一、环境配置时

（一）Ubuntu虚拟机中

对实验还没有整体把握的时候，一些傻傻的环境配置——

1、启动master节点

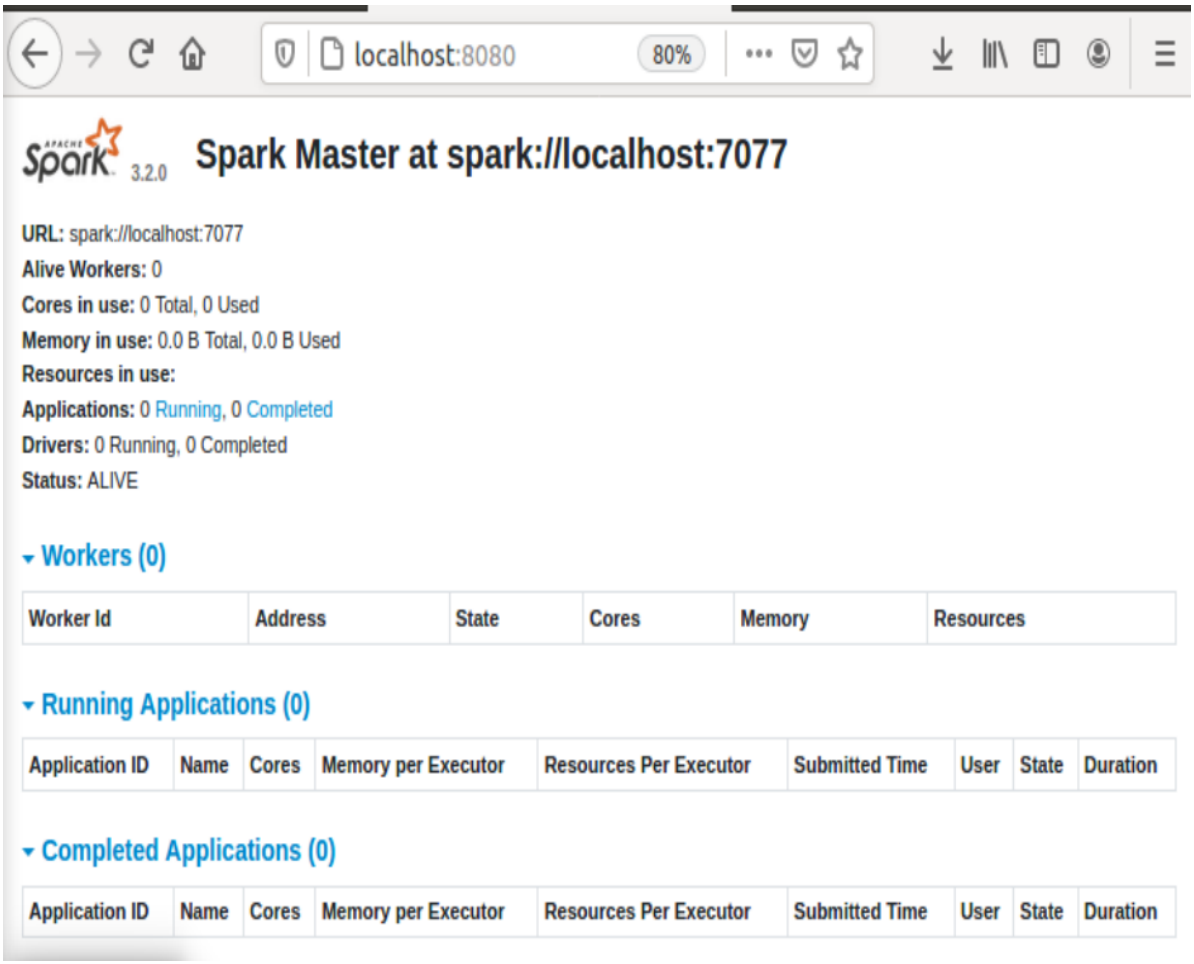
```
zjx@zjx-VirtualBox:~/spark/spark-3.2.0-bin-hadoop3.2/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home/zjx/spark/spark
-3.2.0-bin-hadoop3.2/logs/spark-zjx-org.apache.spark.deploy.master.Master-1-zjx-
VirtualBox.out
zjx@zjx-VirtualBox:~/spark/spark-3.2.0-bin-hadoop3.2/sbin$ jps
6466 SecondaryNameNode
6275 DataNode
6147 NameNode
16134 Jps
16092 Master
```

运行Spark自带的示例，验证Spark是否安装成功：

```
zjx@zjx-VirtualBox:~/spark/spark-3.2.0-bin-hadoop3.2/bin$ ./run-example SparkPi
2>&1 | grep "Pi is"
Pi is roughly 3.1488557442787215
```

2、web IU

spark master：

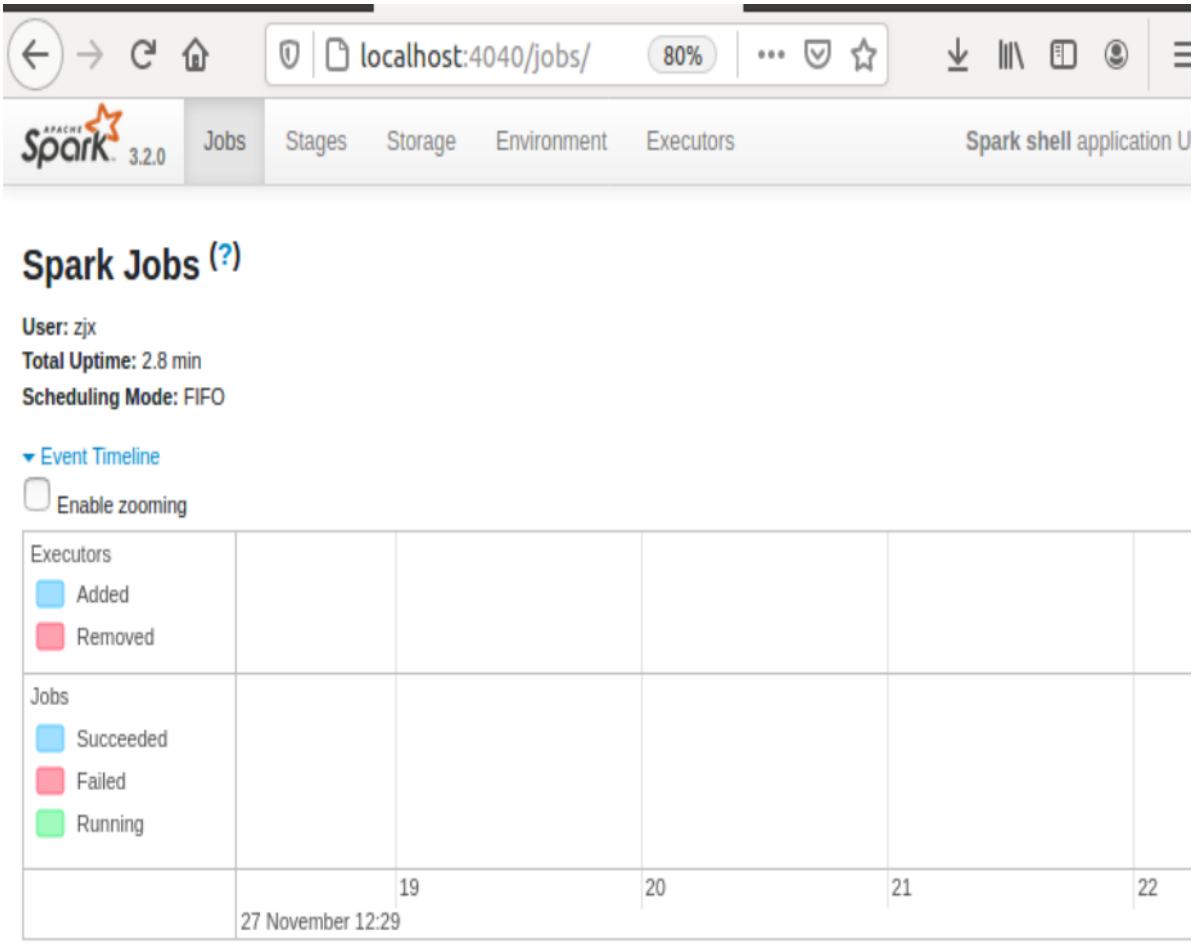


3、运行scala

```
zjx@zjx-VirtualBox:~/spark/spark-3.2.0-bin-hadoop3.2/bin$ ./spark-shell
21/11/26 01:22:01 WARN Utils: Your hostname, zjx-VirtualBox resolves to a loopba
ck address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
21/11/26 01:22:01 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
21/11/26 01:22:12 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1637860933718
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.0
      /_/

Using Scala version 2.12.15 (OpenJDK Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.0
      /_/

Using Scala version 2.12.15 (OpenJDK Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

浏览器中spark状态：



配置完后和虚拟机大眼瞪小眼，不知道怎么进行后续实验，感谢同学们友情提醒，用spark python代码完成此次实验会更舒适。于是开始配置pycharm中的pyspark。

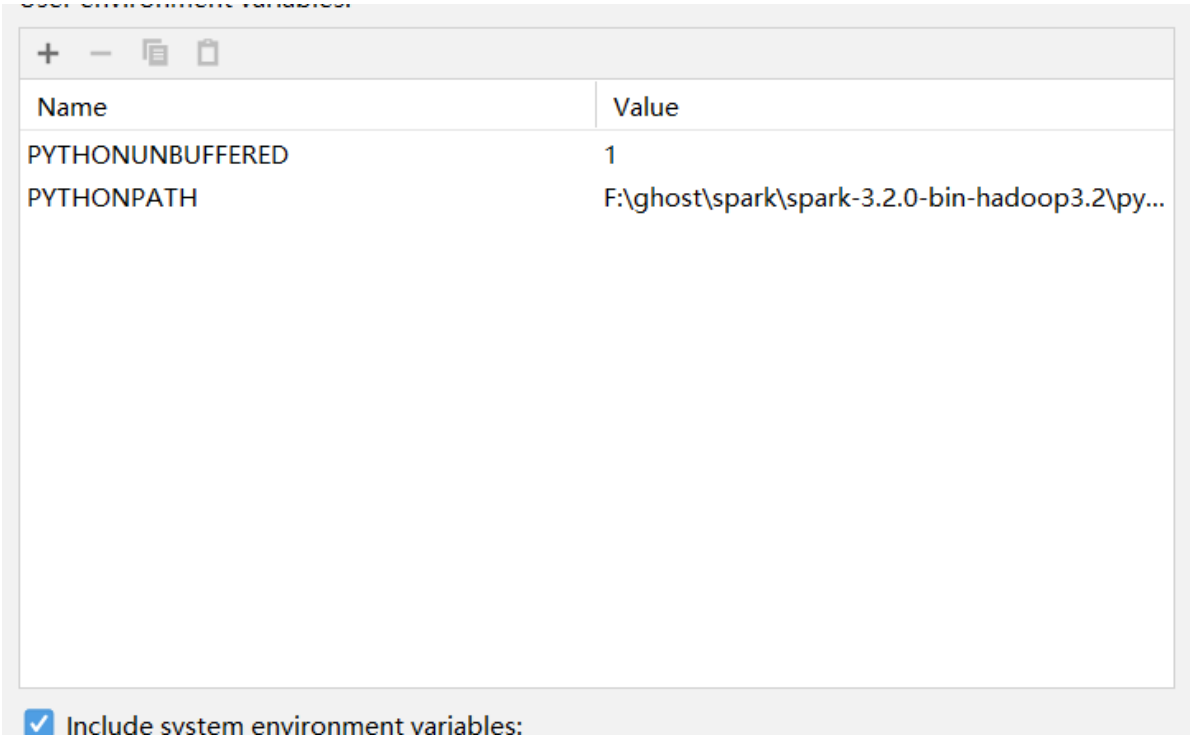
（二）Pycharm中

首先导入pyspark库，以为这样就万事大吉，发现运行代码报错：

```
21/12/16 17:40:01 ERROR Executor: Exception in task 0.0 in stage 0.0 (TID 0)

File "D:\朱家萱\文档\大学\金融大数据\实验4\spark\main.py", line 11, in <module>
  header = lines.first()  # 第一行 print(header)
File "F:\ghost\spark\spark-3.2.0-bin-hadoop3.2\python\pyspark\rdd.py", line 1588, in first
  rs = self.take(1)
File "F:\ghost\spark\spark-3.2.0-bin-hadoop3.2\python\pyspark\rdd.py", line 1568, in take
  res = self.context.runJob(self, takeUpToNumLeft, p)
File "F:\ghost\spark\spark-3.2.0-bin-hadoop3.2\python\pyspark\context.py", line 1227, in runJob
  sock_info = self._jvm.PythonRDD.runJob(self._jsc.sc(), mappedRDD._jrdd, partitions)
File "F:\ghost\spark\spark-3.2.0-bin-hadoop3.2\python\lib\py4j-0.10.9.2-src.zip\py4j\java_gateway.py", line 1309, in __call
File "F:\ghost\spark\spark-3.2.0-bin-hadoop3.2\python\lib\py4j-0.10.9.2-src.zip\py4j\protocol.py", line 326, in get_return_
y4j.protocol.Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.runJob.
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 0.0 failed 1 times, most recent failure:
```

显然是环境配置的问题，但网上的资料很少，一筹莫展，于是采取“大乱炖”法，将可能的解决方法都用一遍。

将编辑器里的环境变量配置如下



并在代码开头指明java路径

```
java8_location = 'F:\ghost\jdk1.8'
os.environ['JAVA_HOME'] = java8_location
```

同时在询问助教并结合网上广罗的方法，更改了原先文件读取方式，强调了master

```
spark = SparkSession.builder \
    .master("local") \
    .appName("xxx") \
    .getOrCreate()
df = spark.read.options(
        header=True,
        inferSchema=True
    ).csv(filename)
```

pycharm的环境设置完成

二、实验中

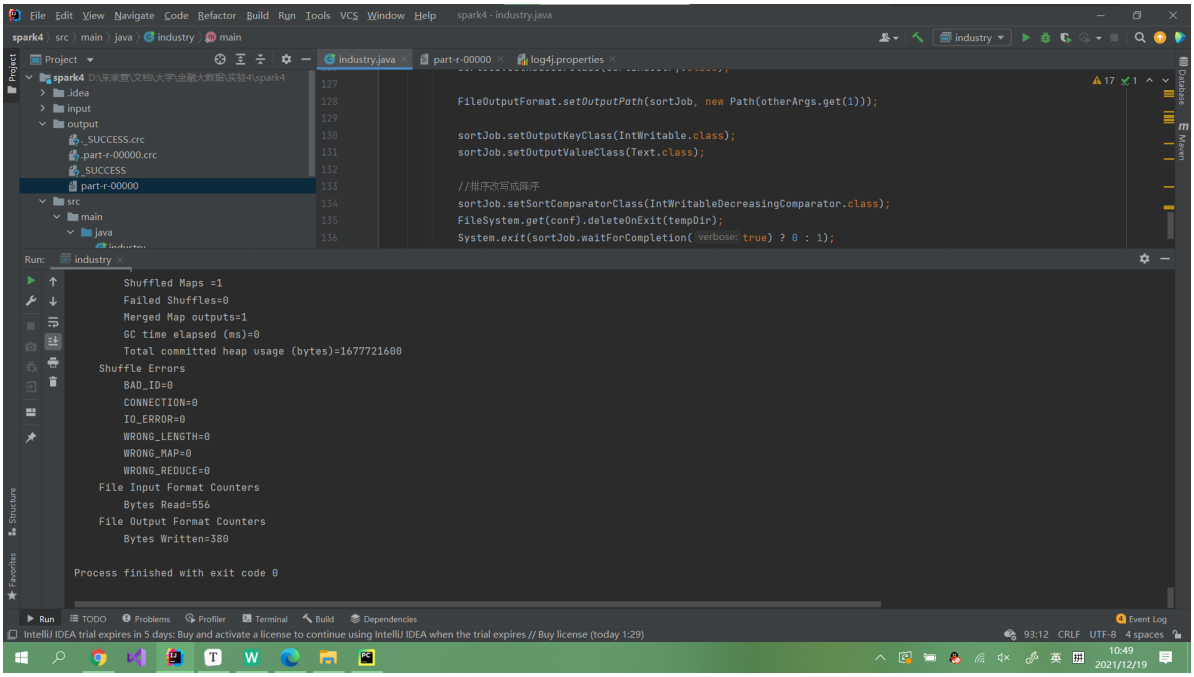
- 任务一延续之前hadoop编程的方法，在intellij idea下用采用java语言完成。
- 任务二三四用pycharm的pyspark库采用python语言完成。

任务一：

> 编写 MapReduce 程序，统计每个工作领域 industry 的网贷记录的数量，并按数量从大到小进行排序。

将训练的csv文件内容读进line，然后处理不需要的表头，industryMapper处理需要count的数据并用逗号分割，输出<industry,1>后，用reducer对次数求和，输出<industry,counts>，再按照industry数量从大到小进行排序。

运行成功



结果写入*task1-output-part-r-00000*中

> 参考资料https://blog.csdn.net/weixin_37275456/article/details/83345858

任务二：

> 编写 Spark 程序，统计网络信用贷产品记录数据中所有用户的贷款金额 total_loan 的分布情况。以 1000 元为区间进行输出

1、pyspark在windows加载数据集 训练模型出现 以下错误

```
java.net.SocketException: Connection reset by peer: socket write error
```

解决：在stackoverflow.com上找到的解决办法
修改spark/python/pyspark目录下worker.py文件
修改 def process():为以下内容，成功解决。

```
def process():
    iterator = deserializer.load_stream(infile)
    serializer.dump_stream(func(split_index, iterator), outfile)
    for obj in iterator:
        pass
```

2、实现方法

读取训练数据集后，先过滤掉header，再利用int类型的除法取整：

```
(total_loan÷1000)×1000
```

将total_loan归纳在它所属的区间，用**map**将函数应用于RDD中的每个元素， 将返回值构成新的RDD。

将output按第一个数值升序**排序**：

```
output.sort(key=lambda x:x[0])
```

3、输出

将output中的值按要求输出：

```
((0,1000),2)
((1000,2000),4043)
((2000,3000),6341)
((3000,4000),9317)
((4000,5000),10071)
((5000,6000),16514)
((6000,7000),15961)
((7000,8000),12789)
((8000,9000),16384)
((9000,10000),10458)
((10000,11000),27170)
((11000,12000),7472)
((12000,13000),20513)
((13000,14000),5928)
((14000,15000),8888)
((15000,16000),18612)
((16000,17000),11277)
((17000,18000),4388)
((18000,19000),9342)
((19000,20000),4077)
((20000,21000),17612)
((21000,22000),5507)
((22000,23000),3544)
((23000,24000),2308)
((24000,25000),8660)
```

```
((25000,26000),8813)
((26000,27000),1604)
((27000,28000),1645)
((28000,29000),5203)
((29000,30000),1144)
((30000,31000),6864)
((31000,32000),752)
((32000,33000),1887)
((33000,34000),865)
((34000,35000),587)
((35000,36000),11427)
((36000,37000),364)
((37000,38000),59)
((38000,39000),85)
((39000,40000),30)
((40000,41000),1493)
```

结果存储在文件夹*output2*中

任务三：

一、统计所有用户所在公司类型 employer_type 的数量分布占比情况。

实现方法：

将employer_type提取出来，统计数量后，存储在type_counts中，再返回出RDD中所有元素，将数量求和，相除后得到数量分布占比情况。用**pandas库**写入*3_1.csv*中，结果保存在*output3_1*文件夹中。

```
type_counts = types.map(lambda x:(x,1)).reduceByKey(lambda a,b:a+b)
```

二、统计每个用户最终须缴纳的利息金额

实现方法：

选择出user_id、year_of_loan、monthly_payment、total_loan

```
lines=spark_df.select("user_id","year_of_loan","monthly_payment", "total_loan")
```

按公式计算最终须缴纳的利息金额，并保留4位小数

```
total_money = lines.rdd.map(
        lambda x: (x.user_id, round((float(x.year_of_loan) *
float(x.monthly_payment) * 12 -float(x.total_loan)),4)))
```

写入*3_2.csv*中，结果保存在*output3_2*文件夹中。

三、统计工作年限 work_year 超过 5 年的用户的房贷情况 censor_status 的数量分布占比情况

实现方法：

首先挑选出user_id、censor_status、work_year三列，为防止影响后续工作，先用

```
lines=lines.fillna("0 year")
```

将空白处填补。

work_year有四种类型：10+ years、n years(n<10,n∈Z*)、n(n≤10,n∈Z*)、 < 1 year，需要特别处理的是10+ years、n years和 < 1 year。抓住<和1之间有空格的特征，我们用map函数将lines中的work_year一列按空格分割并提取倒数第二位的string。

处理后的work_year有两种类型：10+和n(n<10,n∈Z*)。符合条件的是10+和大于5的n，即筛选字符串长度大于1的和长度等于1但转换为int类型后值大于5的work_year，具体实现代码为：

```
res = lines.rdd.map(
        lambda x: (x.user_id,x.censor_status,x.work_year.split(" ")[-2])).
        filter(lambda y: len(y[2])>1 or (len(y[2])==1 and int(y[2])>5))
```

写入*3_3.csv*，结果输出在*output3_3*文件夹中。

任务四：

由于先有折磨的机器学习导论的熏陶，对模型的训练和分类器都有了一定的知识积累，写起来就没有那么难以下手。对我来说主要难点是在数据处理上，需要剔除无关的列、处理缺失值、处理数据类型。

剔除无关的列：

```
spark_df = spark_df.drop('loan_id').drop('user_id')
```

处理缺失值：

```
spark_df = spark_df.na.fill("0")
```

处理数据类型：

```
indexer=StringIndexer(inputCol=item,outputCol="%sIndex"%item)
spark_df=spark_df.withColumn(item,spark_df[item].cast(typ.DoubleType()))
```

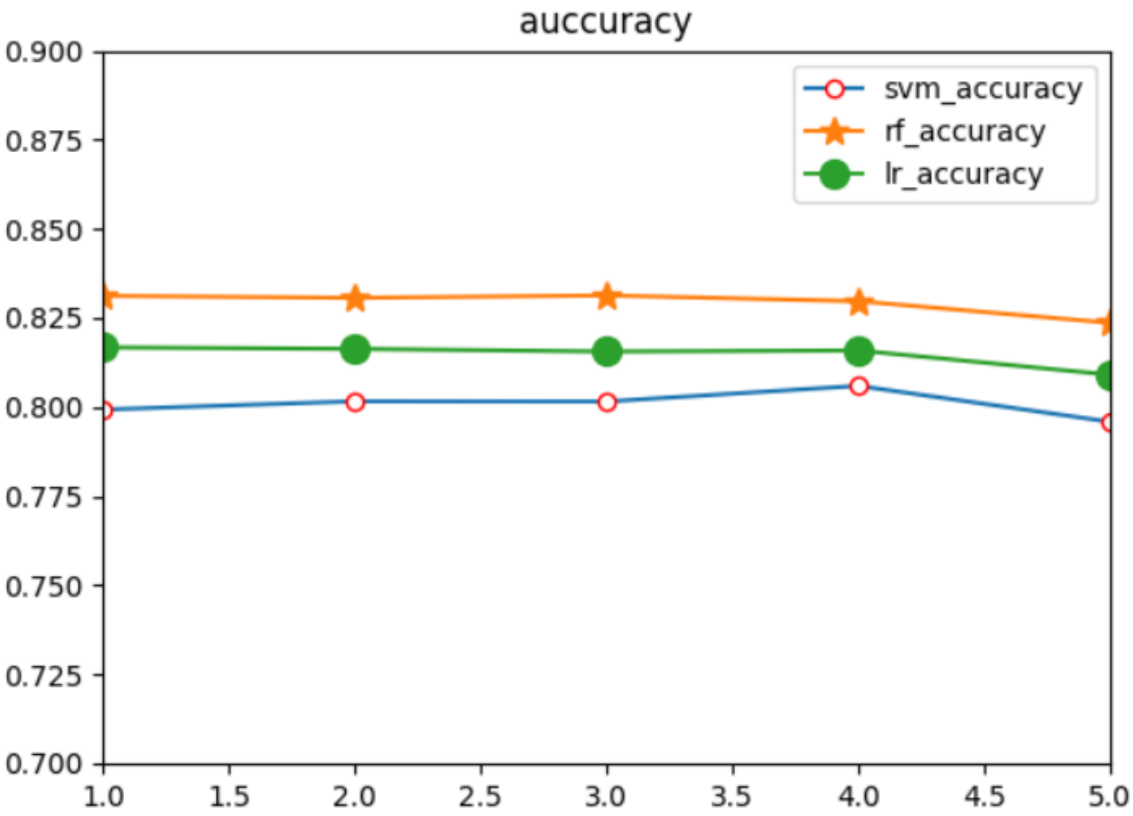
结果：

按9：1到5：5拆分数据集，得到的结果列示在下表中

训练集：测试集	SVM支持向量机	随机森林	逻辑回归
9：1	79.59%	82.26%	80.91%
8：2	80.60 %	83.41 %	81.59 %
7：3	80.16%	83.34%	81.56%
6：4	80.16%	83.46%	81.63%
5：5	79.73%	83.36%	81.68%

部分输出展示：

```
When the ratio is 6.0 to 4.0
Accuracy of SVM on the test set is :  80.16 %
Accuracy of RandomForest on the test set is :  83.46 %
Accuracy of LogisticRegression on the test set is :  81.63 %
When the ratio is 5.0 to 5.0
Accuracy of SVM on the test set is :  79.93 %
Accuracy of RandomForest on the test set is :  83.46 %
Accuracy of LogisticRegression on the test set is :  81.68 %
```



纵向对比，发现不同拆分比对各分类器的准确率几乎没有影响，横向对比，均是随机森林的精度较高，逻辑回归次之，SVM最差，但差距并不大。可能是数据集分布较均匀，不同的训测比不会对结果有影响。