实验四 基于Spark的银行贷款违约预测

211275009 陈铭浩

1. 问题背景

在借贷交易中，银行和其他金融机构通常提供资金给借款人，期望借款人能够按时还款本金和利
息。然而，由于各种原因，有时借款人可能无法按照合同规定的方式履行还款义务，从而导致贷
款违约。本次实验以银行贷款违约为背景，选取了约30万条贷款信息，包含在application_data.csv文件中，数据描述包含在columns_description.csv文件夹中。
数据来源：https://www.kaggle.com/datasets/mishra5001/credit-card/data

---

2. 环境配置

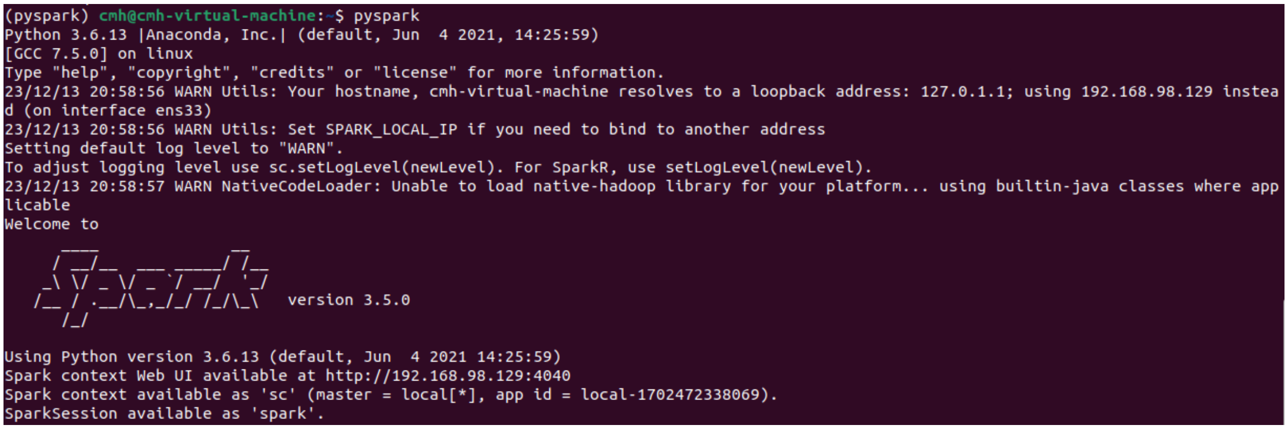
2.1 版本信息

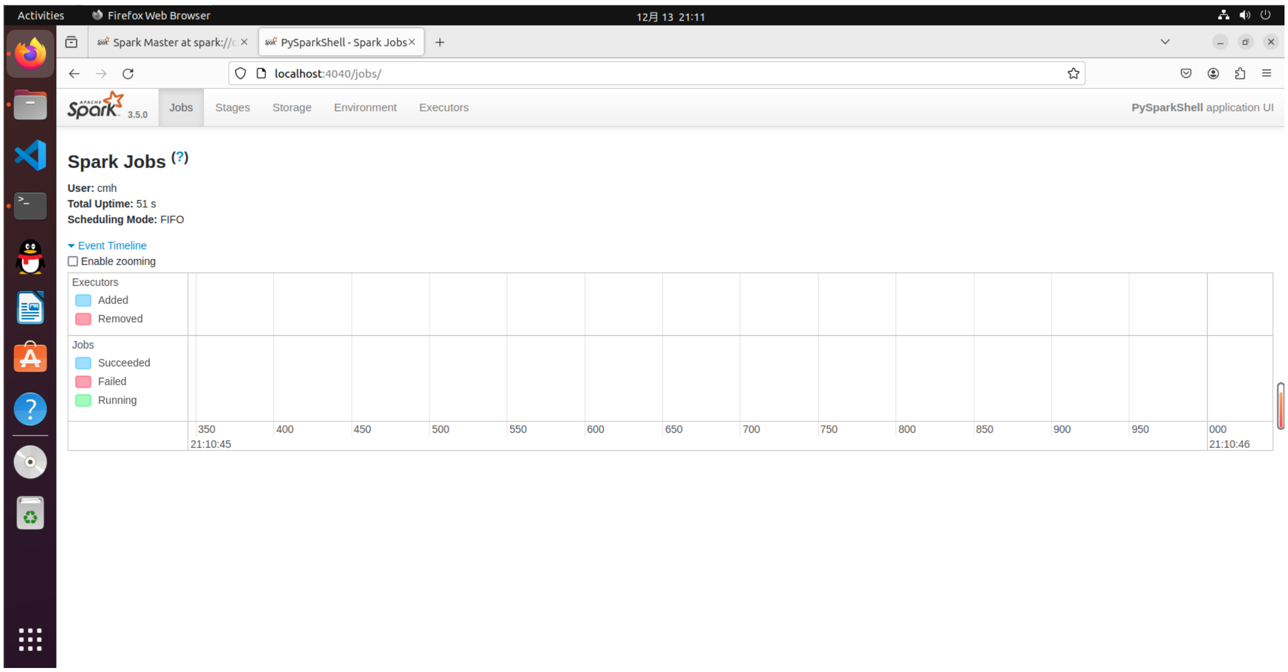
- 操作系统：LINUX（ubuntu-22.04.3）
- JAVA：java-8-openjdk-amd64（1.8.0_382）
- Hadoop：3.3.6
- Spark：3.5.0

2.2 Spark安装及配置

```
1. 安装并配置Anaconda环境
wget https://mirror.nju.edu.cn/anaconda/archive/Anaconda3-2021.11-Linux-x86_64.sh
sh ./Anaconda3-2021.11-Linux-x86_64.sh
```

```
conda create -n pyspark python=3.6  # 基于python3.6创建pyspark虚拟环境
conda activate pyspark      # 激活（切换）到pyspark虚拟环境
conda install pyspark
2. 安装Spark
wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
tar -zxvf spark-3.5.0-bin-hadoop3.tgz
mv spark-3.5.0-bin-hadoop3/ spark
3. spark相关配置
- 在/etc/bash.bashrc中添加：
export SPARK_HOME=/home/cmh/spark
export PATH=$SPARK_HOME/bin:$PATH
export PYSPARK_PYTHON=/home/cmh/anaconda3/envs/pyspark/bin/python
- 在spark/conf/spark-env.sh（该文件复制自spark-env.sh.template）中添加：
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export SPARK_DIST_CLASSPATH=$(/home/cmh/hadoop/hadoop-3.3.6/bin/hadoop classpath)
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.fs.logDirectory=hdfs://localhost:9000/spark-logs"
- 在spark/conf/spark-defaults.conf（该文件复制自spark-defaults.conf.template）中添加：
spark.eventLog.enabled           true
spark.eventLog.dir               hdfs://localhost:9000/spark-logs
- 创建hdfs://localhost:9000/spark-logs文件夹：
hdfs dfs -mkdir /spark-logs
4. 验证spark是否安装成功（终端环境）
pyspark
http://localhost:4040/
```

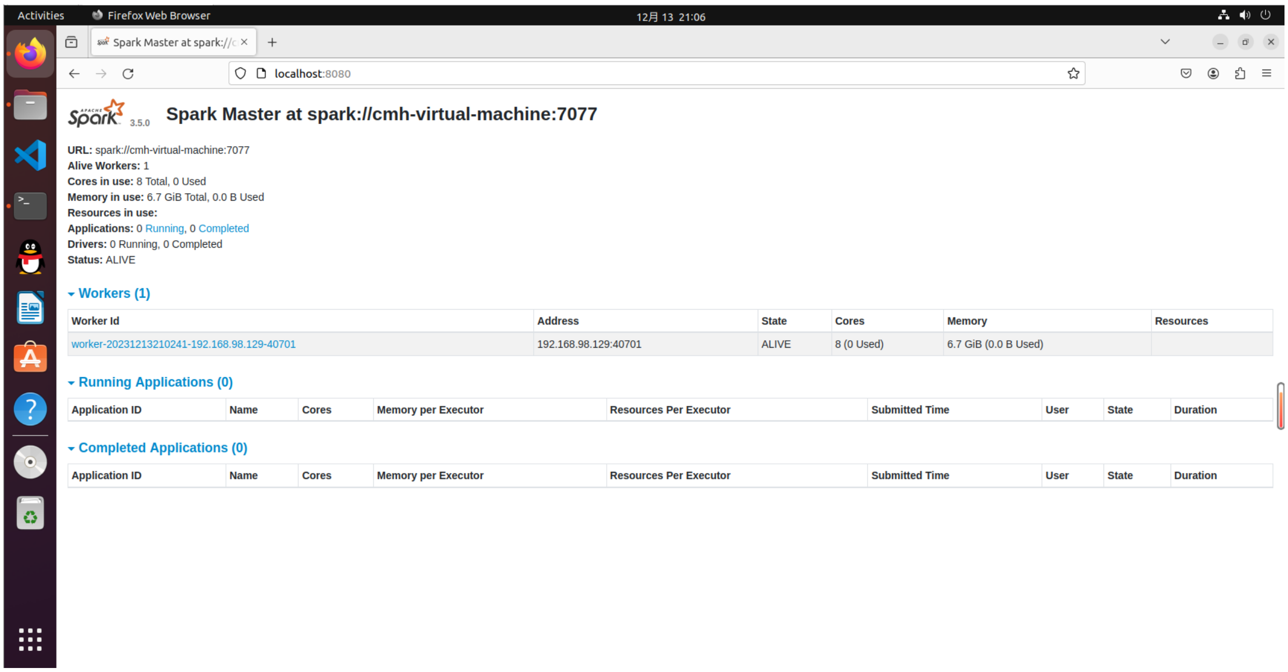


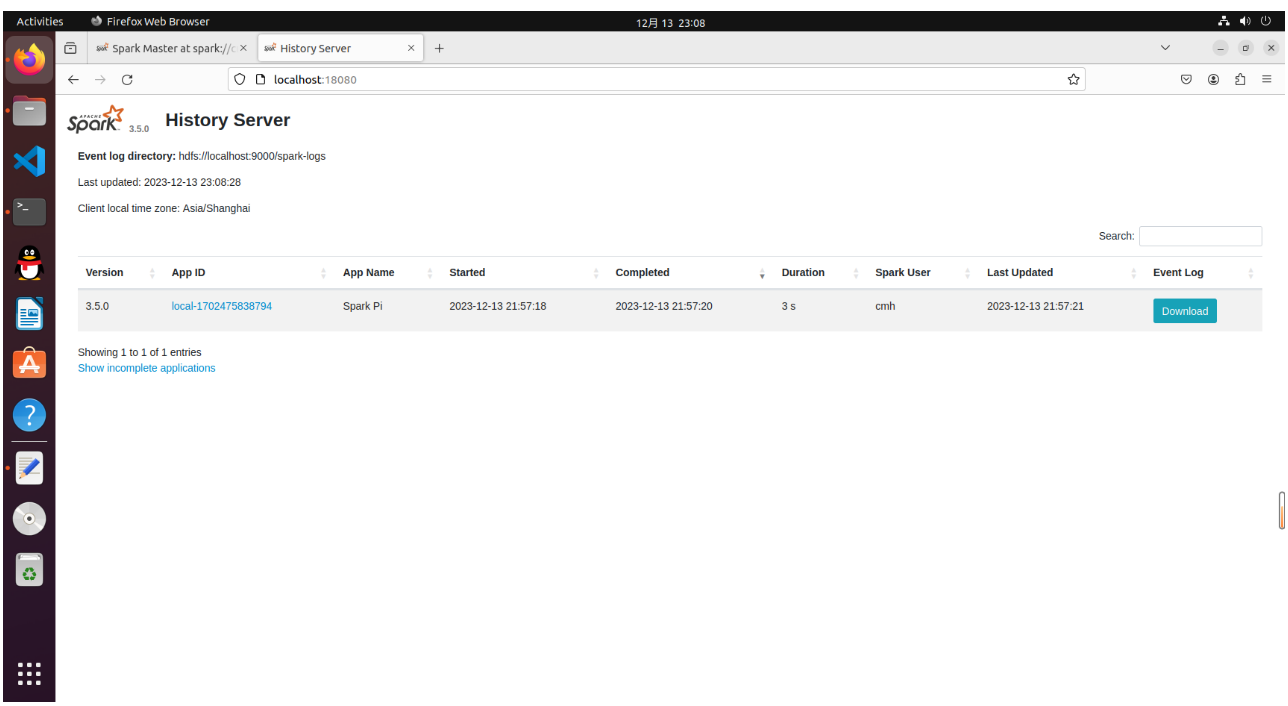


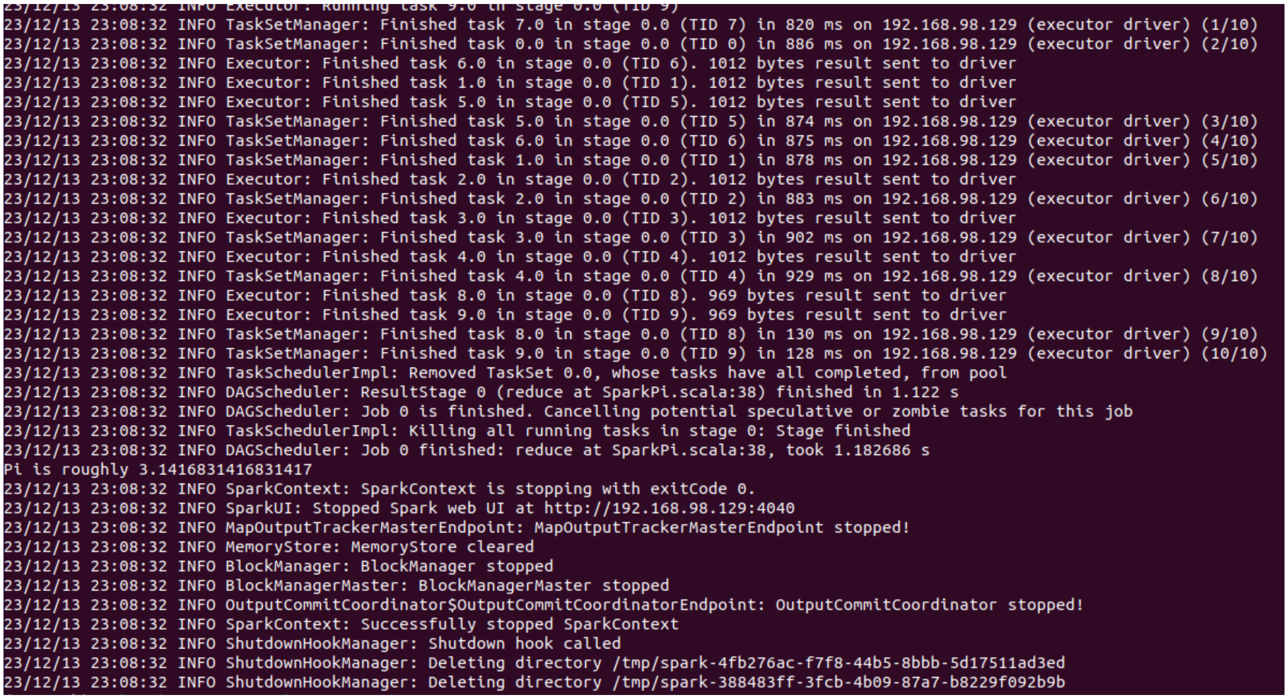

```
5. 程序测试
start-all.sh(开启Hadoop)
spark/sbin/start-all.sh（开启spark）(注意：这里直接使用start-all.sh会与hadoop的start-all冲突)
spark/sbin/start-history-server.sh
# spark master: http://localhost:8080/
# spark history server: http://localhost:8080/
http://localhost:18080/
run-example SparkPi 10
spark/sbin/start-history-server.sh
spark/sbin/stop-all.sh
```

```
(pyspark) cmh@cmh-virtual-machine:~$ pyspark
Python 3.6.13 |Anaconda, Inc.| (default, Jun  4 2021, 14:25:59)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
23/12/13 20:58:56 WARN Utils: Your hostname, cmh-virtual-machine resolves to a loopback address: 127.0.1.1; using 192.168.98.129 instea
d (on interface ens33)
23/12/13 20:58:56 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).
23/12/13 20:58:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where app
licable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Python version 3.6.13 (default, Jun  4 2021 14:25:59)
Spark context Web UI available at http://192.168.98.129:4040
Spark context available as 'sc' (master = local[*], app id = local-1702472338069).
SparkSession available as 'spark'.
```







```
23/12/13 23:08:32 INFO Executor: Running task 9.0 in stage 0.0 (TID 9)
23/12/13 23:08:32 INFO TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 820 ms on 192.168.98.129 (executor driver) (1/10)
23/12/13 23:08:32 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 886 ms on 192.168.98.129 (executor driver) (2/10)
23/12/13 23:08:32 INFO Executor: Finished task 6.0 in stage 0.0 (TID 6). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO Executor: Finished task 5.0 in stage 0.0 (TID 5). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 874 ms on 192.168.98.129 (executor driver) (3/10)
23/12/13 23:08:32 INFO TaskSetManager: Finished task 6.0 in stage 0.0 (TID 6) in 875 ms on 192.168.98.129 (executor driver) (4/10)
23/12/13 23:08:32 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 878 ms on 192.168.98.129 (executor driver) (5/10)
23/12/13 23:08:32 INFO Executor: Finished task 2.0 in stage 0.0 (TID 2). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 883 ms on 192.168.98.129 (executor driver) (6/10)
23/12/13 23:08:32 INFO Executor: Finished task 3.0 in stage 0.0 (TID 3). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 902 ms on 192.168.98.129 (executor driver) (7/10)
23/12/13 23:08:32 INFO Executor: Finished task 4.0 in stage 0.0 (TID 4). 1012 bytes result sent to driver
23/12/13 23:08:32 INFO TaskSetManager: Finished task 4.0 in stage 0.0 (TID 4) in 929 ms on 192.168.98.129 (executor driver) (8/10)
23/12/13 23:08:32 INFO Executor: Finished task 8.0 in stage 0.0 (TID 8). 969 bytes result sent to driver
23/12/13 23:08:32 INFO Executor: Finished task 9.0 in stage 0.0 (TID 9). 969 bytes result sent to driver
23/12/13 23:08:32 INFO TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 130 ms on 192.168.98.129 (executor driver) (9/10)
23/12/13 23:08:32 INFO TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 128 ms on 192.168.98.129 (executor driver) (10/10)
23/12/13 23:08:32 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
23/12/13 23:08:32 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 1.122 s
23/12/13 23:08:32 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
23/12/13 23:08:32 INFO TaskSchedulerImpl: Killing all running tasks in stage 0: Stage finished
23/12/13 23:08:32 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1.182686 s
Pi is roughly 3.1416831416831417
23/12/13 23:08:32 INFO SparkContext: SparkContext is stopping with exitCode 0.
23/12/13 23:08:32 INFO SparkUI: Stopped Spark web UI at http://192.168.98.129:4040
23/12/13 23:08:32 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
23/12/13 23:08:32 INFO MemoryStore: MemoryStore cleared
23/12/13 23:08:32 INFO BlockManager: BlockManager stopped
23/12/13 23:08:32 INFO BlockManagerMaster: BlockManagerMaster stopped
23/12/13 23:08:32 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
23/12/13 23:08:32 INFO SparkContext: Successfully stopped SparkContext
23/12/13 23:08:32 INFO ShutdownHookManager: Shutdown hook called
23/12/13 23:08:32 INFO ShutdownHookManager: Deleting directory /tmp/spark-4fb276ac-f7f8-44b5-8bbb-5d17511ad3ed
23/12/13 23:08:32 INFO ShutdownHookManager: Deleting directory /tmp/spark-388483ff-3fcb-4b09-87a7-b8229f092b9b
```

> *运行pyspark程序：*
> *spark-submit [options] [app arguments]*

3.任务一

1. 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。以 10000 元为区间进行输出。输出格式示例：
   ((20000,30000),1234)
   表示20000到30000元之间（包括20000元，但不包括30000元）有1234条记录。

- 设计思路：通过遍历每个区间，将AMT_CREDIT值分配到相应的区间中，然后统计每个区间的记录数量，最终按照区间范围格式化输出。使用groupBy和count方法统计每个区间的记录数量。

- 运行结果：

```
+------------------+------------+
|bin_range         |record_count|
+------------------+------------+
|{100000, 110000}  |1871        |
|{1000000, 1010000}|2587        |
|{1010000, 1020000}|656         |
|{1020000, 1030000}|1438        |
|{1030000, 1040000}|868         |
|{1040000, 1050000}|1463        |
|{1050000, 1060000}|905         |
|{1060000, 1070000}|725         |
|{1070000, 1080000}|3252        |
|{1080000, 1090000}|760         |
|{1090000, 1100000}|768         |
|{110000, 120000}  |1930        |
|{1100000, 1110000}|421         |
|{1110000, 1120000}|1009        |
|{1120000, 1130000}|4199        |
|{1130000, 1140000}|492         |
|{1140000, 1150000}|301         |
|{1150000, 1160000}|379         |
|{1160000, 1170000}|365         |
|{1170000, 1180000}|474         |
+------------------+------------+
only showing top 20 rows
```

---

2. 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。差值=AMT_CREDIT-AMT_INCOME_TOTAL。输出格式：
   <SK_ID_CURR><NAME_CONTRACT_TYPE><AMT_CREDIT><AMT_INCOME_TOTAL>, <差值>

- 设计思路：在df中添加新列“dif”，计算出差值并填入，然后分别按照差值升序和降序排列df，从而得到差值最高和最低的各十条记录。
- 运行结果：

```
+----------+------------------+----------+----------------+---------+
|SK_ID_CURR|NAME_CONTRACT_TYPE|AMT_CREDIT|AMT_INCOME_TOTAL|      dif|
+----------+------------------+----------+----------------+---------+
|    433294|        Cash loans| 4050000.0|        405000.0|3645000.0|
|    210956|        Cash loans| 4031032.5|        430650.0|3600382.5|
|    434170|        Cash loans| 4050000.0|        450000.0|3600000.0|
|    315893|        Cash loans| 4027680.0|        458550.0|3569130.0|
|    238431|        Cash loans| 3860019.0|        292050.0|3567969.0|
|    240007|        Cash loans| 4050000.0|        587250.0|3462750.0|
|    117337|        Cash loans| 4050000.0|        760846.5|3289153.5|
|    120926|        Cash loans| 4050000.0|        783000.0|3267000.0|
|    117085|        Cash loans| 3956274.0|        749331.0|3206943.0|
|    228135|        Cash loans| 4050000.0|        864900.0|3185100.0|
+----------+------------------+----------+----------------+---------+
```

```
+----------+------------------+----------+----------------+-------------+
|SK_ID_CURR|NAME_CONTRACT_TYPE|AMT_CREDIT|AMT_INCOME_TOTAL|          dif|
+----------+------------------+----------+----------------+-------------+
|    114967|        Cash loans|  562491.0|          1.17E8|-1.16437509E8|
|    336147|        Cash loans|  675000.0|      1.800009E7|  -1.732509E7|
|    385674|        Cash loans| 1400503.5|          1.35E7|-1.20994965E7|
|    190160|        Cash loans| 1431531.0|       9000000.0|   -7568469.0|
|    252084|        Cash loans|  790830.0|       6750000.0|   -5959170.0|
|    337151|        Cash loans|  450000.0|       4500000.0|   -4050000.0|
|    317748|        Cash loans|  835380.0|       4500000.0|   -3664620.0|
|    310601|        Cash loans|  675000.0|       3950059.5|   -3275059.5|
|    432980|        Cash loans| 1755000.0|       4500000.0|   -2745000.0|
|    157471|        Cash loans|  953460.0|       3600000.0|   -2646540.0|
+----------+------------------+----------+----------------+-------------+
```

4.任务二

基于Hive或者Spark SQL对application_data.csv进行如下统计：

1. 统计所有男性客户（CODE_GENDER=M）的小孩个数（CNT_CHILDREN）类型占比情况。
   输出格式为：<CNT_CHILDREN>，<类型占比>
   例：0，0.1234　表示没有小孩的男性客户占总男性客户数量的占比为0.1234。

- 设计思路：首先筛选出男性客户，然后统计小孩个数类型占比，并按照占比降序排序，最后输出结果。
- 运行结果：

```
+------------+--------------------+
|CNT_CHILDREN|             percent|
+------------+--------------------+
|           0|  0.6693191444807204|
|           1| 0.21568832751120798|
|           2| 0.09911573496797038|
|           3|0.013763694685843193|
|           4|0.001618138379386...|
|           5|3.141092148221475...|
|           6|1.047030716073825...|
|           7|3.807384422086637E-5|
|           8|9.518461055216593E-6|
|          11|9.518461055216593E-6|
|          14|9.518461055216593E-6|
|           9|9.518461055216593E-6|
+------------+--------------------+
```

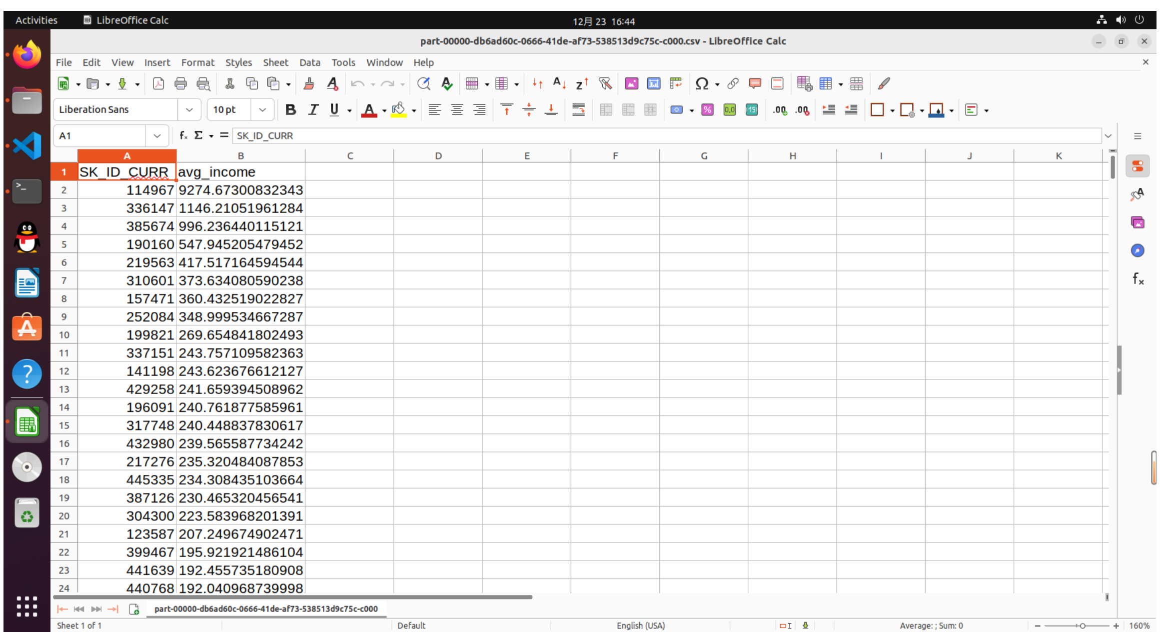
---

2. 统计每个客户出生以来每天的平均收入（avg_income）=总收入（AMT_INCOME_TOTAL）/
   出生天数（DAYS_BIRTH)，统计每日收入大于1的客户，并按照从大到小排序，保存为csv。

输出格式：<SK_ID_CURR>, <avg_income>

- 设计思路：为df新增一列avg_income，通过公式计算并填入，这里需要把出生天数取绝对值，然后筛选avg_income大于1的客户，再按照从大到小排序，最后保存为csv文件。
- 运行结果：

```
+----------+-----------------+
|SK_ID_CURR|       avg_income|
+----------+-----------------+
|    114967| 9274.673008323425|
|    336147|1146.2105196128375|
|    385674| 996.2364401151207|
|    190160|  547.945205479452|
|    219563|417.51716459454445|
|    310601|373.63408059023834|
|    157471| 360.4325190228274|
|    252084| 348.9995346672871|
|    199821| 269.6548418024928|
|    337151|243.75710958236283|
|    141198|243.62367661212704|
|    429258|241.65939450896153|
|    196091|240.76187758596092|
|    317748|240.44883783061715|
|    432980|239.56558773424192|
|    217276|235.32048408785298|
|    445335|234.30843510366373|
|    387126|230.46532045654084|
|    304300| 223.5839682013912|
|    123587|207.24967490247073|
+----------+-----------------+
only showing top 20 rows
```



5.任务三

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。
说明：
1、该任务可视为一个“二分类”任务，因为数据集只存在两种情况，违约（Class=1）和其他（Class=0）。
2、可根据时间特征的先后顺序按照8：2的比例将数据集application_data.csv拆分成训练集和测
试集，时间小的为训练集，其余为测试集；也可以按照8：2的比例随机拆分数据集。最后评估模
型的性能，评估指标可以为accuracy、f1-score等。
3、基于数据集application_data.csv，可以自由选择特征属性的组合，自行选用一种或多种分类
算法对目标属性TARGET进行预测。

- 设计思路：在实验二中数据清洗后得到的train.csv和test.csv数据基础上进行模型训练与评估，对类别型变量进行独热编码处理，而后分别构建logistic模型和随机森林模型，构建pipeline流水线进行模型的训练和评估，最后计算出accuracy
- 运行结果：

```
Logistic Regression Model Accuracy: 0.6708912078965459
```

```
Random Forest Model Accuracy: 0.6690078136635321
```