南京大学 2023 秋 FBDP 实验报告

姓名	康钊源
学号	211275007
实验名称	实验四：用 Spark 进行违约贷款预测
实验时间	2023/12/10-2023/12/23

一、实验目标

1. 安装并配置 spark 环境
2. 熟悉在 vscode 中用 pyspark 模块编程
3. 用 pyspark 解决违约贷款预测问题

二、实验内容

（1）安装并配置 spark

首先要安装 python 环境，参考以下步骤

首先在官网下载/Python-3.7.2.tgz

解压目录：

```
tar zxvf /root/download/Python-3.7.2.tgz -C /root/download/
```

进入解压后目录：

```
cd Python-3.7.2
```

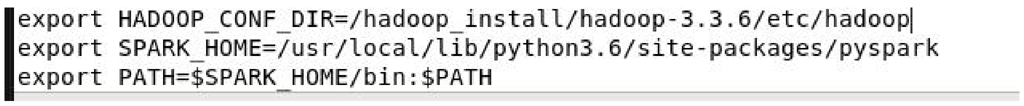
手动编译：

```
./configure prefix=/usr/local/python3
make && make install
```

之后下载 pyspark 包

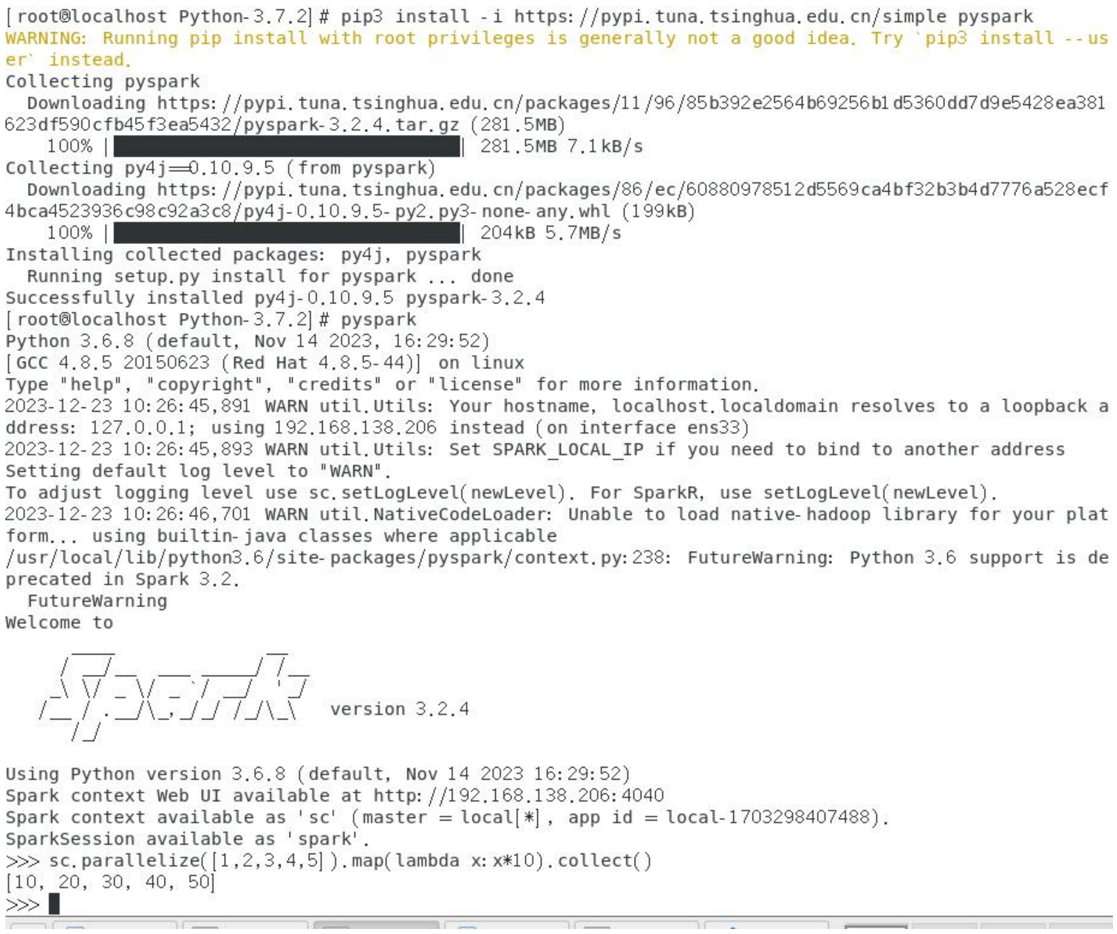
```
pip3 install pyspark
```

安装后需要配置一下 SPARK_HOME



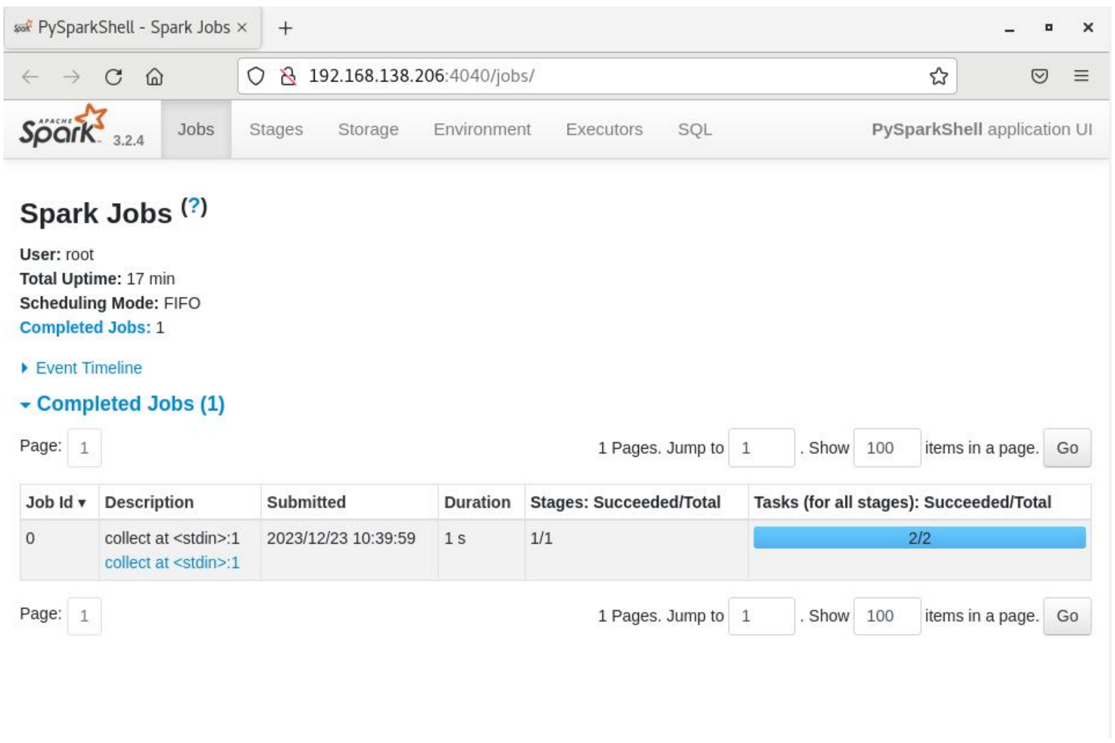
```
export HADOOP_CONF_DIR=/hadoop_install/hadoop-3.3.6/etc/hadoop
export SPARK_HOME=/usr/local/lib/python3.6/site-packages/pyspark
export PATH=$SPARK_HOME/bin:$PATH
```

之后就可以用 pyspark 命令启动 spark shell 了



```
[root@localhost Python-3.7.2]# pip3 install -i https://pypi.tuna.tsinghua.edu.cn/simple pyspark
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --us
er` instead.
Collecting pyspark
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/11/96/85b392e2564b69256b1d5360dd7d9e5428ea381
623df590cfb45f3ea5432/pyspark-3.2.4.tar.gz (281.5MB)
    100% |████████████████████████████████| 281.5MB 7.1kB/s
Collecting py4j==0.10.9.5 (from pyspark)
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/86/ec/60880978512d5569ca4bf32b3b4d7776a528ecf
4bca4523936c98c92a3c8/py4j-0.10.9.5-py2.py3-none-any.whl (199kB)
    100% |████████████████████████████████| 204kB 5.7MB/s
Installing collected packages: py4j, pyspark
  Running setup.py install for pyspark ... done
Successfully installed py4j-0.10.9.5 pyspark-3.2.4
[root@localhost Python-3.7.2]# pyspark
Python 3.6.8 (default, Nov 14 2023, 16:29:52)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
2023-12-23 10:26:45,891 WARN util.Utils: Your hostname, localhost.localdomain resolves to a loopback a
ddress: 127.0.0.1; using 192.168.138.206 instead (on interface ens33)
2023-12-23 10:26:45,893 WARN util.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).
2023-12-23 10:26:46,701 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
/usr/local/lib/python3.6/site-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is de
precated in Spark 3.2.
  FutureWarning
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.4
      /_/

Using Python version 3.6.8 (default, Nov 14 2023 16:29:52)
Spark context Web UI available at http://192.168.138.206:4040
Spark context available as 'sc' (master = local[*], app id = local-1703298407488).
SparkSession available as 'spark'.
>>> sc.parallelize([1,2,3,4,5]).map(lambda x: x*10).collect()
[10, 20, 30, 40, 50]
>>> 
```

还可以在 4040 端口查看 spark 的 WEB 端


PySparkShell - Spark Jobs
192.168.138.206:4040/jobs/
Spark 3.2.4
Jobs
Stages
Storage
Environment
Executors
SQL
PySparkShell application UI

Spark Jobs (?)

User: root
Total Uptime: 17 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

▾ **Completed Jobs (1)**

Page: 1   1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <stdin>:1 collect at <stdin>:1	2023/12/23 10:39:59	1 s	1/1	2/2

Page: 1   1 Pages. Jump to 1 . Show 100 items in a page. Go

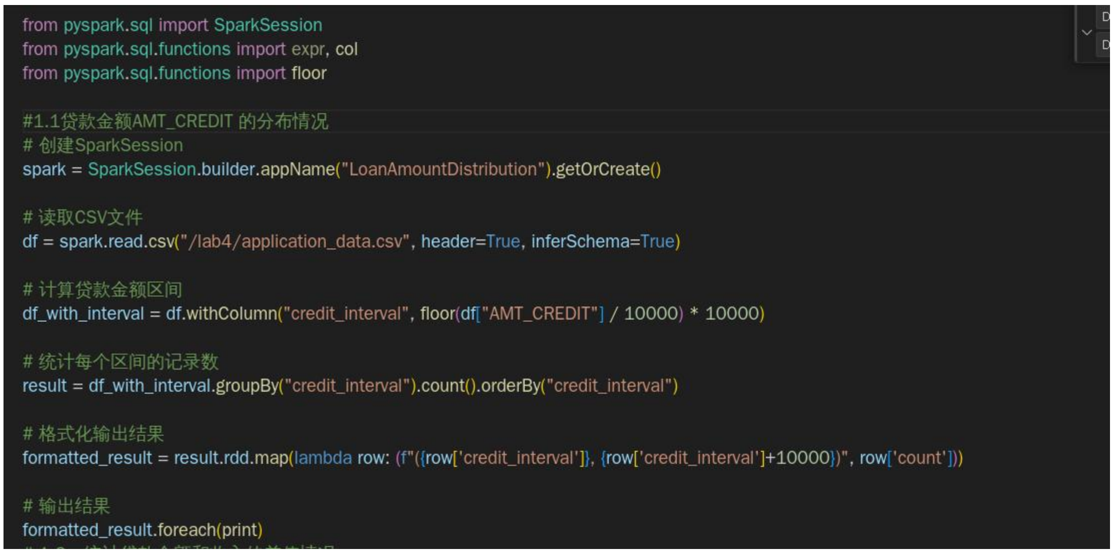
（2）任务一（程序运行结果均在第三部分）

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。

首先将原数据按 AMT_CREDIT 每隔 10000 元分割区间
接着用 groupby+count 统计每个区间内的记录数
最后用要求的格式输出



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr, col
from pyspark.sql.functions import floor

#1.1贷款金额AMT_CREDIT 的分布情况
# 创建SparkSession
spark = SparkSession.builder.appName("LoanAmountDistribution").getOrCreate()

# 读取CSV文件
df = spark.read.csv("/lab4/application_data.csv", header=True, inferSchema=True)

# 计算贷款金额区间
df_with_interval = df.withColumn("credit_interval", floor(df["AMT_CREDIT"] / 10000) * 10000)

# 统计每个区间的记录数
result = df_with_interval.groupBy("credit_interval").count().orderBy("credit_interval")

# 格式化输出结果
formatted_result = result.rdd.map(lambda row: (f"({row['credit_interval']}, {row['credit_interval']+10000})", row['count']))

# 输出结果
formatted_result.foreach(print)
```

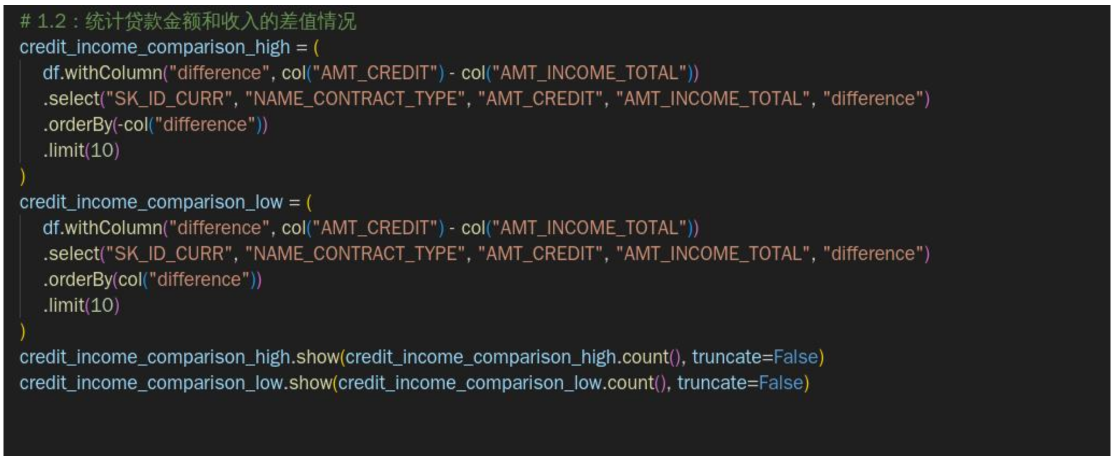
2、编写 Spark 程序，统计 application_data.csv 中客户贷款金额 **AMT_CREDIT** 比客户收入 **AMT_INCOME_TOTAL** 差值最高和最低的各十条记录。

输出格式：

<SK_ID_CURR><NAME_CONTRACT_TYPE><AMT_CREDIT><AMT_INCOME_TOTAL>, <差值>

差值=AMT_CREDIT-AMT_INCOME_TOTAL

添加新列 difference = AMT_CREDIT - AMT_INCOME_TOTAL
分别按 difference 值逆序和顺序输出前十条记录的对应列即可



```python
# 1.2：统计贷款金额和收入的差值情况
credit_income_comparison_high = (
    df.withColumn("difference", col("AMT_CREDIT") - col("AMT_INCOME_TOTAL"))
    .select("SK_ID_CURR", "NAME_CONTRACT_TYPE", "AMT_CREDIT", "AMT_INCOME_TOTAL", "difference")
    .orderBy(-col("difference"))
    .limit(10)
)
credit_income_comparison_low = (
    df.withColumn("difference", col("AMT_CREDIT") - col("AMT_INCOME_TOTAL"))
    .select("SK_ID_CURR", "NAME_CONTRACT_TYPE", "AMT_CREDIT", "AMT_INCOME_TOTAL", "difference")
    .orderBy(col("difference"))
    .limit(10)
)
credit_income_comparison_high.show(credit_income_comparison_high.count(), truncate=False)
credit_income_comparison_low.show(credit_income_comparison_low.count(), truncate=False)
```

（3）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。

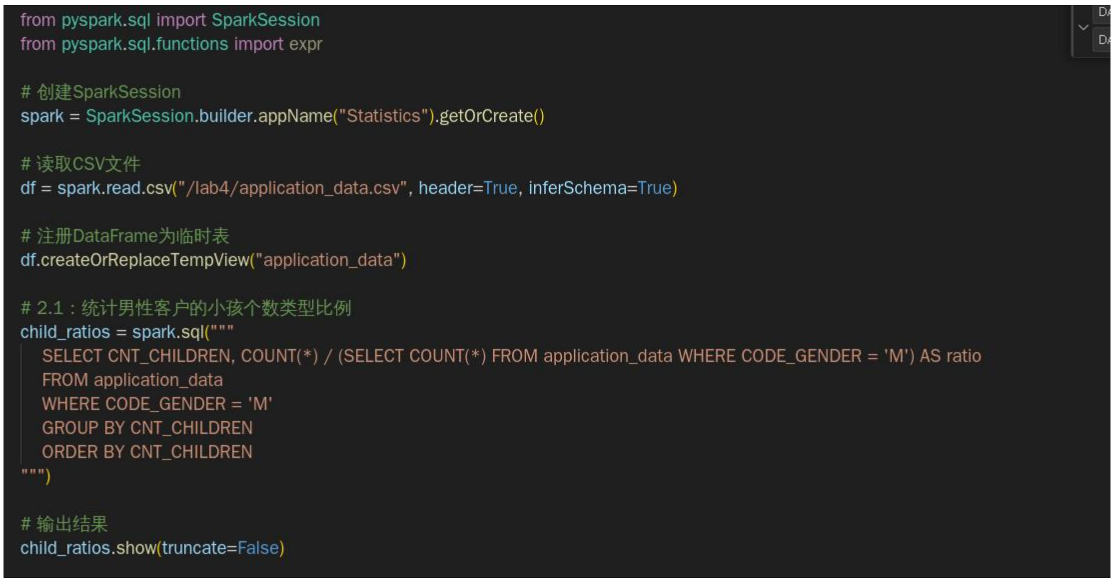
输出格式为：<CNT_CHILDREN>，<类型占比>

例：

0，0.1234

表示没有小小孩的男性客户占总男性客户数量的占比为 0.1234。

选择使用 pyspark 自带的 spark sql 完成任务



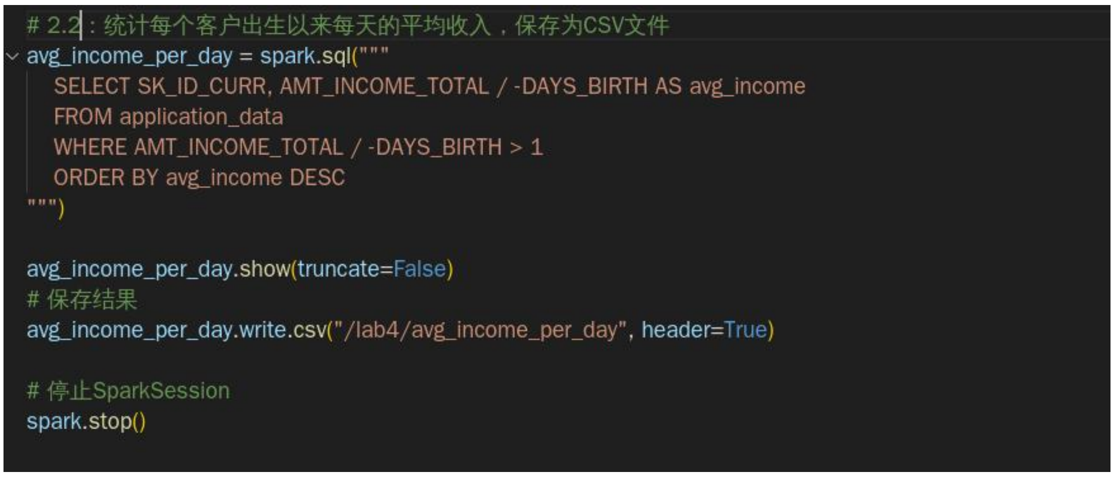
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr

# 创建SparkSession
spark = SparkSession.builder.appName("Statistics").getOrCreate()

# 读取CSV文件
df = spark.read.csv("/lab4/application_data.csv", header=True, inferSchema=True)

# 注册DataFrame为临时表
df.createOrReplaceTempView("application_data")

# 2.1：统计男性客户的小孩个数类型比例
child_ratios = spark.sql("""
    SELECT CNT_CHILDREN, COUNT(*) / (SELECT COUNT(*) FROM application_data WHERE CODE_GENDER = 'M') AS ratio
    FROM application_data
    WHERE CODE_GENDER = 'M'
    GROUP BY CNT_CHILDREN
    ORDER BY CNT_CHILDREN
""")

# 输出结果
child_ratios.show(truncate=False)
```

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>

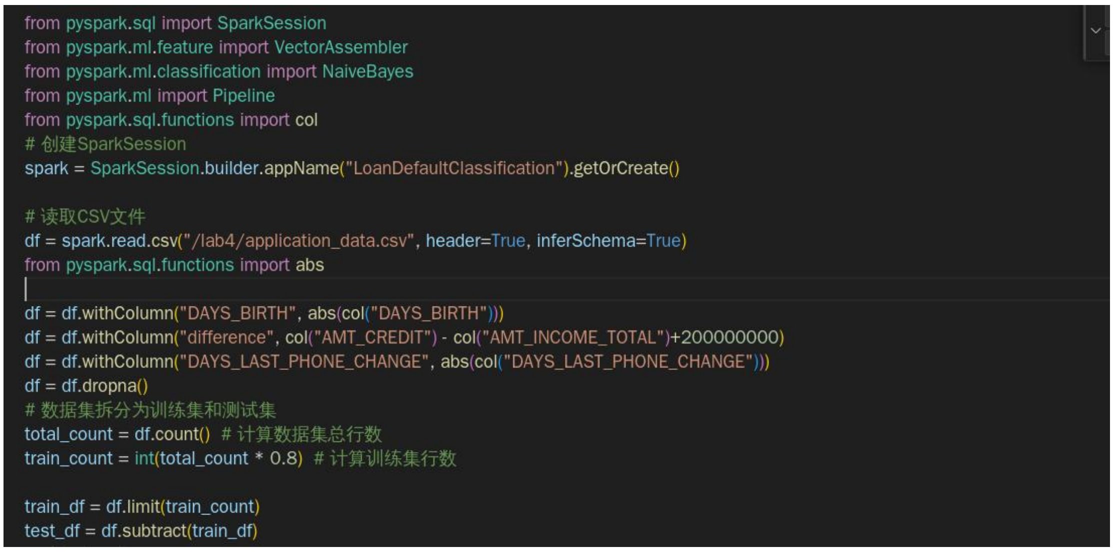
值得注意的是原数据中的（DAYS_BIRTH）均为负数，故在计算时需要加个负号同时还要将结果保存为 csv 文件



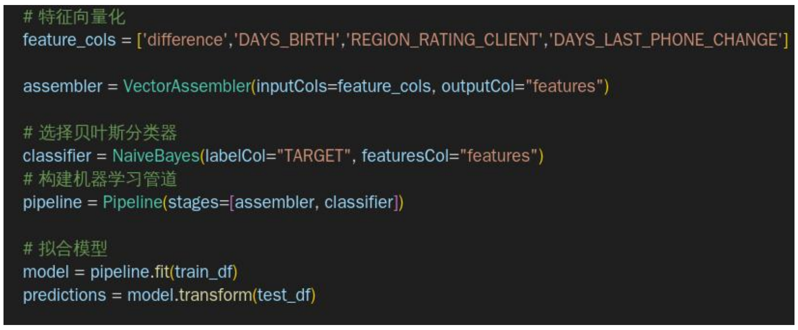
```
# 2.2：统计每个客户出生以来每天的平均收入，保存为CSV文件
avg_income_per_day = spark.sql("""
    SELECT SK_ID_CURR, AMT_INCOME_TOTAL / -DAYS_BIRTH AS avg_income
    FROM application_data
    WHERE AMT_INCOME_TOTAL / -DAYS_BIRTH > 1
    ORDER BY avg_income DESC
""")

avg_income_per_day.show(truncate=False)
# 保存结果
avg_income_per_day.write.csv("/lab4/avg_income_per_day", header=True)

# 停止SparkSession
spark.stop()
```

（4）预测违约
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。

分析：对原数据进行分析后，不难发现原数据是极不平衡的，不违约（TARGET=0）占 92%，而违约（TARGET=1）仅占 8%。
而预测贷款违约最重要的目标在于识别那些将会违约的客户，因为违约的客户将会给银行带来很大损失，而识别出一个不会违约的客户只会带来较少的利息收入。因此，这个问题中最重要的评价指标是召回率 Recall.
综合考虑后选择了贝叶斯分类器。
程序首先完成了读取文件，以及按时间顺序将数据集按 8：2 划分为了训练集和测试集。

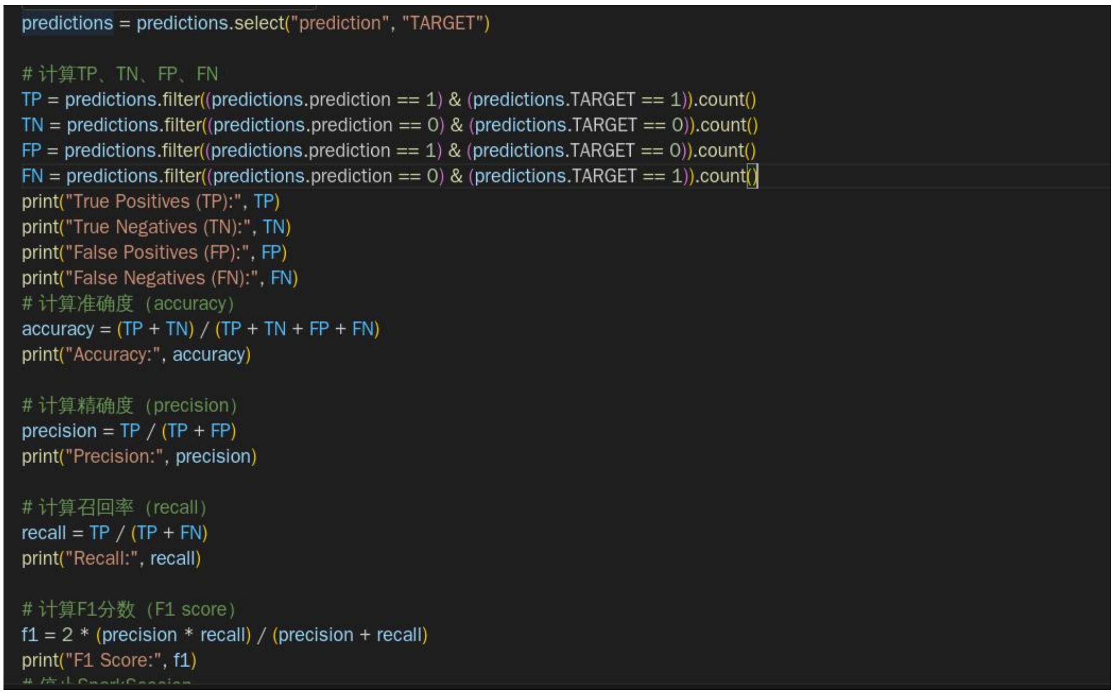

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import NaiveBayes
from pyspark.ml import Pipeline
from pyspark.sql.functions import col
# 创建SparkSession
spark = SparkSession.builder.appName("LoanDefaultClassification").getOrCreate()

# 读取CSV文件
df = spark.read.csv("/lab4/application_data.csv", header=True, inferSchema=True)
from pyspark.sql.functions import abs

df = df.withColumn("DAYS_BIRTH", abs(col("DAYS_BIRTH")))
df = df.withColumn("difference", col("AMT_CREDIT") - col("AMT_INCOME_TOTAL")+200000000)
df = df.withColumn("DAYS_LAST_PHONE_CHANGE", abs(col("DAYS_LAST_PHONE_CHANGE")))
df = df.dropna()
# 数据集拆分为训练集和测试集
total_count = df.count()  # 计算数据集总行数
train_count = int(total_count * 0.8)  # 计算训练集行数

train_df = df.limit(train_count)
test_df = df.subtract(train_df)
```

接着，选择了['difference','DAYS_BIRTH','REGION_RATING_CLIENT','DAYS_LAST_PHONE_CHANGE'] 作为特征向量（是在多次尝试后效果还不错的特征）
之后将特征列放入贝叶斯分类器，并拟合模型



```
# 特征向量化
feature_cols = ['difference','DAYS_BIRTH','REGION_RATING_CLIENT','DAYS_LAST_PHONE_CHANGE']

assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")

# 选择贝叶斯分类器
classifier = NaiveBayes(labelCol="TARGET", featuresCol="features")
# 构建机器学习管道
pipeline = Pipeline(stages=[assembler, classifier])

# 拟合模型
model = pipeline.fit(train_df)
predictions = model.transform(test_df)
```

最后将预测结果与 TARGET 对比得到 TP, TN, FP, FN
并根据 TP, TN, FP, FN 计算准确度，精确度，召回率，F1 分数

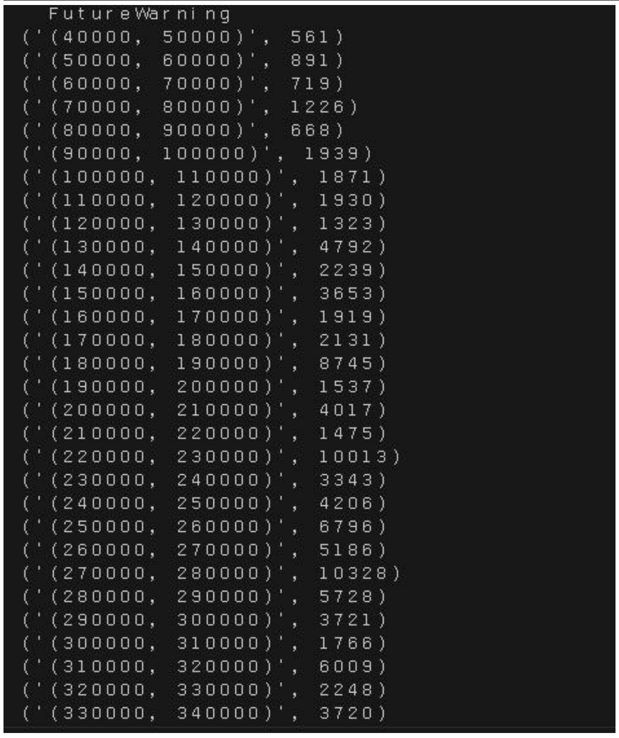


```
predictions = predictions.select("prediction", "TARGET")

# 计算TP、TN、FP、FN
TP = predictions.filter((predictions.prediction == 1) & (predictions.TARGET == 1)).count()
TN = predictions.filter((predictions.prediction == 0) & (predictions.TARGET == 0)).count()
FP = predictions.filter((predictions.prediction == 1) & (predictions.TARGET == 0)).count()
FN = predictions.filter((predictions.prediction == 0) & (predictions.TARGET == 1)).count()
print("True Positives (TP):", TP)
print("True Negatives (TN):", TN)
print("False Positives (FP):", FP)
print("False Negatives (FN):", FN)
# 计算准确度（accuracy）
accuracy = (TP + TN) / (TP + TN + FP + FN)
print("Accuracy:", accuracy)

# 计算精确度（precision）
precision = TP / (TP + FP)
print("Precision:", precision)

# 计算召回率（recall）
recall = TP / (TP + FN)
print("Recall:", recall)

# 计算F1分数（F1 score）
f1 = 2 * (precision * recall) / (precision + recall)
print("F1 Score:", f1)
```

三、实验结果

任务一

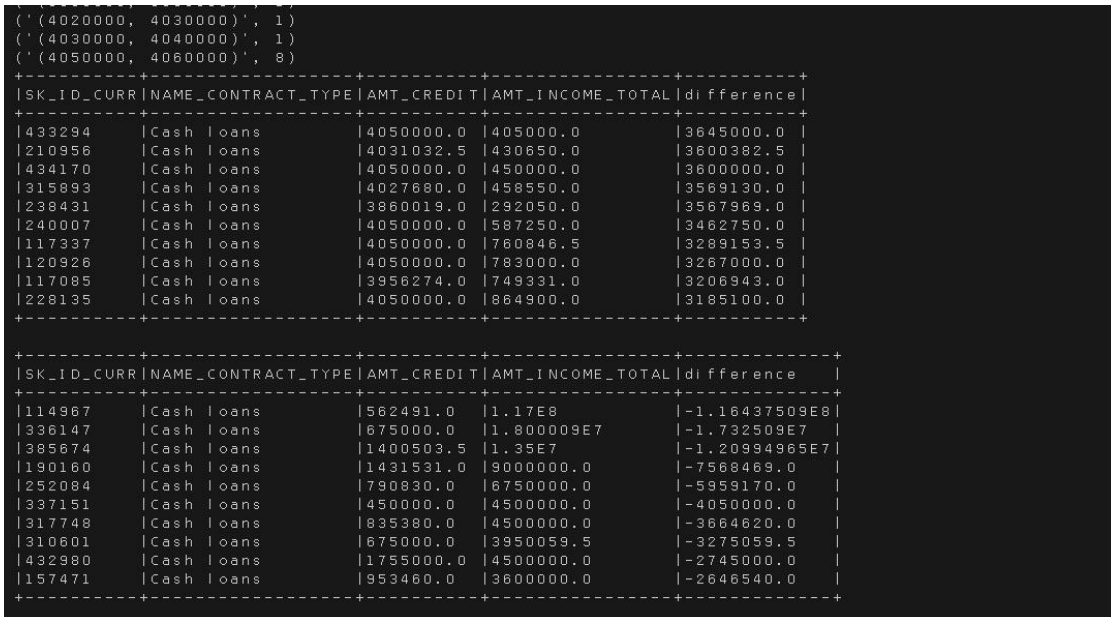
贷款金额分布



```
  FutureWarning
('(40000, 50000)', 561)
('(50000, 60000)', 891)
('(60000, 70000)', 719)
('(70000, 80000)', 1226)
('(80000, 90000)', 668)
('(90000, 100000)', 1939)
('(100000, 110000)', 1871)
('(110000, 120000)', 1930)
('(120000, 130000)', 1323)
('(130000, 140000)', 4792)
('(140000, 150000)', 2239)
('(150000, 160000)', 3653)
('(160000, 170000)', 1919)
('(170000, 180000)', 2131)
('(180000, 190000)', 8745)
('(190000, 200000)', 1537)
('(200000, 210000)', 4017)
('(210000, 220000)', 1475)
('(220000, 230000)', 10013)
('(230000, 240000)', 3343)
('(240000, 250000)', 4206)
('(250000, 260000)', 6796)
('(260000, 270000)', 5186)
('(270000, 280000)', 10328)
('(280000, 290000)', 5728)
('(290000, 300000)', 3721)
('(300000, 310000)', 1766)
('(310000, 320000)', 6009)
('(320000, 330000)', 2248)
('(330000, 340000)', 3720)
```

（部分结果截图）

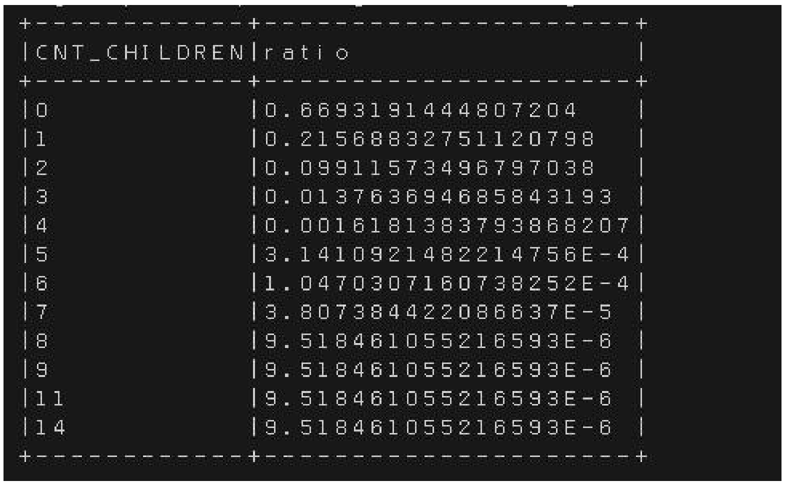
差值最高和最低的各十条记录



```
('(4020000, 4030000)', 1)
('(4030000, 4040000)', 1)
('(4050000, 4060000)', 8)
+----------+------------------+----------+----------------+----------+
|SK_ID_CURR|NAME_CONTRACT_TYPE|AMT_CREDIT|AMT_INCOME_TOTAL|difference|
+----------+------------------+----------+----------------+----------+
|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|difference   |
+----------+------------------+----------+----------------+-------------+
|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   |
+----------+------------------+----------+----------------+-------------+
```

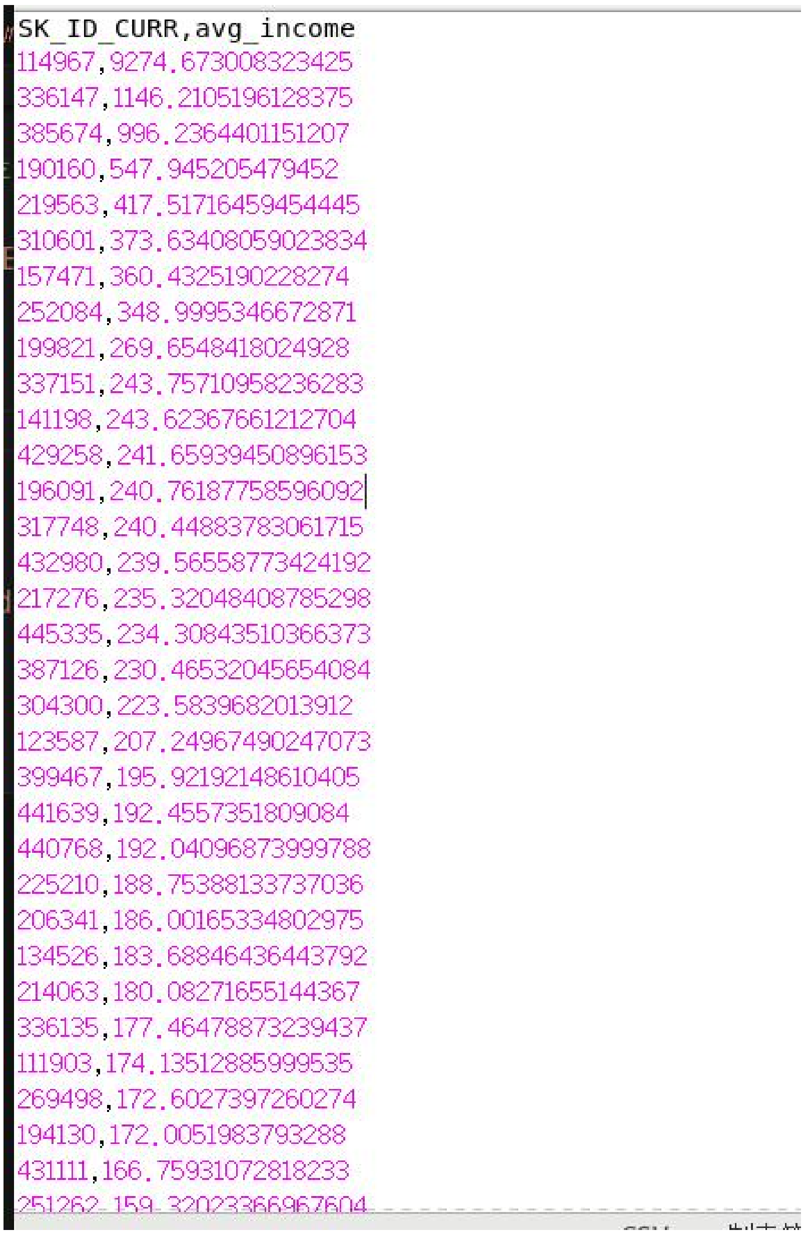
任务二

男性客户小孩类型占比



```
+------------+--------------------+
|CNT_CHILDREN|ratio               |
+------------+--------------------+
|0           |0.669319144480720 4 |
|1           |0.21568832751120798 |
|2           |0.09911573496797038 |
|3           |0.013763694685843193|
|4           |0.0016181383793868207|
|5           |3.1410921482214756E-4|
|6           |1.0470307160738252E-4|
|7           |3.807384422086637E-5|
|8           |9.518461055216593E-6|
|9           |9.518461055216593E-6|
|11          |9.518461055216593E-6|
|14          |9.518461055216593E-6|
+------------+--------------------+
```

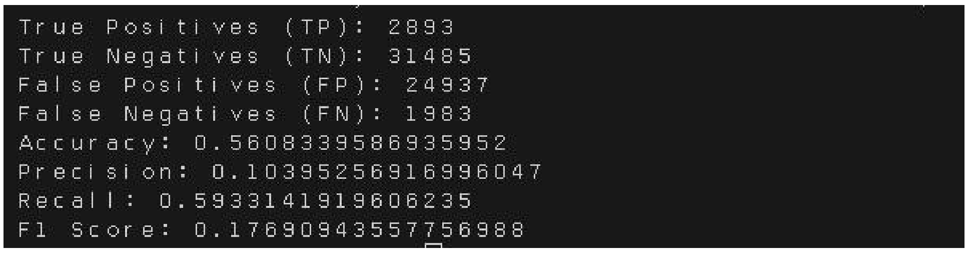
日均收入大于 1 的客户



```
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
399467,195.92192148610405
441639,192.4557351809084
440768,192.04096873999788
225210,188.75388133737036
206341,186.00165334802975
134526,183.68846436443792
214063,180.08271655144367
336135,177.46478873239437
111903,174.13512885999535
269498,172.6027397260274
194130,172.0051983793288
431111,166.75931072818233
251262,159.32023366967604
```

任务三

预测指标



```
True Positives (TP): 2893
True Negatives (TN): 31485
False Positives (FP): 24937
False Negatives (FN): 1983
Accuracy: 0.5608339586935952
Precision: 0.10395256916996047
Recall: 0.5933141919606235
F1 Score: 0.17690943557756988
```

虽然准确度只有 56.1%

但召回率达到了将近 60%还是相对不错的

说明我的贝叶斯分类器可以在确保拥有大概 56%的客户群体的情况下规避 60%的违约损失

四、存在/遇到的问题及改进

1. pyspark 安装后运行问题

在首次安装 pyspark 并重启后发现无法运行

报错为权限不够

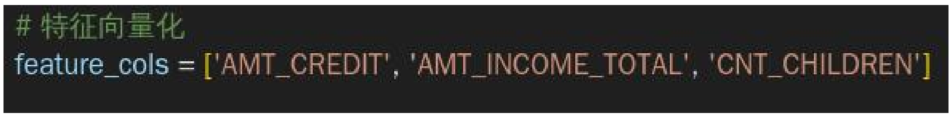
后发现是 pyspark 文件夹变成了管理员才能打开的权限设置

在重新设置权限并重启虚拟机后可以正确打开 pyspark 并在 vscode 中调用相关模块

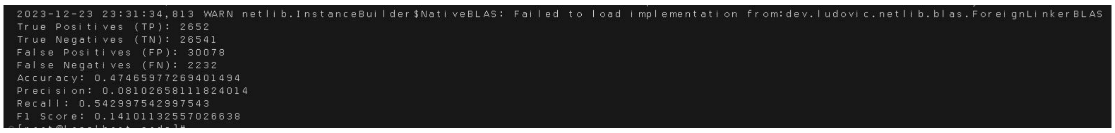
2. 模型预测能力不足

可以看到其实最后得到的预测模型也仅有 60%左右的准确率与召回率

而在之前的尝试中甚至得到的是效果更糟糕的模型，如下面这个贝叶斯分类器的特征选择及对应结果



```
# 特征向量化
feature_cols = ['AMT_CREDIT', 'AMT_INCOME_TOTAL', 'CNT_CHILDREN']
```



```
2023-12-23 23:31:34,813 WARN netlib.InstanceBuilder$NativeBLAS: Failed to load implementation from:dev.ludovic.netlib.blas.ForeignLinkerBLAS
True Positives (TP): 2652
True Negatives (TN): 26541
False Positives (FP): 30078
False Negatives (FN): 2232
Accuracy: 0.47465977269401494
Precision: 0.08102658111824014
Recall: 0.542997542997543
F1 Score: 0.14101132557026638
```

本质上可以通过不断更换、增减特征向量让结果更优，但为避免模型过拟合所以没有在这方面一直做下去

而与此同时，我也选择过其它机器学习分类器，比如逻辑回归、决策树、随机森林等，但效果也不佳，甚至很多模型最终的预测结果为所有测试集都不违约，这样的预测显然没有意义

进一步改进可以考虑在用其它模型的情况下进一步特征优化，也许能得到更优的结果，但同时也需要谨防模型过拟合。