
72 决策树30次平均准确度 = 0.8074639525021208
73
74 maxDepth is : 7 maxBins is : 4
75 决策树30次平均准确度 = 0.8235793044953351
76
77 maxDepth is : 7 maxBins is : 5
78 决策树30次平均准确度 = 0.8320610687022904
79
80 maxDepth is : 7 maxBins is : 6
81 决策树30次平均准确度 = 0.825275657336726
82
83
84 Process finished with exit code 0
85
```