181840356-周娴静-实验4报告

一、题目

> 1. 分别编写MapReduce程序和Spark程序统计双十一最热门的商品和最受年轻人(age<30)关注的商家 （“添加购物车+购买+添加收藏夹”前100名）；
> 2. 编写Spark程序统计双十一购买了商品的男女比例，以及购买了商品的买家年龄段的比例；
> 3. 基于Hive或者Spark SQL查询双十一购买了商品的男女比例，以及购买了商品的买家年龄段的比例；
> 4. 预测给定的商家中，哪些新消费者在未来会成为忠实客户，即需要预测这些新消费者在6个月内再次购买的概率。基于Spark MLlib编写程序预测回头客，评估实验结果的准确率。

用户行为日志	user_log_format1
用户画像	user_info_format1
训练数据	train_format1
测试数据	test_format1

二、设计思路及程序关键代码

1. Task1

（1）Task11（Mapreduce）

- 类比作业5 wordcount shakespeare
- 【限定时间为双十一】首先筛选出time_stamp=1111的商品
- 【对商品计数】筛选action_type !=0的数据，以商品item_id作为它的key

```
    public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            //把每一行拆解成一个一个属性,下标从0开始
            String[] temp=word.toString().split(",");
            //首先筛选出time_stamp=1111的商品   [5]
            //对action_type，如果是1，2，3（不等于0）则计数  [6]
            if(temp[5].equals("1111"))
                if(!temp[6].equals("0"))
                {
                    //以商品id   item_id作为它的key     [1]
                    context.write(new Text(temp[1]), one);
                }
            }
        }
}
```

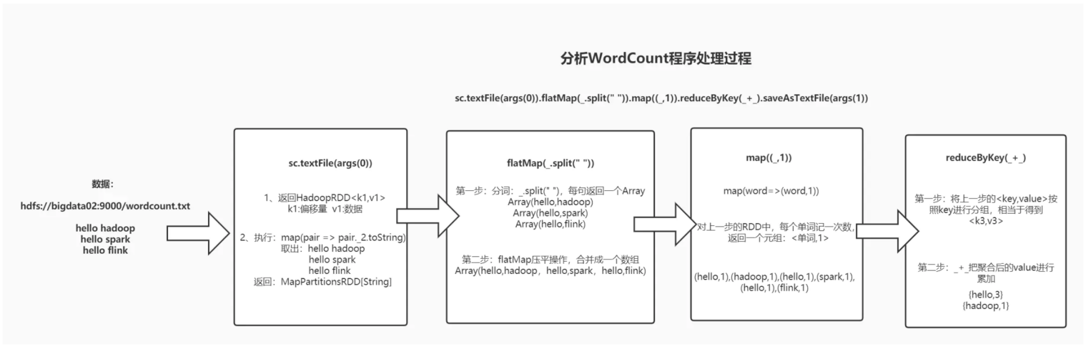
(2) Task12 (Mapreduce)

- 与Task11类似
- 仿照作业5（wordcount shakespeare）设置停词文件user_info_skip.csv
- 存储patternsToSkip的HashSet，供后续使用

```
public void setup(Context context) throws IOException,
InterruptedException {
    conf = context.getConfiguration( );
    //caseSensitive = conf.getBoolean ( "wordcount.case.sensitive",true);
    Path patternsPath=new
Path("hdfs://localhost:9000/Task12/user_info_skip.csv");
    String patternsFileName = patternsPath.getName( ).toString();
    parseSkipFile(patternsFileName);
    //    }
    //}
}
```

```
while(itr.hasMoreTokens()){
    word.set(itr.nextToken());
    //把每一行拆解成一个一个属性,下标从0开始
    String[] temp=word.toString().split(",");
    //首先筛选出time_stamp=1111的商品   [5]
    //对action_type，如果是1，2，3（不等于0）则计数  [6]
    //购物者是年轻人   [0]
    if(temp[5].equals("1111"))
        if(!temp[6].equals("0"))
            if(! patternsToSkip.contains(temp[0])){
                //以商家id   merchant_id作为它的key     [3]
                context.write(new Text(temp[3]),one);       //用
context.write收集<key,value>对
                Counter counter =
context.getCounter(CountersEnum.class.getName(),CountersEnum.INPUT_WORDS.t
oString());  //2.0
                counter.increment(1);   //2.0
    }
```

(3) Task11 (Spark)


分析WordCount程序处理过程
sc.textFile(args(0)).flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).saveAsTextFile(args(1))
数据:
hdfs://bigdata02:9000/wordcount.txt
hello hadoop
hello spark
hello flink
sc.textFile(args(0))
1、返回HadoopRDD<k1,v1>
k1:偏移量 v1:数据
2、执行: map(pair => pair._2.toString)
取出: hello hadoop
hello spark
hello flink
返回: MapPartitionsRDD[String]
flatMap(_.split(" "))
第一步: 分词: _.split(" "), 每句返回一个Array
Array(hello,hadoop)
Array(hello,spark)
Array(hello,flink)
第二步: flatMap压平操作, 合并成一个数组
Array(hello,hadoop, hello,spark, hello,flink)
map((_,1))
map(word=>(word,1))
对上一步的RDD中, 每个单词记一次数, 返回一个元组: <单词,1>
(hello,1),(hadoop,1),(hello,1),(spark,1),(hello,1),(flink,1)
reduceByKey(_+_)
第一步: 将上一步的<key,value>按照key进行分组, 相当于得到<k3,v3>
第二步: _+_把聚合后的value进行累加
{hello,3}
{hadoop,1}

- 读取数据文件，去除第一行
- 首先筛选出双十一（“1111”）的数据
- 进一步筛选action_type !=0的数据
- 实现二次排序

```
val data=sc.textFile("hdfs://localhost:9000/Task12/input2/user_log_format1.csv").flatMap(_.split("\n"))
val arr = data.take(1)
val data1 = data.filter(!arr.contains(_)).filter(line=>line.split(",")(5).equals("1111")).map{
            line => (line.split(",")(1), line.split(",")(6))
          }.mapValues(_.toInt)
val data2 = data1.filter(value => value._2>0)
val data3 = data2.countByKey().toSeq.sortWith(_._2>_._2).take(100)
val data4 = sc.parallelize(data3)
data4.saveAsTextFile("hdfs://localhost:9000/Task12/output1-1")
:quit
hdfs dfs -cat /Task12/output1-1/*
```

(4) Task12 (Spark)

```
val log = spark.read.format("csv").option("header","true").load("hdfs://localhost:9000/Task12/input2/user_log_format1.csv")
val info = spark.read.format("csv").option("header","true").load("hdfs://localhost:9000/Task12/user_info_format1.csv")
val info1 = info.filter("age_range<4 and age_range>0").select("user_id","age_range")
val log1 = log.filter("time_stamp=1111 and action_type!=0").select("user_id","seller_id","action_type")
val dfjoin = info1.join(log1,"user_id")
val dfss = dfjoin.groupBy("seller_id").count()
  //orderby count in desc and take first 100
```

```
val rddss = dfss.orderBy(dfss("count").desc).rdd.map(x=>
(x(0),x(1))).take(100)
  //save as testFile
sc.parallelize(rddss).saveAsTextFile("hdfs://localhost:9000/Task12/output1
-2")
:quit
hdfs dfs -cat /Task12/outp
```

2. Task2

(1) Task21

- 基本数据筛选，筛选出双十一购买商品的数据条以及gender为（0，1）的数据
- 进行dataframe的join
- 对gender作进一步分组，并统计

```
val log
=spark.read.format("csv").option("header","true").load("hdfs://localhost:9
000/Task12/input2/user_log_format1.csv")
val info
=spark.read.format("csv").option("header","true").load("hdfs://localhost:9
000/Task12/user_info_format1.csv")
val log1 =
log.select("user_id","time_stamp","action_type").filter("time_stamp=1111
and action_type=2")
val info1 = info.select("user_id","gender").filter("gender=1 or gender=0")
val dfjoin1 = info1.join(log1,"user_id")
val dfcount1 = dfjoin1.groupBy("gender").count()
dfcount1.withColumn("ratio",dfcount1("count")/dfjoin1.count).show
val rddr1 =
dfcount1.withColumn("ratio",dfcount1("count")/dfjoin1.count).select("gende
r","ratio").rdd.map(x=>(x(0),x(1))).collect()
sc.parallelize(rddr1).saveAsTextFile("hdfs://localhost:9000/Task12/output2
-1")
```

(2) Task22

- 与Task21类似

```
val log
=spark.read.format("csv").option("header","true").load("hdfs://localhost:9
000/Task12/input2/user_log_format1.csv")
val info
=spark.read.format("csv").option("header","true").load("hdfs://localhost:9
000/Task12/user_info_format1.csv")
```

```
val info1 = info.select("user_id","age_range").filter("age_range>0 and age_range<9")
val log1 = log.select("user_id","time_stamp","action_type").filter("time_stamp=1111 and action_type=2")
val dfjoin = info1.join(log1,"user_id")
val dfcount = dfjoin.groupBy("age_range").count()
dfcount.withColumn("ratio",dfcount("count")/dfjoin.count).show
/*将答案保存到本地*/
val dfratio = dfcount.withColumn("ratio",dfcount("count")/dfjoin.count).select("age_range","ratio").orderBy("age_range")
val rddr = dfratio.rdd.map(x=>(x(0),x(1)))
val rddr = dfratio.rdd.map(x=>(x(0),x(1))).collect()
sc.parallelize(rddr).saveAsTextFile("hdfs://localhost:9000/Task12/output2-2")
```

3. Task3

(1）Task31

- 将数据存储为dataframe以后进行createOrReplaceTempView试图转化，方便后续进行表的连接
- 采用嵌套查询方法，内部select初步筛选出购买了商品且gender为（0，1）的数据，设置distinct保证最多指读取该用户id一次
- 外部select进一步根据gender分组，并进行统计

```
//terminal
cd $SPARK_HOME
bin/spark-shell


//scala
val user_log = spark.read.format("csv").option("header","true").load("hdfs://localhost:9000/Task12/input2/user_log_format1.csv")

val user_info = spark.read.format("csv").option("header","true").load("hdfs://localhost:9000/Task12/user_info_format1.csv")

user_log.createOrReplaceTempView("table_log")

user_info.createOrReplaceTempView("table_info")
//3-1
val people1=spark.sql("select gender,count(*) as num from (select distinct a.user_id,gender from table_log a,table_info b where a.user_id=b.user_id and a.action_type='2' and gender in('0','1')) group by gender")
```

```
people1.show
//下面显示ratio
people1.agg("num"-> "sum").show //结果为407308
people1.withColumn("ratio",people1("num")/407308).show
```

(2) Task32

- 与Task31类似

```
//3-2
val people2=spark.sql("select age_range,count(*) as num from (select distinct a.user_id,age_range from table_log a,table_info b where a.user_id=b.user_id and a.action_type='2' and age_range in('1','2','3','4','5','6','7','8')) group by age_range order by age_range")
people2.show
// //下面显示ratio(但不是sql语句
people2.agg("num"-> "sum").show //结果为329039
people2.withColumn("ratio",people2("num")/329039).show
```

4. Task4

参考:

https://www.jianshu.com/p/aa6cb1ef6f69
https://tianchi.aliyun.com/notebook-ai/detail?postId=143593

- 参考Baseline做法，选取age_range、gender、total_logs、unique_item_ids、categories、browse_days、one_clicks、shopping_carts、purchase_times、favourite_times 建立特征工程
- 对建立好的特征作缺失值处理
- 利用pyspark-ml随机森林进行预测

```
#jupyter notebook
# In[1]:
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('random_forest').getOrCreate()
df=spark.read.csv('hdfs://localhost:9000/Task12/feature.csv',inferSchema=True,header=True)
# In[2]:
from pyspark.ml.feature import VectorAssembler
df_assembler = VectorAssembler(inputCols=['age_range','gender','total_logs','unique_item_ids','categories','browse_days','one_clicks','shopping_carts','purchase_times','favourite_times'], outputCol="features")
df = df_assembler.transform(df)
```

```
-测试集和训练集划分比例：0.75：0.25
-随机森林基学习器数量：120
# In[3]:
model_df = df.select(['features','label'])
train_df,test_df=model_df.randomSplit([0.75,0.25])
# In[4]:
from pyspark.ml.classification import RandomForestClassifier
rf_classifier=RandomForestClassifier(labelCol='label',numTrees=120).fit(train_df)
rf_predictions=rf_classifier.transform(test_df)
# In[5]:
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
rf_accuracy=MulticlassClassificationEvaluator(labelCol='label',metricName='accuracy').evaluate(rf_predictions)
print('MulticlassClassificationEvaluator 随机森林测试的准确性： {0:.0%}'.format(rf_accuracy))
# In[6]:
rf_auc=BinaryClassificationEvaluator(labelCol='label').evaluate(rf_predictions)
print('BinaryClassificationEvaluator 随机森林测试的准确性： {0:.0%}'.format(rf_auc))
```

三、Ubuntu命令行关键代码

1. Mapreduce指令

```
cd ~/hadoop_installs/hadoop-3.3.0
sbin/start-all.sh
jps
cd /mnt/hgfs/ubuntu_share/Experiment4/No1.2
bin/hdfs dfs -mkdir /Task12
#input1放的是mytest.CSV测试样例
bin/hdfs dfs -mkdir /Task12/input1
bin/hdfs dfs -put /mnt/hgfs/ubuntu_share/Experiment4/No1.2/user_info_skip.csv /Task12/
bin/hdfs dfs -put /mnt/hgfs/ubuntu_share/Experiment4/No1.2/mytest11.CSV /Task12/input1
bin/hadoop jar /mnt/hgfs/ubuntu_share/Experiment4/No1.2/popular_merchant-7.0-SNAPSHOT.jar project.code /Task12/input2 /Task12/output7
```

2. 安装Spark

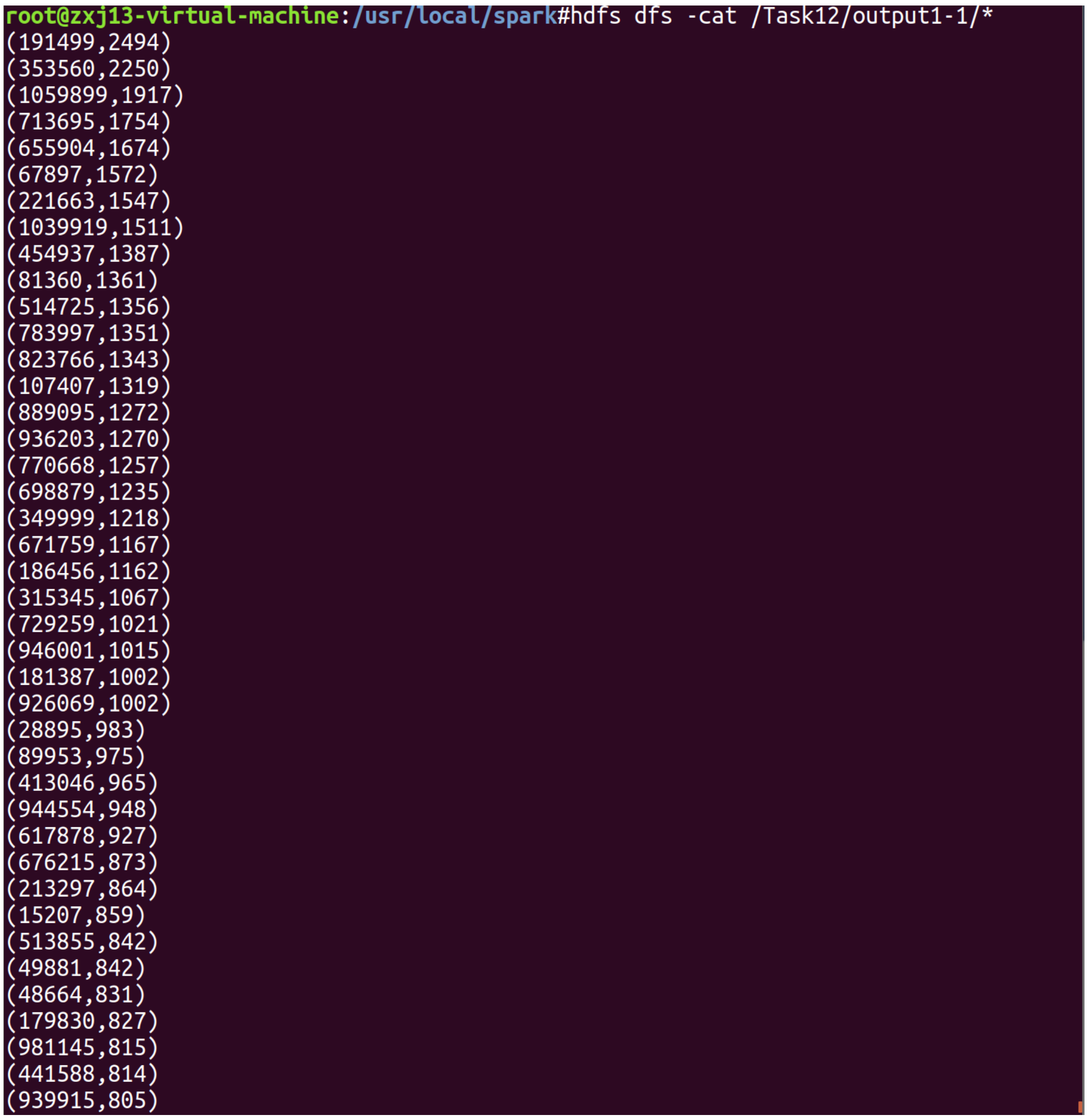
```
tar zxvf spark-3.0.1-bin-hadoop2.7.tgz  -C /usr/local/
cd /usr/local/
```

```
mv spark-3.0.1-bin-hadoop2.7 spark
cd /mnt/hgfs/ubuntu_share/
tar zxvf scala-2.12.11.tgz -C /usr/local/
cd /usr/local/
mv scala-2.12.11 scala
vim ~/.bashrc
# ~/.bashrc内容
#SCALA
export SCALA_HOME=/usr/local/scala
export PATH=$PATH:$HOME/bin:$JAVA_HOME/bin:$SCALA_HOME/bin
#SPARK
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
#localhost:8080
cd /usr/local/spark/conf
cp spark-env.sh.template spark-env.sh
vim spark-env.sh
#内容
export JAVA_HOME=/usr/java/jdk1.8.0_261
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SCALA_HOME=/usr/local/scala
export SPARK_HOME=/usr/local/spark
export SPARK_MASTER_IP=127.0.0.1
export SPARK_MASTER_PORT=7077
export SPARK_MASTER_WEBUI_PORT=8099
export SPARK_WORKER_CORES=3
export SPARK_WORKER_INSTANCES=1
export SPARK_WORKER_MEMORY=5G
export SPARK_WORKER_WEBUI_PORT=8081
export SPARK_EXECUTOR_CORES=1
export SPARK_EXECUTOR_MEMORY=1G
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HADOOP_HOME/lib/native
```

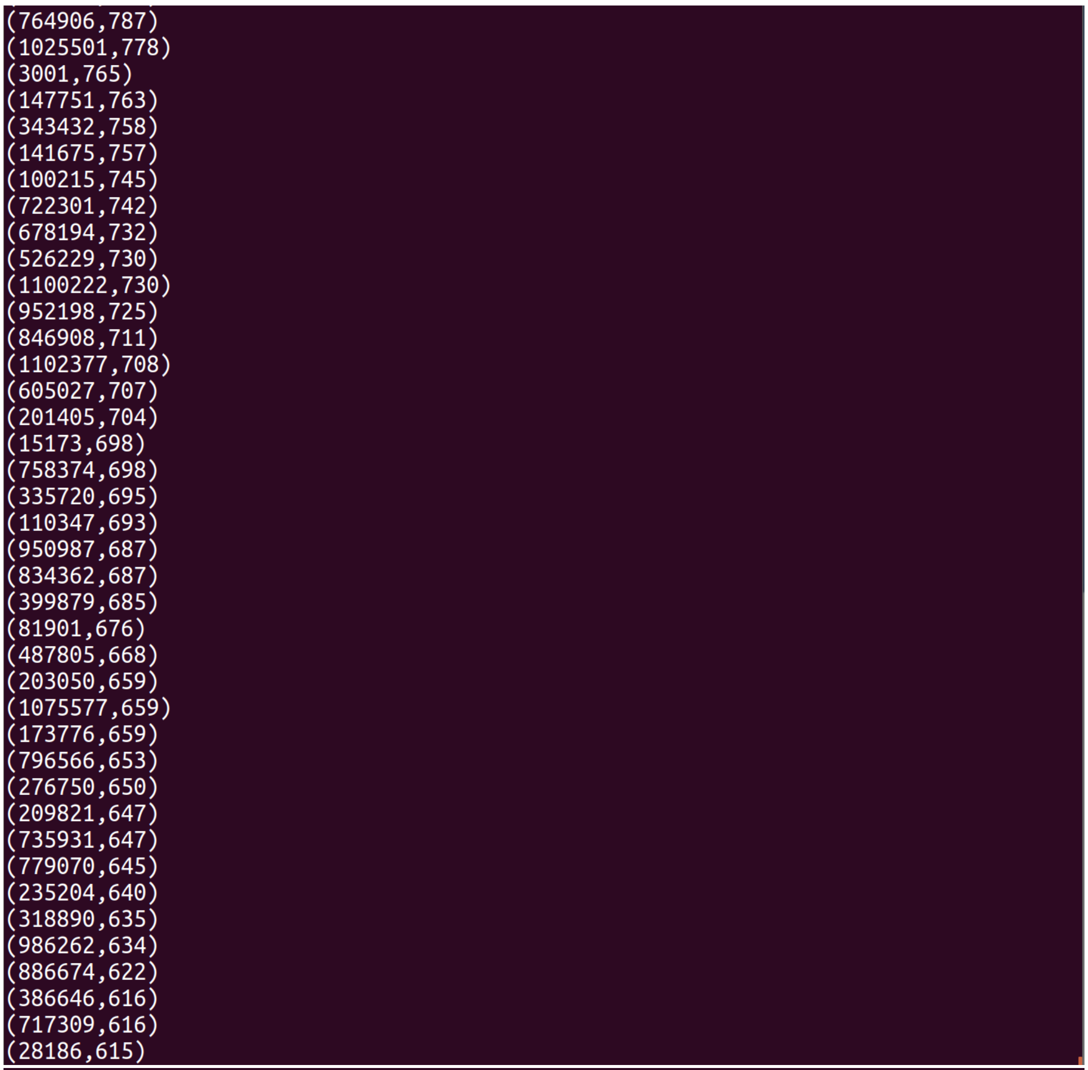
四、实验结果及截图

1. Task1截图

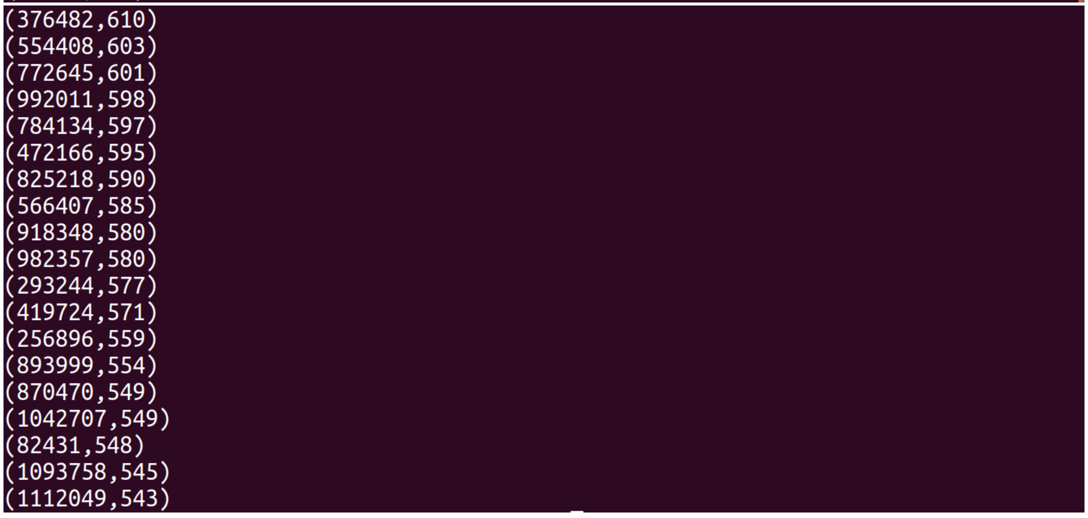
Task11截图



```
root@zxj13-virtual-machine:/usr/local/spark#hdfs dfs -cat /Task12/output1-1/*
(191499,2494)
(353560,2250)
(1059899,1917)
(713695,1754)
(655904,1674)
(67897,1572)
(221663,1547)
(1039919,1511)
(454937,1387)
(81360,1361)
(514725,1356)
(783997,1351)
(823766,1343)
(107407,1319)
(889095,1272)
(936203,1270)
(770668,1257)
(698879,1235)
(349999,1218)
(671759,1167)
(186456,1162)
(315345,1067)
(729259,1021)
(946001,1015)
(181387,1002)
(926069,1002)
(28895,983)
(89953,975)
(413046,965)
(944554,948)
(617878,927)
(676215,873)
(213297,864)
(15207,859)
(513855,842)
(49881,842)
(48664,831)
(179830,827)
(981145,815)
(441588,814)
(939915,805)
```

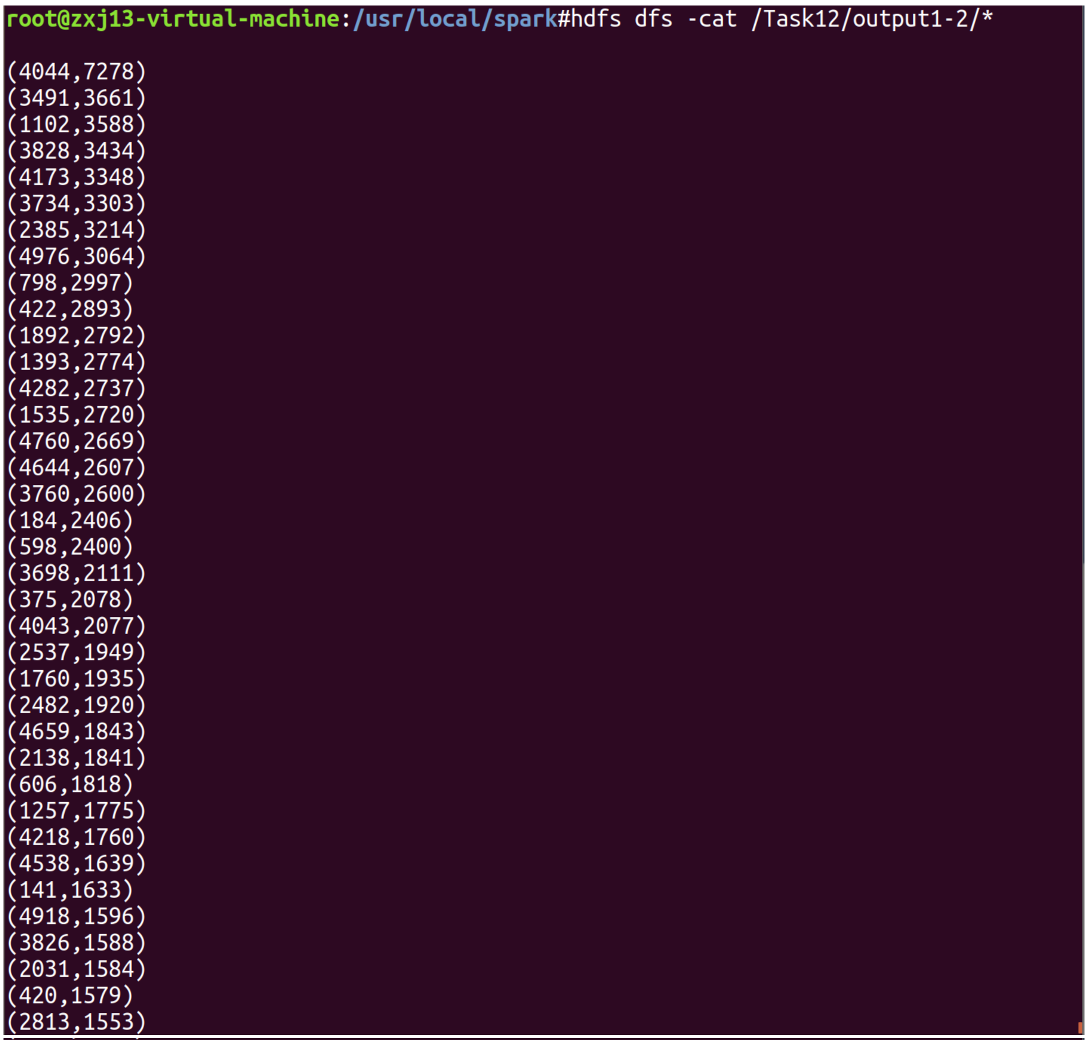


```
(764906,787)
(1025501,778)
(3001,765)
(147751,763)
(343432,758)
(141675,757)
(100215,745)
(722301,742)
(678194,732)
(526229,730)
(1100222,730)
(952198,725)
(846908,711)
(1102377,708)
(605027,707)
(201405,704)
(15173,698)
(758374,698)
(335720,695)
(110347,693)
(950987,687)
(834362,687)
(399879,685)
(81901,676)
(487805,668)
(203050,659)
(1075577,659)
(173776,659)
(796566,653)
(276750,650)
(209821,647)
(735931,647)
(779070,645)
(235204,640)
(318890,635)
(986262,634)
(886674,622)
(386646,616)
(717309,616)
(28186,615)
```

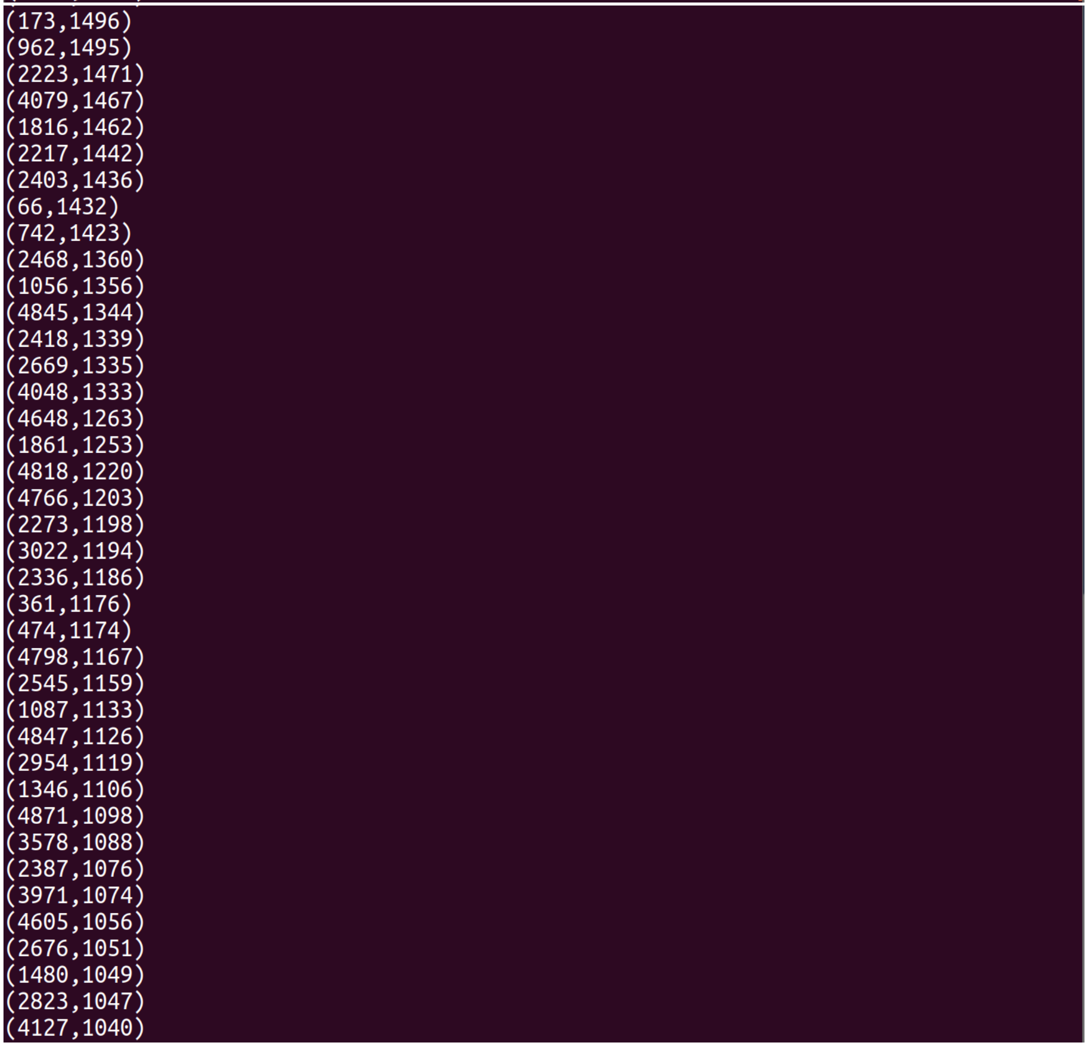


```
(376482,610)
(554408,603)
(772645,601)
(992011,598)
(784134,597)
(472166,595)
(825218,590)
(566407,585)
(918348,580)
(982357,580)
(293244,577)
(419724,571)
(256896,559)
(893999,554)
(870470,549)
(1042707,549)
(82431,548)
(1093758,545)
(1112049,543)
```

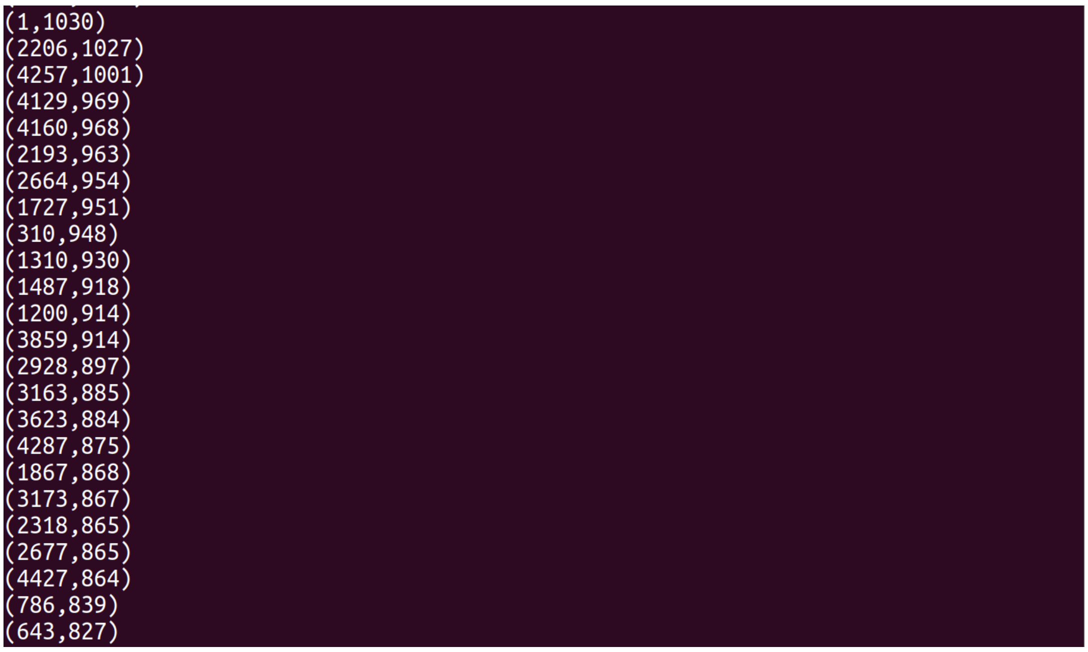
Task12截图



```
root@zxj13-virtual-machine:/usr/local/spark#hdfs dfs -cat /Task12/output1-2/*

(4044,7278)
(3491,3661)
(1102,3588)
(3828,3434)
(4173,3348)
(3734,3303)
(2385,3214)
(4976,3064)
(798,2997)
(422,2893)
(1892,2792)
(1393,2774)
(4282,2737)
(1535,2720)
(4760,2669)
(4644,2607)
(3760,2600)
(184,2406)
(598,2400)
(3698,2111)
(375,2078)
(4043,2077)
(2537,1949)
(1760,1935)
(2482,1920)
(4659,1843)
(2138,1841)
(606,1818)
(1257,1775)
(4218,1760)
(4538,1639)
(141,1633)
(4918,1596)
(3826,1588)
(2031,1584)
(420,1579)
(2813,1553)
```



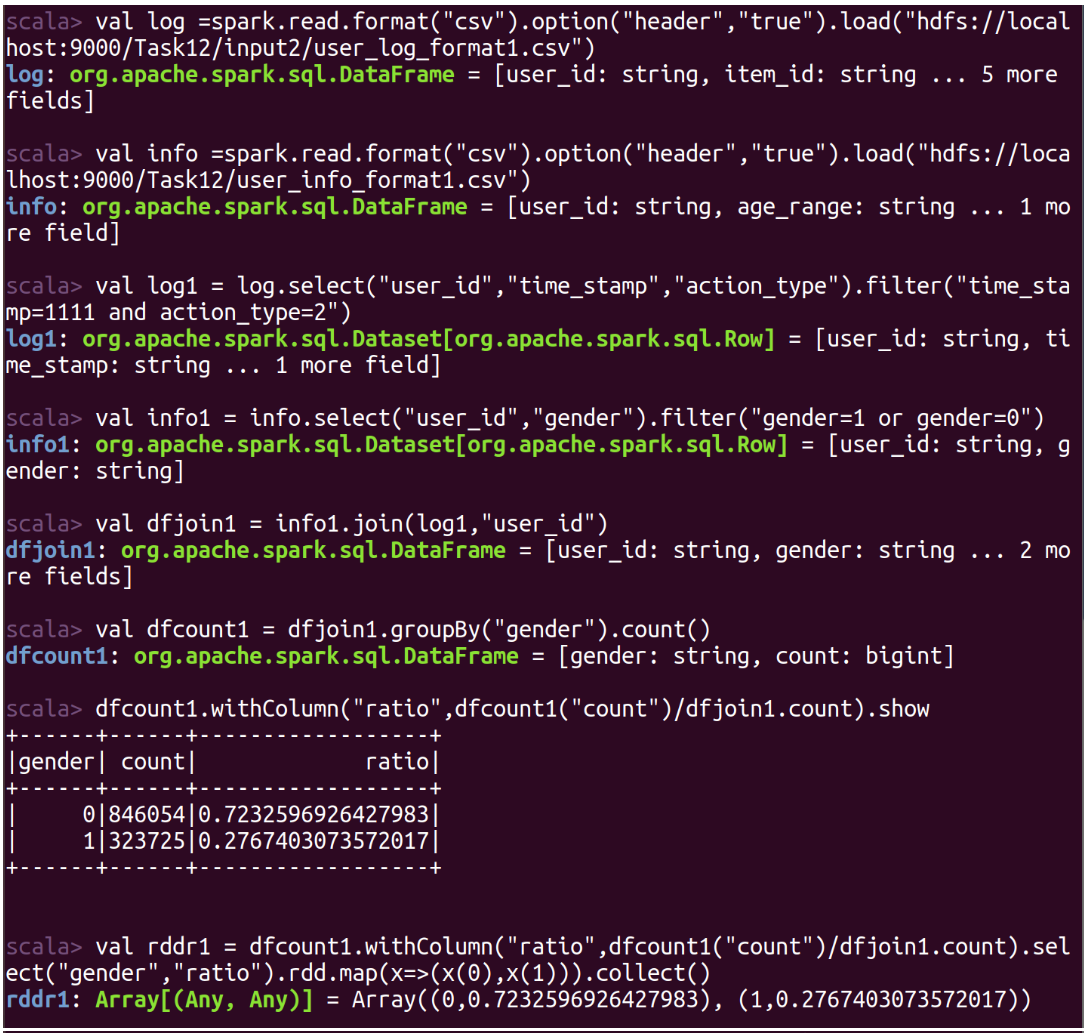
```
(173,1496)
(962,1495)
(2223,1471)
(4079,1467)
(1816,1462)
(2217,1442)
(2403,1436)
(66,1432)
(742,1423)
(2468,1360)
(1056,1356)
(4845,1344)
(2418,1339)
(2669,1335)
(4048,1333)
(4648,1263)
(1861,1253)
(4818,1220)
(4766,1203)
(2273,1198)
(3022,1194)
(2336,1186)
(361,1176)
(474,1174)
(4798,1167)
(2545,1159)
(1087,1133)
(4847,1126)
(2954,1119)
(1346,1106)
(4871,1098)
(3578,1088)
(2387,1076)
(3971,1074)
(4605,1056)
(2676,1051)
(1480,1049)
(2823,1047)
(4127,1040)
```



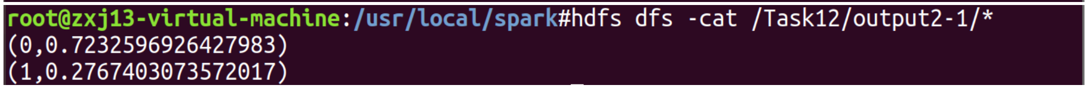
```
(1,1030)
(2206,1027)
(4257,1001)
(4129,969)
(4160,968)
(2193,963)
(2664,954)
(1727,951)
(310,948)
(1310,930)
(1487,918)
(1200,914)
(3859,914)
(2928,897)
(3163,885)
(3623,884)
(4287,875)
(1867,868)
(3173,867)
(2318,865)
(2677,865)
(4427,864)
(786,839)
(643,827)
```

2. Task2截图

Task21截图

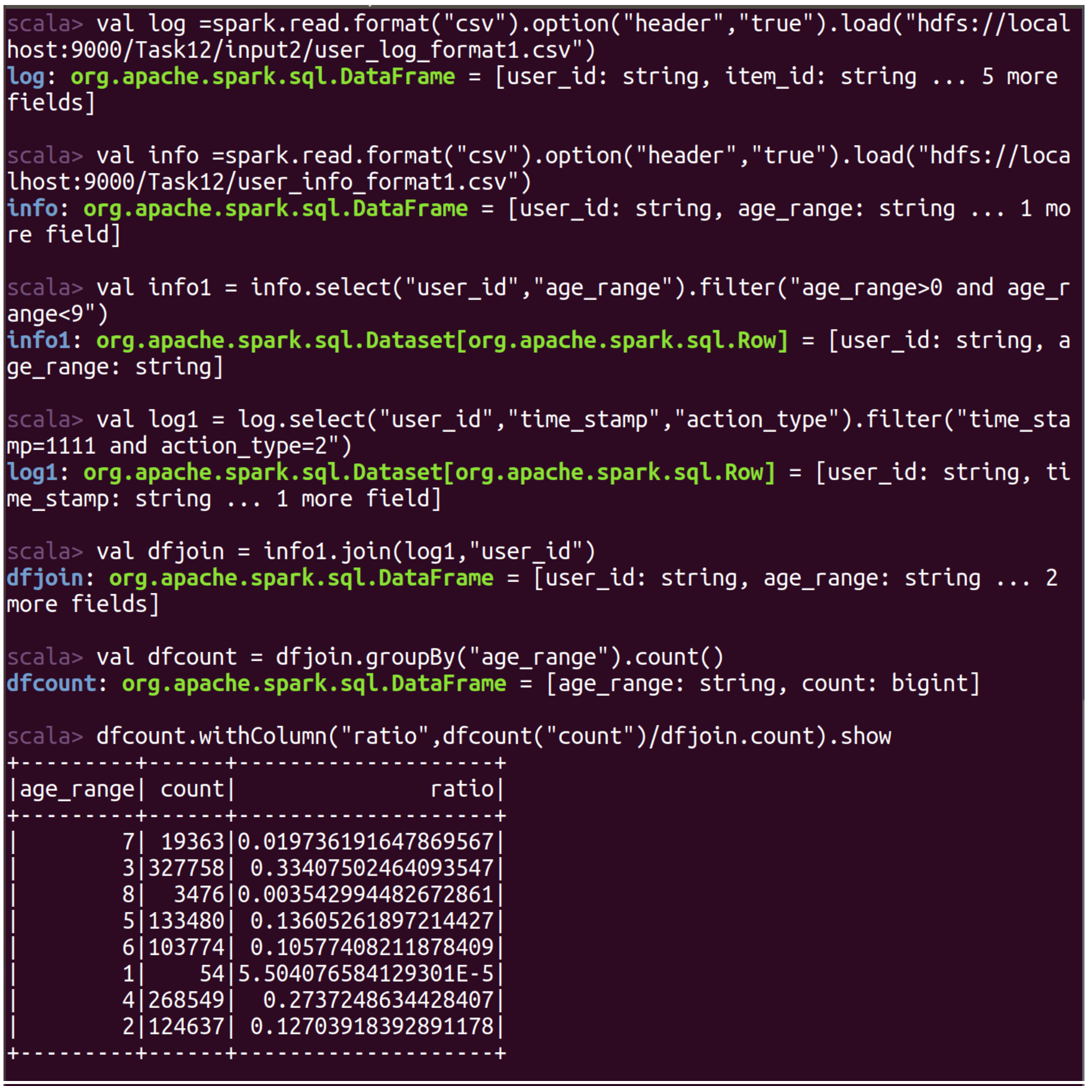


```
scala> val log =spark.read.format("csv").option("header","true").load("hdfs://local
host:9000/Task12/input2/user_log_format1.csv")
log: org.apache.spark.sql.DataFrame = [user_id: string, item_id: string ... 5 more
fields]

scala> val info =spark.read.format("csv").option("header","true").load("hdfs://loca
lhost:9000/Task12/user_info_format1.csv")
info: org.apache.spark.sql.DataFrame = [user_id: string, age_range: string ... 1 mo
re field]

scala> val log1 = log.select("user_id","time_stamp","action_type").filter("time_sta
mp=1111 and action_type=2")
log1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [user_id: string, ti
me_stamp: string ... 1 more field]

scala> val info1 = info.select("user_id","gender").filter("gender=1 or gender=0")
info1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [user_id: string, g
ender: string]

scala> val dfjoin1 = info1.join(log1,"user_id")
dfjoin1: org.apache.spark.sql.DataFrame = [user_id: string, gender: string ... 2 mo
re fields]

scala> val dfcount1 = dfjoin1.groupBy("gender").count()
dfcount1: org.apache.spark.sql.DataFrame = [gender: string, count: bigint]

scala> dfcount1.withColumn("ratio",dfcount1("count")/dfjoin1.count).show
+------+------+------------------+
|gender| count|             ratio|
+------+------+------------------+
|     0|846054|0.7232596926427983|
|     1|323725|0.2767403073572017|
+------+------+------------------+


scala> val rddr1 = dfcount1.withColumn("ratio",dfcount1("count")/dfjoin1.count).sel
ect("gender","ratio").rdd.map(x=>(x(0),x(1))).collect()
rddr1: Array[(Any, Any)] = Array((0,0.7232596926427983), (1,0.2767403073572017))
```

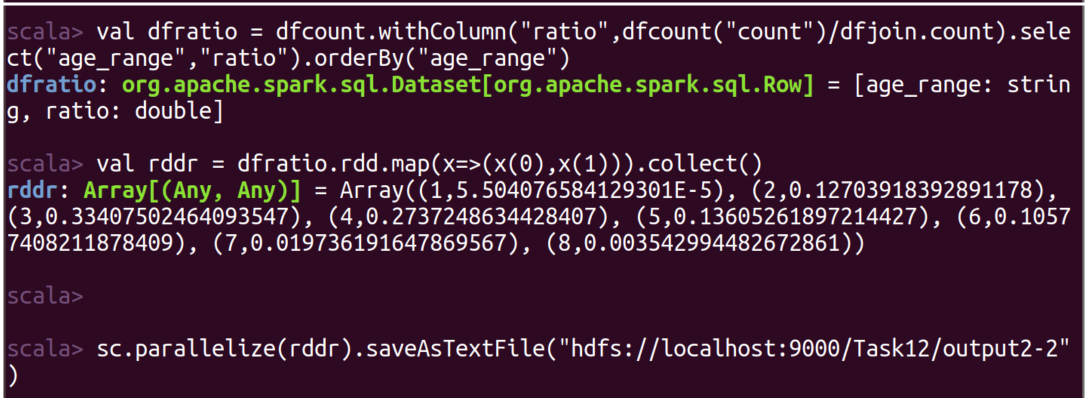


```
root@zxj13-virtual-machine:/usr/local/spark#hdfs dfs -cat /Task12/output2-1/*
(0,0.7232596926427983)
(1,0.2767403073572017)
```

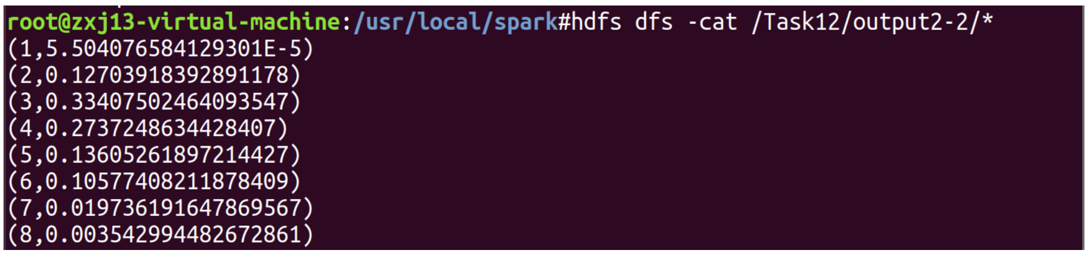
Task22截图



```
scala> val log =spark.read.format("csv").option("header","true").load("hdfs://local
host:9000/Task12/input2/user_log_format1.csv")
log: org.apache.spark.sql.DataFrame = [user_id: string, item_id: string ... 5 more
fields]

scala> val info =spark.read.format("csv").option("header","true").load("hdfs://loca
lhost:9000/Task12/user_info_format1.csv")
info: org.apache.spark.sql.DataFrame = [user_id: string, age_range: string ... 1 mo
re field]

scala> val info1 = info.select("user_id","age_range").filter("age_range>0 and age_r
ange<9")
info1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [user_id: string, a
ge_range: string]

scala> val log1 = log.select("user_id","time_stamp","action_type").filter("time_sta
mp=1111 and action_type=2")
log1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [user_id: string, ti
me_stamp: string ... 1 more field]

scala> val dfjoin = info1.join(log1,"user_id")
dfjoin: org.apache.spark.sql.DataFrame = [user_id: string, age_range: string ... 2
more fields]

scala> val dfcount = dfjoin.groupBy("age_range").count()
dfcount: org.apache.spark.sql.DataFrame = [age_range: string, count: bigint]

scala> dfcount.withColumn("ratio",dfcount("count")/dfjoin.count).show
+---------+------+--------------------+
|age_range| count|               ratio|
+---------+------+--------------------+
|        7| 19363|0.019736191647869567|
|        3|327758| 0.33407502464093547|
|        8|  3476|0.003542994482672861|
|        5|133480| 0.13605261897214427|
|        6|103774| 0.10577408211878409|
|        1|    54|5.504076584129301E-5|
|        4|268549|  0.2737248634428407|
|        2|124637| 0.12703918392891178|
+---------+------+--------------------+
```



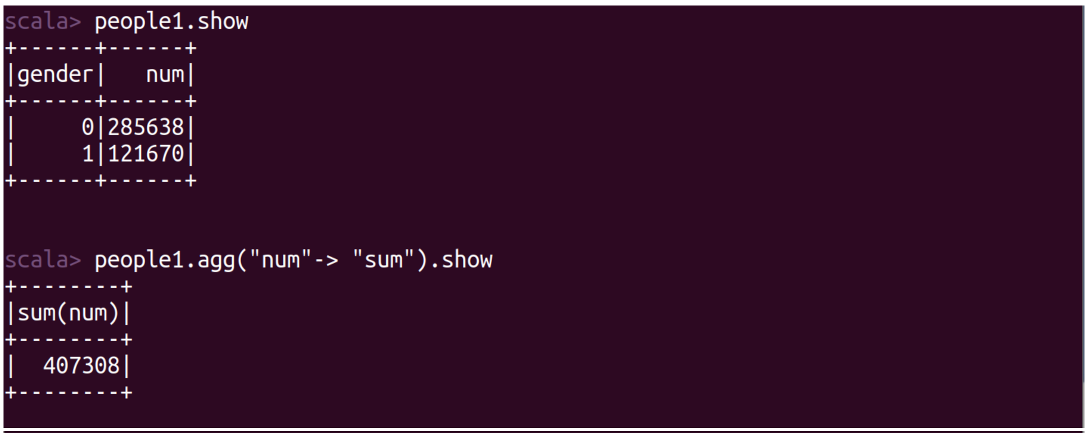
```
scala> val dfratio = dfcount.withColumn("ratio",dfcount("count")/dfjoin.count).sele
ct("age_range","ratio").orderBy("age_range")
dfratio: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [age_range: strin
g, ratio: double]

scala> val rddr = dfratio.rdd.map(x=>(x(0),x(1))).collect()
rddr: Array[(Any, Any)] = Array((1,5.504076584129301E-5), (2,0.12703918392891178),
(3,0.33407502464093547), (4,0.2737248634428407), (5,0.13605261897214427), (6,0.1057
7408211878409), (7,0.019736191647869567), (8,0.003542994482672861))

scala>

scala> sc.parallelize(rddr).saveAsTextFile("hdfs://localhost:9000/Task12/output2-2"
)
```



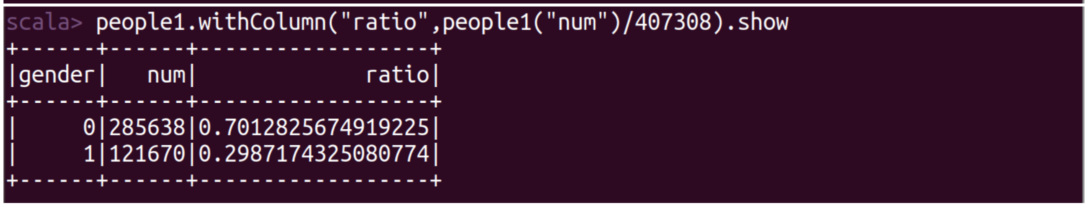
```
root@zxj13-virtual-machine:/usr/local/spark#hdfs dfs -cat /Task12/output2-2/*
(1,5.504076584129301E-5)
(2,0.12703918392891178)
(3,0.33407502464093547)
(4,0.2737248634428407)
(5,0.13605261897214427)
(6,0.10577408211878409)
(7,0.019736191647869567)
(8,0.003542994482672861)
```

3. Task3截图

Task31截图

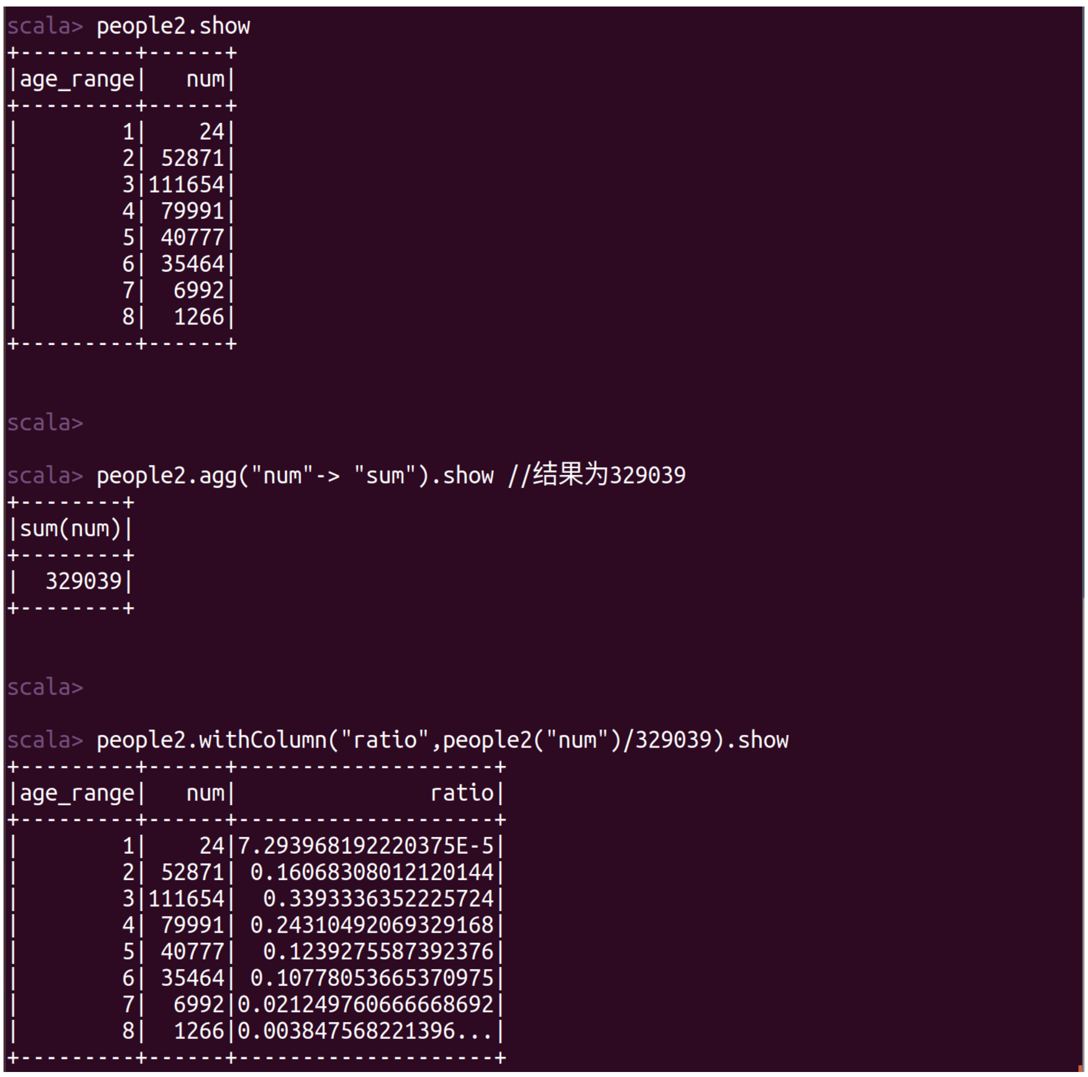


```
scala> people1.show
+------+------+
|gender|   num|
+------+------+
|     0|285638|
|     1|121670|
+------+------+


scala> people1.agg("num"-> "sum").show
+--------+
|sum(num)|
+--------+
|  407308|
+--------+
```



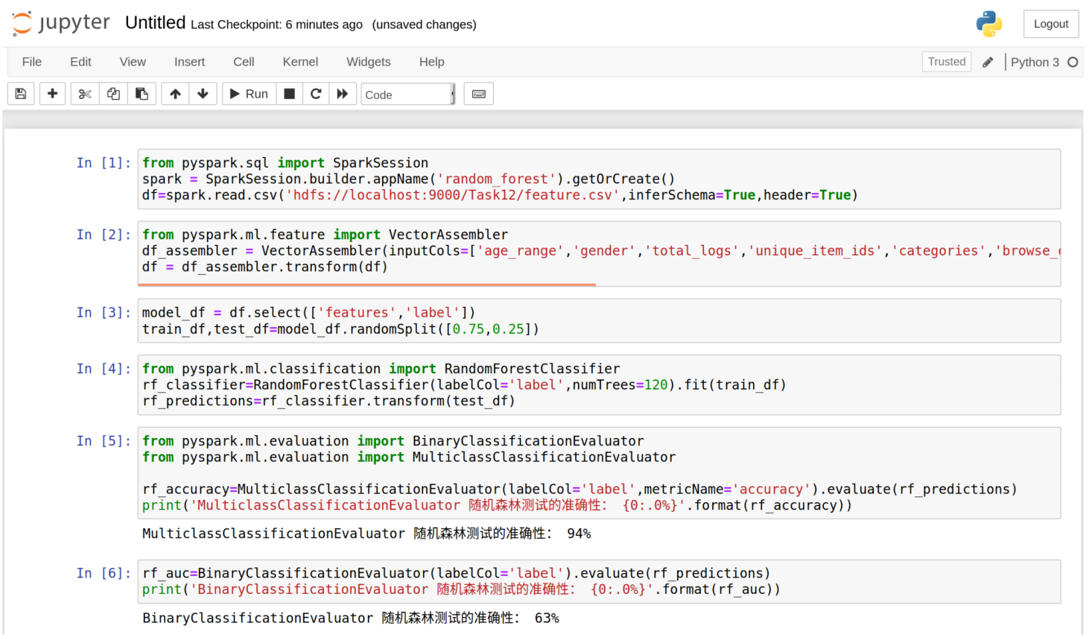
```
scala> people1.withColumn("ratio",people1("num")/407308).show
+------+------+------------------+
|gender|   num|             ratio|
+------+------+------------------+
|     0|285638|0.7012825674919225|
|     1|121670|0.2987174325080774|
+------+------+------------------+
```

Task32截图



```
scala> people2.show
+---------+------+
|age_range|   num|
+---------+------+
|        1|    24|
|        2| 52871|
|        3|111654|
|        4| 79991|
|        5| 40777|
|        6| 35464|
|        7|  6992|
|        8|  1266|
+---------+------+


scala>

scala> people2.agg("num"-> "sum").show //结果为329039
+--------+
|sum(num)|
+--------+
|  329039|
+--------+


scala>

scala> people2.withColumn("ratio",people2("num")/329039).show
+---------+------+--------------------+
|age_range|   num|               ratio|
+---------+------+--------------------+
|        1|    24|7.293968192220375E-5|
|        2| 52871| 0.16068308012120144|
|        3|111654|  0.3393336352225724|
|        4| 79991| 0.24310492069329168|
|        5| 40777|  0.1239275587392376|
|        6| 35464| 0.10778053665370975|
|        7|  6992|0.021249760666668692|
|        8|  1266|0.003847568221396...|
+---------+------+--------------------+
```

4. Task4截图



jupyter Untitled Last Checkpoint: 6 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: from pyspark.sql import SparkSession
        spark = SparkSession.builder.appName('random_forest').getOrCreate()
        df=spark.read.csv('hdfs://localhost:9000/Task12/feature.csv',inferSchema=True,header=True)

In [2]: from pyspark.ml.feature import VectorAssembler
        df_assembler = VectorAssembler(inputCols=['age_range','gender','total_logs','unique_item_ids','categories','browse_
        df = df_assembler.transform(df)

In [3]: model_df = df.select(['features','label'])
        train_df,test_df=model_df.randomSplit([0.75,0.25])

In [4]: from pyspark.ml.classification import RandomForestClassifier
        rf_classifier=RandomForestClassifier(labelCol='label',numTrees=120).fit(train_df)
        rf_predictions=rf_classifier.transform(test_df)

In [5]: from pyspark.ml.evaluation import BinaryClassificationEvaluator
        from pyspark.ml.evaluation import MulticlassClassificationEvaluator

        rf_accuracy=MulticlassClassificationEvaluator(labelCol='label',metricName='accuracy').evaluate(rf_predictions)
        print('MulticlassClassificationEvaluator 随机森林测试的准确性： {0:.0%}'.format(rf_accuracy))

        MulticlassClassificationEvaluator 随机森林测试的准确性： 94%

In [6]: rf_auc=BinaryClassificationEvaluator(labelCol='label').evaluate(rf_predictions)
        print('BinaryClassificationEvaluator 随机森林测试的准确性： {0:.0%}'.format(rf_auc))

        BinaryClassificationEvaluator 随机森林测试的准确性： 63%
```

五、问题总结

1. pyspark连接jupyter时出错
   解决方案：
   参考https://blog.csdn.net/hecongqing/article/details/85016154
   https://blog.csdn.net/donaldsy/article/details/96194346

   修改配置文件 `vim ~/.jupyter/jupyter_notebook_config.py`
   找到 `#c.NotebookApp.allow_root = False` ，去掉#，并修改为True

六、可能的改进之处

1. 第四题预测回头客没来得及用其他方法进行预测对比分析，后续有时间可以尝试别的预测方法