

# 211275026-陈畅-实验四

---

<https://github.com/Cc17952/FBDP>

## 环境安装

已有环境

Scala安装

Spark环境安装

使用pyspark遇到的问题

IP未设置

pyspark库无法导入

WARN: Setting default log level to "WARN".

## 实验内容

### 任务一

编写 Spark 程序，统计application\_data.csv中所有用户的贷款金额AMT\_CREDIT 的分布情况。

编写Spark程序，统计application\_data.csv中客户贷款金额AMT\_CREDIT 比客户收入AMT\_INCOME\_TOT...

### 任务二

统计所有男性客户（CODE\_GENDER=M）的小孩个数（CNT\_CHILDREN）类型占比情况。

统计每个客户出生以来每天的平均收入。

### 任务三

实验设计

选取变量

实验结果

实验存在的不足

<https://github.com/Cc17952/FBDP>

## 环境安装

已有环境

Scala版本	Spark 2.x版本	Spark 3.x版本
2.11	✓	
2.12	✓	✓

Scala和Spark对应关系

Spark版本	Hadoop版本
2.4.x	2.7.x
3.0.x	3.2.x

Spark和Hadoop对应关系

当前环境：Hadoop-2.9.2；Java-8

预下载版本：Scala-2.11.7；Spark-2.4.2

## Scala安装

下载链接：<https://github.com/scala/scala/releases/tag/v2.11.7>

参考教程：<https://www.runoob.com/scala/scala-install.html>

```

hadoop@ubuntu:/usr/local
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
export PATH=${JAVA_HOME}/bin:$PATH
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_LIBEXEC_DIR=$HADOOP_HOME/libexec
export JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native:$JAVA_LIBRARY_PATH
export HDFS_DATANODE_USER=root
export HDFS_DATANODE_SECURE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export HDFS_NAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
export MAVEN_HOME=/usr/local/apache-maven-3.9.5
export PATH=${MAVEN_HOME}/bin:$PATH
export HBASE_HOME=/usr/local/hbase
export PATH=$PATH:$HBASE_HOME/bin:$HBASE_HOME/sbin
export PATH=$PATH:/usr/local/scala/bin
53,12 底端

```

```
hadoop@ubuntu:~/Downloads$ sudo mv scala-2.11.7 /usr/local/scala
[sudo] hadoop 的密码:
hadoop@ubuntu:~/Downloads$ cd ..
hadoop@ubuntu:~$ cd /usr/local/
hadoop@ubuntu:/usr/local$ sudo vim /etc/profile
hadoop@ubuntu:/usr/local$ source /etc/profile
hadoop@ubuntu:/usr/local$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_362).
Type in expressions for evaluation. Or try :help.

scala>
```

由于2.11.7解压得到2.11.12，大版本没有出现问题所以采用2.11.12

## Spark环境安装

参考教程：[https://blog.csdn.net/Nurbiya\\_K/article/details/100982166](https://blog.csdn.net/Nurbiya_K/article/details/100982166)

```
hadoop@ubuntu:~/spark/conf
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# Generic options for the daemons used in the standalone deploy mode
# - SPARK_CONF_DIR      Alternate conf dir. (Default: ${SPARK_HOME}/conf)
# - SPARK_LOG_DIR       Where log files are stored. (Default: ${SPARK_HOME}/logs)
# - SPARK_PID_DIR       Where the pid file is stored. (Default: /tmp)
# - SPARK_IDENT_STRING  A string representing this instance of spark. (Default: $USER)
# - SPARK_NICENESS      The scheduling priority for daemons. (Default: 0)
# - SPARK_NO_DAEMONIZE  Run the proposed command in the foreground. It will not output a PID file.
# Options for native BLAS, like Intel MKL, OpenBLAS, and so on.
# You might get better performance to enable these options if using native BLAS (see SPARK-21305).
# - MKL_NUM_THREADS=1   Disable multi-threading of Intel MKL
# - OPENBLAS_NUM_THREADS=1 Disable multi-threading of OpenBLAS
export SCALA_HOME=/usr/local/scala
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
SPARK_MASTER_IP=ubuntu #这里Master11是我的主机名
SPARK_LOCAL_DIRS=/usr/local/spark
SPARK_DRIVER_MEMORY=1G

-- 插入 --
71,51 底端
```

mv slaves.template slaves

vim slaves//修改节点名称（改为了h01、h02、h03）

```

LICENSE-antlr.txt          100% 1490      7.7MB/s   00:00
LICENSE-netlib.txt         100% 2300     10.8MB/s   00:00
LICENSE-paranamer.txt      100% 1661      8.4MB/s   00:00
spark-2.4.0-yarn-shuffle.jar 100% 9489KB 229.1MB/s  00:00
hadoop@ubuntu:/usr/local$

```

scp -r spark/ h01:/usr/local/

```

hadoop@ubuntu:/usr/local$ scp -r scala/ h01:/usr/local/
Warning: Permanently added 'h01' (ECDSA) to the list of known hosts.
hadoop@h01's password:

```

scp -r scala/ h01:/usr/local/

```

9.2.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of
security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further
illegal access operations
WARNING: All illegal access operations will be denied in a future
release
localhost: Warning: Permanently added 'localhost' (ECDSA) to the list of
known hosts.
hadoop@localhost's password:
localhost: starting nodemanager, logging to /usr/local/hadoop/nodemanager-ubuntu.out
hadoop@ubuntu:/usr/local/hadoop$ jps
8275 SecondaryNameNode
7732 NameNode
8872 NodeManager
9048 Jps
7980 DataNode
8476 ResourceManager
hadoop@ubuntu:/usr/local/hadoop$

```

先启动hadoop

```

hadoop@ubuntu:/usr/local/spark$ sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs
/spark-hadoop-org.apache.spark.deploy.master.Master-1-ubuntu.out
h01: Warning: Permanently added 'h01' (ECDSA) to the list of known hosts.
hadoop@h01's password: h03: ssh: Could not resolve hostname h03: Name or service
not known
h02: ssh: Could not resolve hostname h02: Name or service not known

```

再启动Spark

这里发现上述slaves文件配置错误，修改为h01。

```
hadoop@ubuntu:/usr/local/spark$ jps
9218 Jps
8275 SecondaryNameNode
7732 NameNode
8872 NodeManager
9098 Master
7980 DataNode
8476 ResourceManager
```

```
Spark context Web UI available at http://192.168.217.145:4040
Spark context available as 'sc' (master = local[*], app id = local-170262871
).
Spark session available as 'spark'.
Welcome to

  ____  __
 / ___/ /_
/ /   / __ \
/ /___/ /_/ /
/_____/_/

version 2.4.0

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_362)
Type in expressions to have them evaluated.
Type :help for more information.

scala> █
```

## 使用pyspark遇到的问题

### IP未设置

```
vim spark-env.sh
```

来设置SPARK\_LOCAL\_IP=主机ip

### pyspark库无法导入

尝试了比较多的解决方法，例如pip install、修改环境变量

最终通过代码导入。

```
task1.py > ...
1  import sys
2  sys.path.append("/usr/local/spark/python/lib/py4j-0.10.7-src.zip")
3  sys.path.append("/usr/local/spark/python/lib/pyspark.zip")
4
5  from pyspark import SparkContext
```

WARN: Setting default log level to "WARN".

具体报错信息：Setting default log level to "WARN".To adjust logging level use  
sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

```
7  sc = SparkContext( 'local', 'test')  
8  # sc.setLogLevel("INFO")
```

解决方法：

注释行

```
2023-12-24 12:16:01 INFO SparkContext:54 - Invoking stop() from shutdown hook  
hadoop@ubuntu:~/lab4$ 2023-12-24 12:16:01 INFO AbstractConnector:318 - Stopped Spark@392ffb88{HTTP/1.1,[http/1.1]}{192.168.217.147:4040}  
2023-12-24 12:16:01 INFO SparkUI:54 - Stopped Spark web UI at http://192.168.217.147:4040  
2023-12-24 12:16:01 INFO MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!  
2023-12-24 12:16:01 INFO MemoryStore:54 - MemoryStore cleared  
2023-12-24 12:16:01 INFO BlockManager:54 - BlockManager stopped  
2023-12-24 12:16:01 INFO BlockManagerMaster:54 - BlockManagerMaster stopped  
2023-12-24 12:16:01 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!  
2023-12-24 12:16:01 INFO SparkContext:54 - Successfully stopped SparkContext  
2023-12-24 12:16:01 INFO ShutdownHookManager:54 - Shutdown hook called  
2023-12-24 12:16:01 INFO ShutdownHookManager:54 - Deleting directory /usr/local/spark/spark-9579c5e2-a646-48f6-a1e8-db3ee7f5658a/pyspark-61abe4f9-6c7  
2023-12-24 12:16:01 INFO ShutdownHookManager:54 - Deleting directory /tmp/spark-e99d8b3f-6d16-4bc5-9b03-5710ed5febcb8  
2023-12-24 12:16:01 INFO ShutdownHookManager:54 - Deleting directory /usr/local/spark/spark-9579c5e2-a646-48f6-a1e8-db3ee7f5658a  
/usr/bin/python /home/hadoop/lab4/task1.py
```

实际上这个报错信息可以不用理会

## 实验内容

### 任务一

编写 Spark 程序，统计application\_data.csv中所有用户的贷款金额AMT\_CREDIT 的分布情况。

```
1  # -*- coding: utf-8 -*-
2  import sys
3  import math
4  from pyspark import SparkContext
5  from pyspark.sql import SparkSession
6  from pyspark.sql.functions import col
7
8  sys.path.append("/usr/local/spark/python/lib/py4j-0.10.7-src.zip")
9  sys.path.append("/usr/local/spark/python/lib/pyspark.zip")
10 spark = SparkSession.builder.appName("LoanAmountDistribution").getOrCreate()
11 df = spark.read.csv("file:///home/hadoop/lab4/data/application_data.csv",
12                     header=True, inferSchema=True)
13 df = df.withColumn("AMT_CREDIT_BIN", ((col("AMT_CREDIT") / 10000).cast("int") * 10000).alias("AMT_CREDIT_BIN"))
14 credit_distribution = df.groupBy("AMT_CREDIT_BIN").count().sort("AMT_CREDIT_BIN")
15
16 credit_distribution.show(truncate=False)
17 result = credit_distribution.rdd.map(lambda x: "({},{}) , {}".format(x['AMT_CREDIT_BIN'], x['AMT_CREDIT_BIN'] + 10000, x['count'])).collect()
18
19 with open('./result/task1-1.txt', 'w') as file:
20     for line in result:
21         file.write(line + '\n')
```

```
resource manager ... task1.py 3 task1-1.txt x
LAB4
  .vscode
  {} launch.json
  data
  application_data.csv
  result
  task1-1.txt
  task1.py 3
result > task1-1.txt
1 ((40000,50000), 561)
2 ((50000,60000), 891)
3 ((60000,70000), 719)
4 ((70000,80000), 1226)
5 ((80000,90000), 668)
6 ((90000,100000), 1939)
7 ((100000,110000), 1871)
8 ((110000,120000), 1930)
9 ((120000,130000), 1323)
10 ((130000,140000), 4792)
11 ((140000,150000), 2239)
12 ((150000,160000), 3653)
13 ((160000,170000), 1919)
14 ((170000,180000), 2131)
15 ((180000,190000), 8745)
16 ((190000,200000), 1537)
17 ((200000,210000), 4017)
18 ((210000,220000), 1475)
19 ((220000,230000), 10013)
20 ((230000,240000), 3343)
21 ((240000,250000), 4206)
22 ((250000,260000), 6796)
23 ((260000,270000), 5186)
24 ((270000,280000), 10328)

问题 3 输出 调试控制台 终端 端口
|110000|1930|
|120000|1323|
|130000|4792|
|140000|2239|
|150000|3653|
|160000|1919|
|170000|2131|
|180000|8745|
|190000|1537|
|200000|4017|
|210000|1475|
|220000|10013|
|230000|3343|
+-----+-----+
only showing top 20 rows
```

运行结果

编写Spark程序，统计application\_data.csv中客户贷款金额AMT\_CREDIT 比客户收入AMT\_INCOME\_TOTAL差值最高和最低的各十条记录。



```
1  # -*- coding: utf-8 -*-
2  import sys
3  import math
4  from pyspark import SparkContext
5  from pyspark.sql import SparkSession
6  from pyspark.sql.functions import col
7  sys.path.append("/usr/local/spark/python/lib/py4j-0.10.7-src.zip")
8  sys.path.append("/usr/local/spark/python/lib/pyspark.zip")
9  spark = SparkSession.builder.appName("LoanAmountDistribution").getOrCreate()
10 df = spark.read.csv("file:///home/hadoop/lab4/data/application_data.csv",
11                      header=True, inferSchema=True)
12
13 df = df.withColumn("DIFF", col("AMT_CREDIT") - col("AMT_INCOME_TOTAL"))
14 top_records = df.orderBy(col("DIFF").desc()).limit(10)
15 bottom_records = df.orderBy(col("DIFF").asc()).limit(10)
16
17 output_path = "./result/task1-2.txt"
18 with open(output_path, 'w') as file:
19     file.write("Top 10 records with highest difference:\n")
20     file.write(top_records.select("SK_ID_CURR", "NAME_CONTRACT_TYPE", "AMT_CREDIT", "AMT_INCOME_TOTAL", "DIFF").toPandas().to_string(index=False))
21     file.write("\n\nBottom 10 records with lowest difference:\n")
22     file.write(bottom_records.select("SK_ID_CURR", "NAME_CONTRACT_TYPE", "AMT_CREDIT", "AMT_INCOME_TOTAL", "DIFF").toPandas().to_string(index=False))
```

```
... task1-2.py 3 task1-2.txt x
result > task1-2.txt
1 Top 10 records with highest difference:
2 SK_ID_CURR NAME_CONTRACT_TYPE AMT_CREDIT AMT_INCOME_TOTAL DIFF
3 433294 Cash loans 4050000.0 405000.0 3645000.0
4 210956 Cash loans 4031032.5 430650.0 3600382.5
5 434170 Cash loans 4050000.0 450000.0 3600000.0
6 315893 Cash loans 4027680.0 458550.0 3569130.0
7 238431 Cash loans 3860019.0 292050.0 3567969.0
8 240007 Cash loans 4050000.0 587250.0 3462750.0
9 117337 Cash loans 4050000.0 760846.5 3289153.5
10 120926 Cash loans 4050000.0 783000.0 3267000.0
11 117085 Cash loans 3956274.0 749331.0 3206943.0
12 228135 Cash loans 4050000.0 864900.0 3185100.0
13 Bottom 10 records with highest difference:
14 SK_ID_CURR NAME_CONTRACT_TYPE AMT_CREDIT AMT_INCOME_TOTAL DIFF
15 114967 Cash loans 562491.0 117000000.0 -116437509.0
16 336147 Cash loans 675000.0 18000090.0 -17325090.0
17 385674 Cash loans 1400503.5 13500000.0 -12099496.5
18 190160 Cash loans 1431531.0 9000000.0 -7568469.0
19 252084 Cash loans 790830.0 6750000.0 -5959170.0
20 337151 Cash loans 450000.0 4500000.0 -4050000.0
21 317748 Cash loans 835380.0 4500000.0 -3664620.0
22 310601 Cash loans 675000.0 3950059.5 -3275059.5
23 432980 Cash loans 1755000.0 4500000.0 -2745000.0
24 157471 Cash loans 953460.0 3600000.0 -2646540.0
```

运行结果

任务二

基于Hive或者Spark SQL对application\_data.csv进行如下统计：

```
1  # -*- coding: utf-8 -*-
2
3  import sys
4  sys.path.append("/usr/local/spark/python/lib/py4j-0.10.7-src.zip")
5  sys.path.append("/usr/local/spark/python/lib/pyspark.zip")
6  import math
7  from pyspark import SparkContext
8  from pyspark.sql import SparkSession
9  from pyspark.sql.functions import col
10 from pyspark.sql import functions as F
11
12 spark = SparkSession.builder.appName("GenderChildIncomeAnalysis").getOrCreate()
13 df = spark.read.csv("file:///home/hadoop/lab4/data/application_data.csv",
14                     header=True, inferSchema=True)
15
16 # task2-1
17 male_customers = df.filter(col("CODE_GENDER") == "M")
18 child_count_stats = male_customers.groupBy("CNT_CHILDREN").count()
19
20 total_male_customers = male_customers.count()
21 child_count_stats = child_count_stats.withColumn("TYPE_RATIO", col("count") / total_male_customers)
22
23 # child_count_stats.select("CNT_CHILDREN", "TYPE_RATIO").show(truncate=False)
24 outputpath_task1 = "./result/task2-1.txt"
25 with open(outputpath_task1, 'w') as file:
26     file.write("CNT_CHILDREN,TYPE_RATIO\n")
27     for row in child_count_stats.collect():
28         file.write("{}{}\n".format(row['CNT_CHILDREN'], row['TYPE_RATIO']))
29
30 # task2-2
31 income_stats = df.withColumn("avg_income", F.abs(col("AMT_INCOME_TOTAL") / col("DAYS_BIRTH")))
32 filtered_income_stats = income_stats.filter(col("avg_income") > 1).orderBy(col("avg_income").desc())
33
34 outputpath_task2 = "./result/task2-2.csv"
35 pandas_filtered_income_stats = filtered_income_stats.select("SK_ID_CURR", "avg_income").toPandas()
36 pandas_filtered_income_stats.to_csv(outputpath_task2, header=True, index=False)
```

```
36
```

```
37     snark.stop()
```

统计所有男性客户（CODE\_GENDER=M）的小孩个数（CNT\_CHILDREN）类型占比情况。

```
task2.py 3 task2-1.txt x
result > task2-1.txt
1  CNT_CHILDREN,TYPE_RATIO
2  1,0.215688327511
3  6,0.000104703071607
4  3,0.0137636946858
5  5,0.000314109214822
6  9,9.51846105522e-06
7  4,0.00161813837939
8  8,9.51846105522e-06
9  7,3.80738442209e-05
10 11,9.51846105522e-06
11 14,9.51846105522e-06
12 2,0.099115734968
13 0,0.669319144481
14 |
```

统计每个客户出生以来每天的平均收入。

```
task2.py 4 task2-2.csv X
result > task2-2.csv
1 SK_ID_CURR,avg_income
2 114967,9274.673008323425
3 336147,1146.2105196128375
4 385674,996.2364401151207
5 190160,547.945205479452
6 219563,417.51716459454445
7 310601,373.63408059023834
8 157471,360.4325190228274
9 252084,348.9995346672871
10 199821,269.6548418024928
11 337151,243.75710958236283
12 141198,243.62367661212704
13 429258,241.65939450896153
14 196091,240.76187758596092
15 317748,240.44883783061715
16 432980,239.56558773424192
17 217276,235.32048408785298
18 445335,234.30843510366373
19 387126,230.46532045654084
20 304300,223.5839682013912
21 123587,207.24967490247073
22 399467,195.92192148610405
23 441639,192.4557351809084
24 440768,192.04096873999788
25 225210,188.75388133737036
26 206341,186.00165334802975
27 134526,183.68846436443792
28 214063,180.08271655144367
29 336135,177.46478873239437
30 111903,174.13512885999535
31 269498,172.6027397260274
32 194130,172.0051983793288
33 431111,166.75931072818233
34 251262,159.32023366067604
```

### 任务三

基于Spark MLlib 或者Spark ML编写程序对贷款是否违约进行分类，并评估实验结果的准确率。

#### 实验设计

选用朴素贝叶斯分类器和决策树算法分别对贷款违约进行预测，比较两个模型的优劣。

#### 选取变量

从5C角度考虑：

1. 个人品质: "NAME\_EDUCATION\_TYPE", "FLAG\_CONT\_MOBILE"
2. 还款能力: "AMT\_INCOME\_TOTAL", "AMT\_CREDIT",  
"NAME\_INCOME\_TYPE", "CNT\_CHILDREN"
3. 资本实力: "FLAG\_OWN\_CAR", "FLAG\_OWN\_REALTY", "NAME\_HOUSING\_TYPE"
4. 担保: 未找到明确的质押信息
5. 外部经营（信誉）环境: "OBS\_30\_CNT\_SOCIAL\_CIRCLE"

## 实验结果

```
50 dt = DecisionTreeClassifier(featuresCol="features", labelCol="TARGET")
51 dt_model = dt.fit(trainingData)
52 dt_predictions = dt_model.transform(testData)
53 dt_evaluator = MulticlassClassificationEvaluator(labelCol="TARGET", pred
54 dt_accuracy = dt_evaluator.evaluate(dt_predictions)
55 print("DecisionTree Accuracy:", dt_accuracy)
56
57
```

问题 输出 调试控制台 终端 端口

```
hadoop@ubuntu:~/lab4$ /usr/bin/python /home/hadoop/lab4/task3.py
DecisionTree Accuracy:0.917754
NaiveBayes Accuracy:0.863485
```

从准确率的角度来说，该实验中决策树的精度高于朴素贝叶斯。

## 实验存在的不足

由于时间关系，没有对变量做筛选再选择，并比较不同变量的效果等，而是直接采取了部分特征变量。在变量特征提取阶段，对于"CNT\_CHILDREN"这样类型变量也没有做归一化处理，可能导致模型训练精度有误。最后的比较只参考了accuracy一个指标。