

# 211275010谢子骐 金融大数据实验4实验报告

代码文件位于ayer/exp4中，实验报告位于ayer/211275010谢子骐金融大数据实验4实验报告

## 1 task1

### 1.1统计贷款分布情况

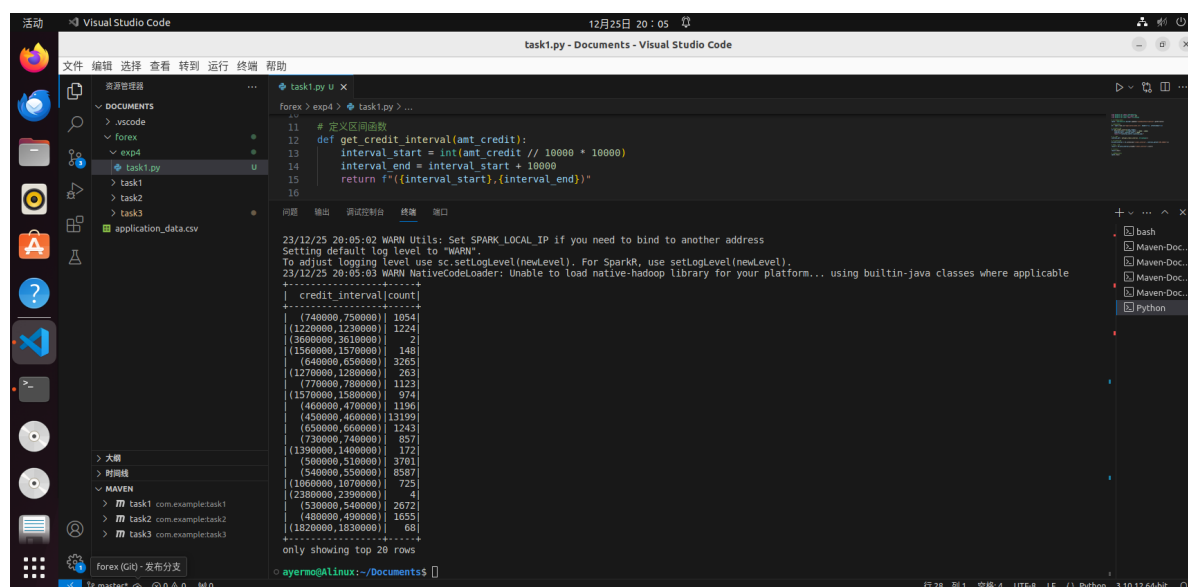
代码位于仓库ayer/exp4/task1.py

#### (1) 设计思路

首先初始化一个Spark会话。这是使用Spark进行任何操作的起点，方便我的程序与spark集群的交互。之后我用一个UDF来对这个application\_data.csv生成的dataframe的每一行进行分组和计数，UDF主要用来计算贷款金额所属的10000元区间。再之后用groupBy函数对计数结果进行分组，最后展示结果，并关闭spark。

#### (2) 实验结果

展示部分实验结果如下：



```
11 # 定义区间函数
12 def get_credit_interval(amt_credit):
13     interval_start = int(amt_credit // 10000 * 10000)
14     interval_end = interval_start + 10000
15     return f"({interval_start},{interval_end})"
16
23/12/25 20:05:02 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/12/25 20:05:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
+-----+
| credit_interval|count|
+-----+
| (740000,750000)| 1054|
| (1220000,1230000)| 1224|
| (1360000,1370000)| 2|
| (1560000,1570000)| 148|
| (640000,650000)| 3265|
| (1270000,1280000)| 263|
| (770000,780000)| 1123|
| (1570000,1580000)| 974|
| (460000,470000)| 1196|
| (450000,460000)| 13199|
| (650000,660000)| 1243|
| (730000,740000)| 837|
| (1390000,1400000)| 172|
| (500000,510000)| 3761|
| (540000,550000)| 8307|
| (1060000,1070000)| 725|
| (2380000,2390000)| 4|
| (530000,540000)| 2072|
| (480000,490000)| 1653|
| (1820000,1830000)| 68|
+-----+
only showing top 20 rows
```

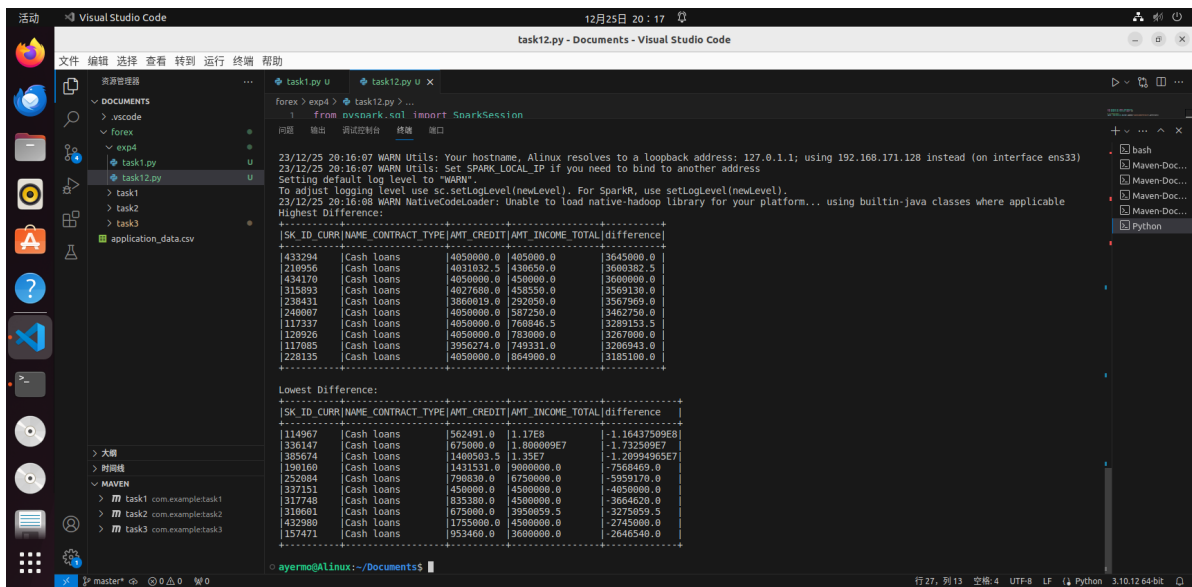
### 1.2 统计收入差值的最大最小前十

代码位于 ayer/exp4/task12.py

#### (1) 设计思路

同样按照上一问的思路开启spark的对话后，将csv文件导入为一个dataframe。之后利用df的withcolumn函数计算客户贷款金额AMT\_CREDIT与客户收入 AMT\_INCOME\_TOTAL两列间的差值。然后分别将计算的差值列按照升序和降序进行排序，分别取列的前十项，最后展示结果，关闭spark。

#### (2) 实验结果



## 2 task2

### 2.1 统计男性客户小孩个数的占比

代码位于ayer/exp4/task21.py

#### (1) 设计思路

同样先将csv文件导入为一个dataframe文件，之后创建spark sql查询语句对dataframe中每一行的数据进行计算和统计。“WHERE

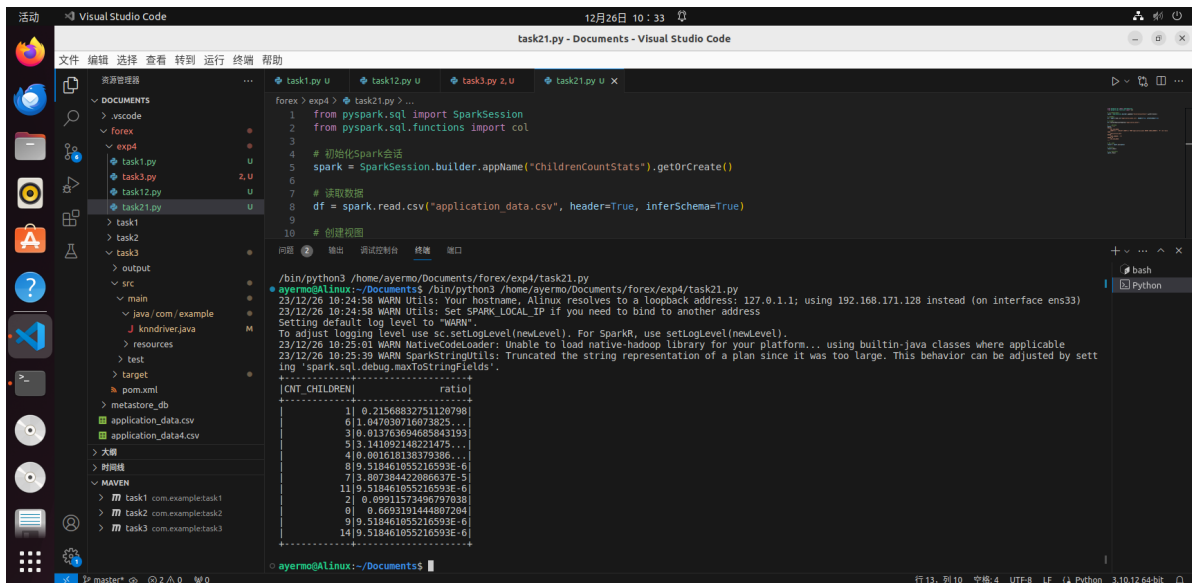
CODE\_GENDER = 'M'

GROUP BY

CNT\_CHILDREN”在语句中，用该语句来定位要计算的数据所在，再用“COUNT() / (SELECT COUNT())”计算相应占比，最后输出结果。

#### (2) 实验结果

由于数据过多，展示部分实验结果如下：



## 2.2 统计每日收入大于1的客户

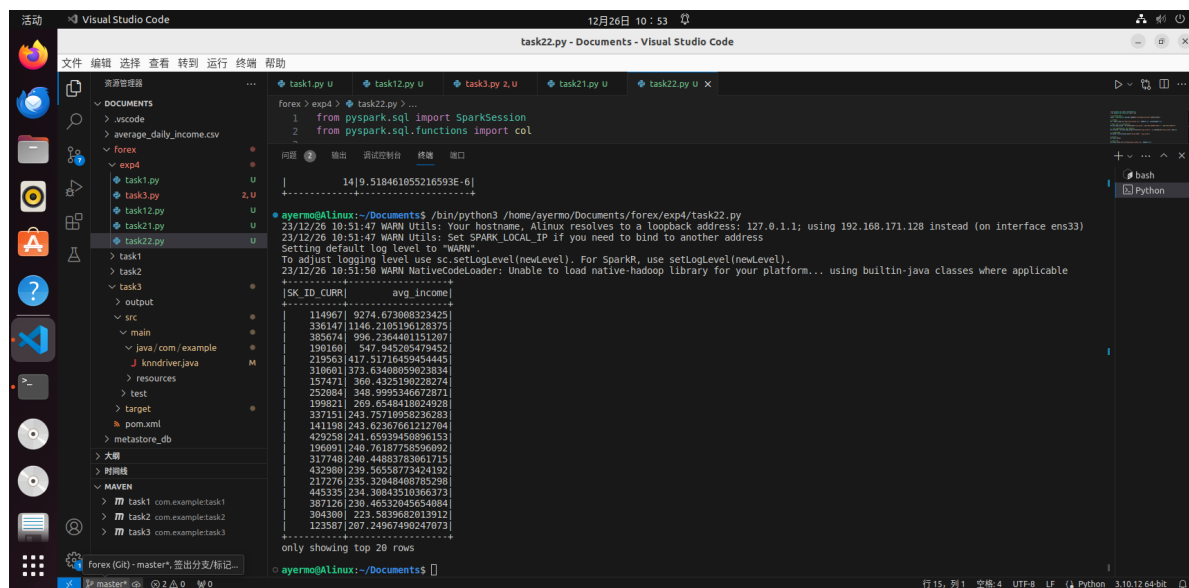
代码位于ayer/exp4/task22.py

### (1) 设计思路

同样先将相应csv文件导入为一个dataframe，之后对该dataframe进行操作即可。导入为df后，首先用“df\_with\_avg\_income = df.withColumn("avg\_income", col("AMT\_INCOME\_TOTAL") / - col("DAYS\_BIRTH"))”计算平均收入，之后再用“df\_filtered = df\_with\_avg\_income.filter(col("avg\_income") > 1).orderBy(col("avg\_income").desc())”即filter方法继续过滤，筛选出平均收入>1的用户。最后将相应的结果展示出来，因为仅仅需要展示,sk\_id\_curr和计算出来的收入结果列，所以在dataframe进行展示时进行筛选。

### (2) 实验结果

由于数据集过多，所以仅展示部分实验结果如下：



```
forex > exp4 > task22.py ...
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import col

14[9.518461055216593E-6]

ayermog@Linux:~/Documents$ /bin/python3 /home/ayermog/Documents/forex/exp4/task22.py
23/12/26 10:51:47 WARN Utils: Your hostname, Alinux resolves to a loopback address: 127.0.1.1; using 192.168.171.128 instead (on interface ens33)
23/12/26 10:51:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/12/26 10:51:50 WARN NativeCodeLoader: Unable to load native-heapdump library for your platform... using builtin-java classes where applicable
+-----+-----+
|SK_ID_CURR|avg_income|
+-----+-----+
|114967|9274.673008323425|
|336147|1146.2105196128375|
|385074|996.236440151207|
|190160|547.945205479452|
|219563|417.51716459454445|
|310601|373.63408059023034|
|157471|360.4325190228274|
|252084|348.9995346672871|
|109821|269.6548418024928|
|337151|243.75710950230283|
|141198|243.62367661212704|
|429258|241.65939450896153|
|106091|240.7618775096992|
|317748|240.44883783061715|
|432980|239.56558773424192|
|217276|239.32048408782298|
|445335|234.30843510366373|
|387126|238.46532045654084|
|304380|223.5836802013912|
|123587|207.24967490247073|
+-----+-----+
only showing top 20 rows
```

## 3 task3

根据给定的数据集，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。可以训练多个模型，比较模型的表现。

在task3的任务中，由于apache对spark ML库的维护较多，于是我选择用spark ML库来进行分类操作。在进行分类操作的方法中，因为在上次实验中我已经选用了KNN方法，所以在这次实验中，我选择用逻辑回归和决策树两种方法来进行回归预测。由于该实验与上次实验较为相似，上次实验的预测结果91%较好，所以在特征选择中，我选择和上次实验差不多的特征来进行预测分析。即feature\_columns = ['NAME\_CONTRACT\_TYPE','AMT\_INCOME\_TOTAL','AMT\_CREDIT','NAME\_INCOME\_TYPE','NAME\_EDUCATION\_TYPE','NAME\_FAMILY\_STATUS','FLAG\_CONT\_MOBILE','REGION\_RATING\_CLIENT','REG\_REGION\_NOT\_LIVE\_REGION','ORGANIZATION\_TYPE','OBS\_30\_CNT\_SOCIAL\_CIRCLE','DEF\_30\_CNT\_SOCIAL\_CIRCLE','OBS\_60\_CNT\_SOCIAL\_CIRCLE','DEF\_60\_CNT\_SOCIAL\_CIRCLE','FLAG\_DOCUMENT\_2','FLAG\_DOCUMENT\_3','FLAG\_DOCUMENT\_14']

同样，数据处理依然采用在上次实验二中处理过的数据集，即NAME\_INCOME\_TYPENAME\_EDUCATION\_TYPE、NAME\_FAMILY\_STATUS、NAME\_CONTRACT\_TYPE为string类型，无法量化比较，因此对其中的值的大小进行量化，因为是分类模型，所以对值的大小没有严谨的要求，故将其中string一样的值视作相同大小的int类型进行量化，int类型的值依次从0开始叠加。将这些数据处理后，同上述选取的其他数据一起，作为application4.csv，是这次模型训练的数据集来源。

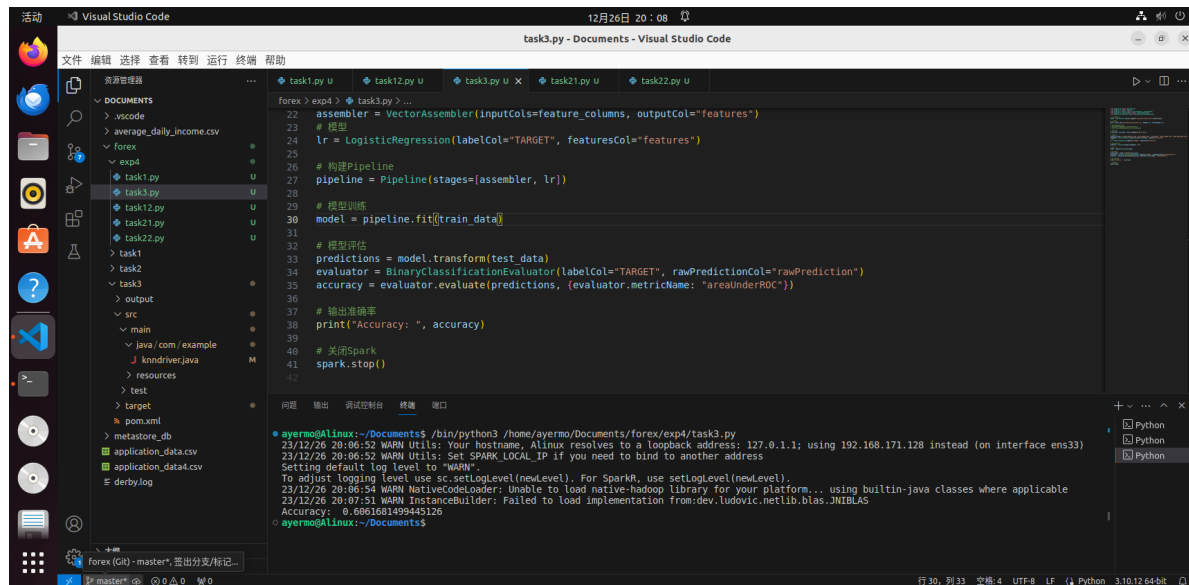
## 3.1 利用逻辑回归进行分类

代码位于ayer/exp4/task3.py

### (1) 设计思路

因为已经选取好了预测特征，所以在进行逻辑回归分析时，只需将这些特征应用于ML库中的逻辑回归方法，即LogisticRegression，将预测变量选为TARGET，特征变量选为feature\_columns中的各个特征。最后利用evaluate方法进行模型的准确性评估即可。

### (2) 实验结果



```
forex > exp4 > task3.py > ...
22 assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
23 # 模型
24 lr = LogisticRegression(labelCol="TARGET", featuresCol="features")
25
26 # 构建Pipeline
27 pipeline = Pipeline(stages=[assembler, lr])
28
29 # 模型训练
30 model = pipeline.fit(train_data)
31
32 # 模型评估
33 predictions = model.transform(test_data)
34 evaluator = BinaryClassificationEvaluator(labelCol="TARGET", rawPredictionCol="rawPrediction")
35 accuracy = evaluator.evaluate(predictions, (evaluator.metricName: "areaUnderROC"))
36
37 # 输出准确率
38 print("Accuracy: ", accuracy)
39
40 # 关闭Spark
41 spark.stop()
42
```

ayerno@Linux:~/Documents\$ /bin/python3 /home/ayerno/Documents/forex/exp4/task3.py  
23/12/26 20:06:52 WARN Utils: Your hostname, Alinux resolves to a loopback address: 127.0.1.1; using 192.168.171.128 instead (on interface ens33)  
23/12/26 20:06:52 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
23/12/26 20:06:54 WARN NativeCodeLoader: Unable to load native-heapoop library for your platform... using builtin-java classes where applicable  
23/12/26 20:07:51 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS  
Accuracy: 0.60618149945126  
ayerno@Linux:~/Documents\$

可以看到逻辑回归得到的准确率为60.6%。

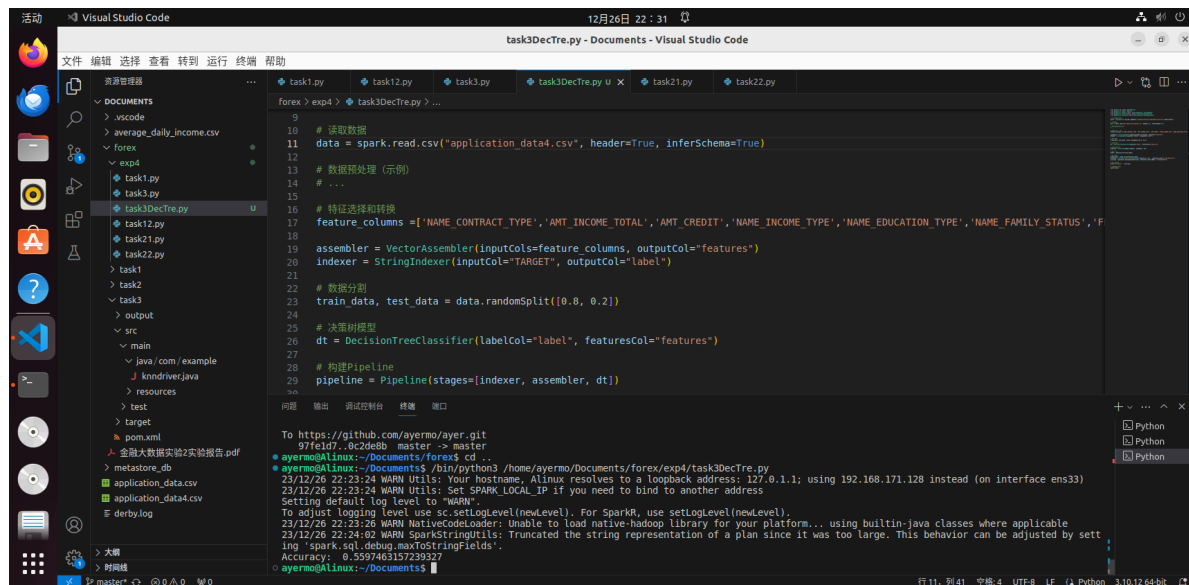
## 3.2 基于决策树的分类

代码位于ayer/exp4/task3DecTre.py

### (1) 设计思路

同样，基于已经选取好的特征，利用ML库中的DecisionTreeClassifier模型进行决策树模型构建，将预测变量选取为TARGET，之后，利用evaluate方法进行准确性评估，整体思路与基于逻辑回归的思路类似。

### (2) 实验结果



```
forex > exp4 > task3DecTre.py > ...
9
10 # 读取数据
11 data = spark.read.csv("application_data4.csv", header=True, inferSchema=True)
12
13 # 数据预处理 (示例)
14 # ...
15
16 # 特征选择和转换
17 feature_columns = ['NAME_CONTRACT_TYPE', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'F
18
19 assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
20 indexer = StringIndexer(inputCol="TARGET", outputCol="label")
21
22 # 数据分割
23 train_data, test_data = data.randomSplit([0.8, 0.2])
24
25 # 决策树模型
26 dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")
27
28 # 构建Pipeline
29 pipeline = Pipeline(stages=[indexer, assembler, dt])
30
```

To https://github.com/ayerno/ayer.git  
97fe1d7..0c2de8b master -> master  
ayerno@Linux:~/Documents/forex\$ cd ..  
ayerno@Linux:~/Documents\$ /bin/python3 /home/ayerno/Documents/forex/exp4/task3DecTre.py  
23/12/26 22:23:24 WARN Utils: Your hostname, Alinux resolves to a loopback address: 127.0.1.1; using 192.168.171.128 instead (on interface ens33)  
23/12/26 22:23:24 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
23/12/26 22:24:02 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by sett  
ing 'spark.sql.debug.maxToStringFields'.  
Accuracy: 0.559746315729327  
ayerno@Linux:~/Documents\$

可以看到基于决策树的回归结果准确率为55.97%

好吧综上所述，逻辑回归的效果要优于决策树的分类效果，（我以为决策树的效果能更优的www）。

我对这次实验的预测结果较为不理想的原因进行了分析，可能是由于之前选择的特征变量，较为适用于KNN算法，而对逻辑回归以及决策树算法较为不适用，所以导致这次实验的结果对比之前的实验二要差一些。