Lab03 - PySpark

Speaker: Wei-Lun Tseng ECE, NYCU eric840610.ee02@g2.nctu.edu.tw

Class Schedule

Total: 5 points

- Lab-1: PySpark Warmup
- Lab-2: PySpark Exercise

- In this course the main way we will be working with Python and Spark is through the DataFrame Syntax.
- If you've worked with pandas in Python,
 R, SQL or even Excel, a DataFrame will feel very familiar!

- Spark DataFrames hold data in a column and row format.
- Each column represents some feature or variable.
- Each row represents an individual data point.

- Spark began with something known as the "RDD" syntax which was a little ugly and tricky to learn.
- Now Spark 2.0 and higher has shifted towards a DataFrame syntax which is much cleaner and easier to work with!

- Spark DataFrames are able to input and output data from a wide variety of sources.
- We can then use these DataFrames to apply various transformations on the data.

- At the end of the transformation calls, we can either show or collect the results to display or for some final processing.
- In this section we'll cover all the main features of working with DataFrames that you need to know.

 Once we have a solid understanding of Spark DataFrames, we can move on to utilizing the DataFrame MLlib API for Machine Learning. google colab





▶ 影片 🚨 圖片 🖺 書籍 🗷 購物

: 更多

丁具

約有 11,100,000 項結果 (搜尋時間: 0.30 秒)

https://colab.research.google.com > ... ▼

Colab - Google

Colaboratory (簡稱為「Colab」) 可讓你在瀏覽器上撰寫及執行Python,且具備下列優點:. 不必 進行任何設定: 免費使用GPU: 輕鬆共用. 無論你是學生、數據資料學家 ...

https://research.google.com > colaboratory ▼ 翻譯這個網頁

Welcome To Colaboratory - Google Research

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code ...

Mounting Google Drive locally · Google Colab FAQ · Google Colab · Colab Widgets

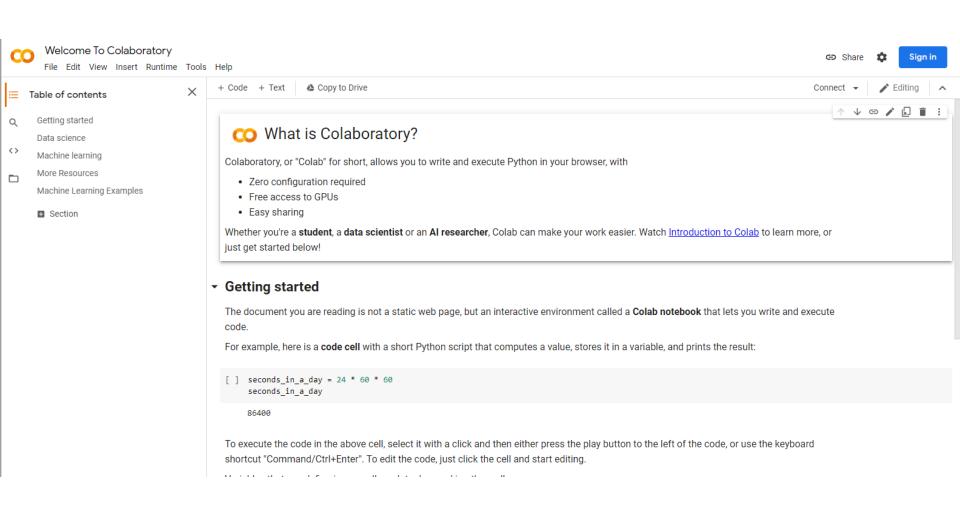
https://medium.com > 透過-google-colaboratory-學習... ▼

透過Google Colaboratory 學習使用Python 做機器學習等科學 ...

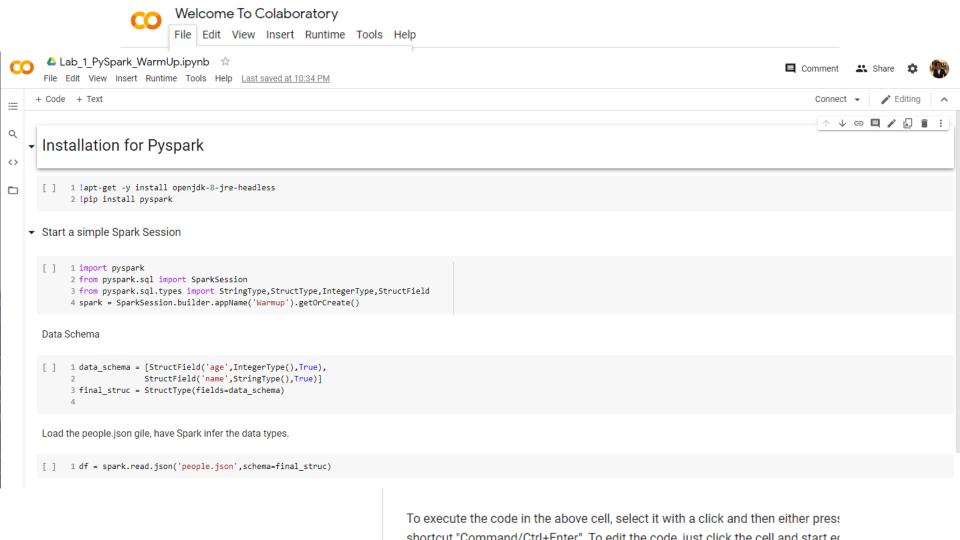
Google Colaboratory (以下簡稱Google Colab)是一個基於Jupyter Notebook 的免費服務(須註 冊一個Google 帳號、其餘部份至少撰文的此刻仍是免費),所以 ...

https://www.bnext.com.tw > article > recommand-to-pr... •

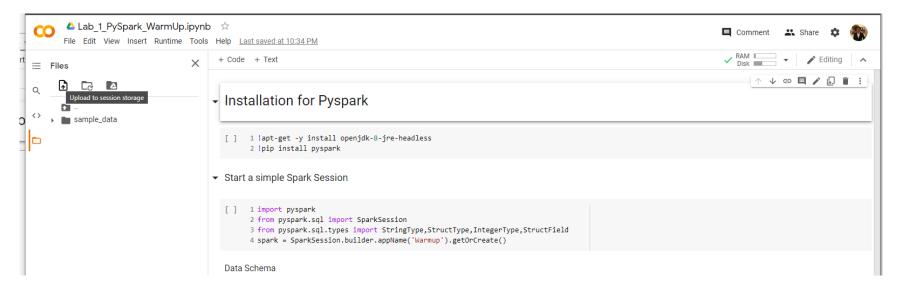




Login your account...







Installation for Pyspark

1 !apt-get -y install openjdk-8-jre-headless

```
2 !pip install pyspark
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
 libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
 fonts-way-microhei fonts-way-zenhei fonts-indic
The following NEW packages will be installed:
 openidk-8-ire-headless
0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded.
Need to get 28.2 MB of archives.
After this operation, 104 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openjdk-8-jre-headless amd64 8u292-b10-0ubuntu1~18.04 [28.2 MB]
Fetched 28.2 MB in 2s (15.9 MB/s)
Selecting previously unselected package openjdk-8-jre-headless:amd64.
(Reading database ... 160772 files and directories currently installed.)
Preparing to unpack .../openjdk-8-jre-headless 8u292-b10-0ubuntu1~18.04 amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
Setting up openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
Collecting pyspark
     212.4MB 63kB/s
Collecting py4j==0.10.9
                                       204kB 19.1MB/s
Building wheels for collected packages: pyspark
 Building wheel for pyspark (setup.py) ... done
 Stored in directory: /root/.cache/pip/wheels/40/1b/20/30f43be2627857ab80062bef1527c0128f7b4070b6b2d02139
Successfully built pyspark
Installing collected packages: py4j, pyspark
```

Successfully installed py4j-0.10.9 pyspark-3.1.2

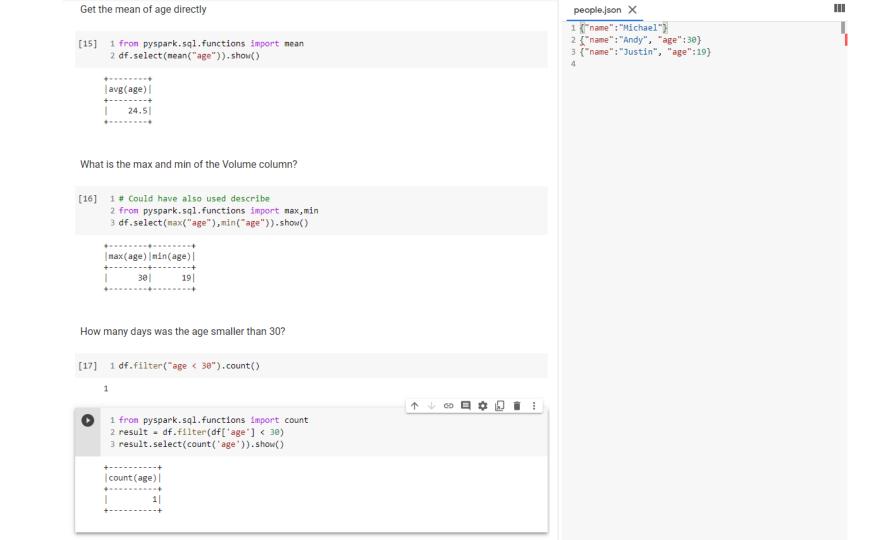
Check-Point-1: Successfully installed Pyspark 0.5 point

```
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
                     <del>using /usr/lib/jvm/java 8 openjdk</del>amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
 Downloading https://files.pythonhosted.org/packages/19/db/e18cfd78e408de957821ec5ca56de1250645b05f8523d169803d8df35a64/pyspark-3.1.2.tar.gz (212.4MB)
 Downloading https://files.pythonhosted.org/packages/ge/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-any.w
 Created wheel for pyspark: filename=pyspark-3.1.2-py1.py3-none-any.whl size=212880768 sha256=b63ae7fb090bc38cdd82055abe3e77e4b9b8aee1d84c4cb3a9b36193
```

```
[2] 1 import pyspark
       2 from pyspark.sql import SparkSession
       3 from pyspark.sql.types import StringType,StructType,IntegerType,StructField
       4 spark = SparkSession.builder.appName('Warmup').getOrCreate()
  Data Schema
  [3] 1 data_schema = [StructField('age',IntegerType(),True),
   2 StructField('name',StringType(),True)]
      3 final_struc = StructType(fields=data_schema)
  Load the people.json gile, have Spark infer the data types.
  [5] 1 df = spark.read.json('people.json',schema=final_struc)
▼ What are the column names?
  [6] 1 df.columns
     ['age', 'name']
▼ What is the schema?
  [7] 1 df.printSchema()
       |-- age: integer (nullable = true)
       |-- name: string (nullable = true)
  Show whole DataFrame
                                                                                                                    ↑ ↓ © 目 ‡ ॄ . i :
  1 df.show()
      age name
      +----+
      |null|Michael|
      | 30| Andy|
      | 19| Justin|
      +----+
```

▼ Start a simple Spark Session

```
Print out the first 2 rows.
[9] 1 # Didn't strictly need a for loop, could have just then head()
     2 for row in df.head(2):
     3 print(row)
     4 print('\n')
    Row(age=None, name='Michael')
    Row(age=30, name='Andy')
Use describe() to learn about the DataFrame
[10] 1 df.describe()
    DataFrame[summary: string, age: string, name: string]
Use another data frame to learn about the statistical report
[11] 1 temp = df.describe()
     2 temp.show()
    +----+
    +----+
     count 2 3 mean 24.5 null
     stddev|7.7781745930520225| null|
       min 19 Andy
              30|Michael|
       max
    +----+
There are too many decimal places for mean and stddev in the describe() dataframe.
How to deal with it?
[13] 1 from pyspark.sql.functions import format_number
                                                                                                           ↑ ↓ ⊖ 🛢 🛊 🖟 👔 🔋
    1 result = df.describe()
     2 result.select(result['summary'],
                  format_number(result['age'].cast('float'),2).alias('age')
                 ).show()
    +----+
    |summary| age|
    +----+
      count| 2.00|
       mean 24.50
     stddev| 7.78|
        min|19.00|
```



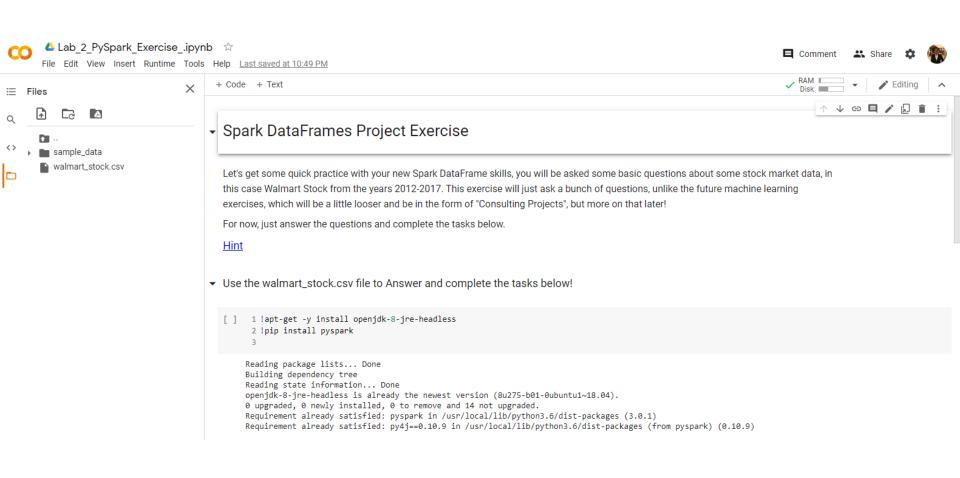
Check-Point-2:

1 point



Lab-2

PySpark - Exercise



[1] 1 !apt-get -y install openjdk-8-jre-headless 2 !pip install pyspark Reading package lists... Done Building dependency tree Reading state information... Done Suggested packages: libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic The following NEW packages will be installed: openjdk-8-jre-headless 0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded. Need to get 28.2 MB of archives. After this operation, 104 MB of additional disk space will be used. Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 openidk-8-ire-headless amd64 8u292-b10-0ubuntu1~18.04 [28.2 MB] Fetched 28.2 MB in 1s (50.6 MB/s) Selecting previously unselected package openjdk-8-jre-headless:amd64. (Reading database ... 160772 files and directories currently installed.) Preparing to unpack .../openjdk-8-jre-headless 8u292-b10-0ubuntu1~18.04 amd64.deb ... Unpacking openjdk-8-jre-headless:amd64 (8u292-b10-0ubuntu1~18.04) ... Setting up openidk-8-ire-headless:amd64 (8u292-b10-0ubuntu1~18.04) ... update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode Collecting pyspark Downloading https://files.pvthonhosted.org/packages/89/db/e18cfd78e408de957821ec5ca56de1250645b05f8523d169803d8df35a64/pvspark-3.1.2.tar.gz (212.4MB) 212.4MB 71kB/s Collecting py4j==0.10.9 Downloading https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-0.10.9-py2.py3-none-anv.w 204kB 21.1MB/s Building wheels for collected packages: pyspark Building wheel for pyspark (setup.py) ... done Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=9dbb7b7ed3d5afe50b472af777003fe07c00f5516b6d6bedadc985bc Stored in directory: /root/.cache/pip/wheels/40/1b/2c/30f43be2627857ab80062bef1527c0128f7b4070b6b2d02139 Successfully built pyspark Installing collected packages: py4j, pyspark Successfully installed py4j-0.10.9 pyspark-3.1.2

▼ Start a simple Spark Session

[2] 1 from pyspark.sql import SparkSession

▼ Load the Walmart Stock CSV File, have Spark infer the data types.

2 spark = SparkSession.builder.appName("walmart").getOrCreate()

[3] 1 df = spark.read.csv('walmart stock.csv',header=True,inferSchema=True)

▼ Use the walmart_stock.csv file to Answer and complete the tasks below!

```
▼ What are the column names?
  [4] 1 df.columns
       ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
▼ What does the Schema look like?
  [6] 1 df.printSchema()
        |-- Date: string (nullable = true)
        |-- Open: double (nullable = true)
        |-- High: double (nullable = true)
        |-- Low: double (nullable = true)
        |-- Close: double (nullable = true)
        |-- Volume: integer (nullable = true)
        |-- Adj Close: double (nullable = true)

    Print out the first 5 columns.

  [7] 1 # Didn't strictly need a for loop, could have just then head()
        2 for row in df.head(5):
              print(row)
              print('\n')
       Row(Date='2012-01-03', Open=59.970001, High=61.060001, Low=59.869999, Close=60.330002, Volume=12668800, Adj Close=52.619234999999996)
       Row(Date='2012-01-04', Open=60.20999899999996, High=60.349998, Low=59.470001, Close=59.7099989999996, Volume=9593300, Adj Close=52.078475)
       Row(Date='2012-01-05', Open=59.349998, High=59.619999, Low=58.369999, Close=59.419998, Volume=12768200, Adj Close=51.825539)
       Row(Date='2012-01-06', Open=59.419998, High=59.450001, Low=58.869999, Close=59.0, Volume=8069400, Adj Close=51.45922)
       Row(Date='2012-01-09', Open=59.029999, High=59.549999, Low=58.919998, Close=59.18, Volume=6679300, Adj Close=51.6162150000000004)
```

▼ Use describe() to learn about the DataFrame.

[8] 1 df.describe().show()

summary	Date	Open	High	Low	Close	Volume	Adj Close
count	1258	1258	1258	1258	1258	1258	1258
mean	null	72.35785375357709	72.83938807631165	71.9186009594594	72.38844998012726	8222093.481717011	67.23883848728146
stddev	null	6.76809024470826	6.768186808159218	6.744075756255496	6.756859163732991	4519780.8431556	6.722609449996857
min	2012-01-03	56.389998999999996	57.060001	56.299999	56.419998	2094900	50.363689
max	2016-12-30	90.800003	90.970001	89.25	90.470001	80898100	84.914216000000001

```
[9] 1 # Uh oh Strings!
      2 df.describe().printSchema()
     root
      |-- summary: string (nullable = true)
       -- Date: string (nullable = true)
      -- Open: string (nullable = true)
      -- High: string (nullable = true)
      -- Low: string (nullable = true)
      |-- Close: string (nullable = true)
      |-- Volume: string (nullable = true)
      |-- Adj Close: string (nullable = true)
[12] 1 # hint
      2 from pyspark.sql.functions import format number
                                                                                                                            ↑ ↓ ⊖ 目 ‡ ♬ 👔
      1 result = df.describe()
      2 result.select(result['summary'],
                     format number(result['Open'].cast('float'),2).alias('Open'),
                     result['Volume'].cast('int').alias('Volume')
                    ).show()
                 Open | Volume |
       count[1,258.00] 1258]
        mean | 72.36 | 8222093 |
       stddevl
              6.77 | 4519780 |
         min| 56.39| 2094900|
                90.80 | 80898100 |
       -----+
```

Check-Point-3:

Check Point 3

format Open, High, Low, Close, Volume, Adj Close

1 point

```
[14] 1
```

```
High
summary
           Open|
                             Low
                                    Close
 count | 1,258.00 | 1,258.00 | 1,258.00 | 1,258.00 |
                                             1258
          72.36
                 72.84
                           71.92 | 72.39 | 8222093 |
  mean
                  6.77 | 6.74 | 6.76
stddevl
           6.77
                                          4519780
                 57.06 | 56.30 | 56.42 | 2094900
   min
          56.39
          90.80
                   90.97
                           89.25 90.47 80898100
   max
```

Create a new dataframe with a column called HV Ratio that is the ratio of the High Price versus volume of stock traded for a day.

```
1 df2 = df.withColumn("HV Ratio",df["High"]/df["Volume"])#.show()
[15]
      2 # df2.show()
      3 df2.select('HV Ratio').show()
                 HV Ratio
     4.819714653321546E-6
      6.290848613094555E-6
      4.669412994783916E-6
      7.367338463826307E-6
      8.915604778943901E-6
      8.644477436914568E-6
      9.351828421515645E-6
       8.29141562102703E-6
     7.712212102001476E-6
     7.071764823529412E-6
     1.015495466386981E-5
      6.576354146362592...
       5.90145296180676E-6
      8.547679455011844E-6
      8.420709512685392E-6
     1.041448341728929...
     8.316075414862431E-6
     9.721183814992126E-6
      8.029436027707578E-6
      6.307432259386365E-6
     only showing top 20 rows
```

▼ What day had the Peak High in Price?

```
[16] 1 # Didn't need to really do this much indexing
2 # Could have just shown the entire row
3 df.orderBy(df["High"].desc()).head(1)[0][0]
'2015-01-13'
```

▼ What is the mean of the Close column?



Check-Point-4:

Check Point 4

[20] 1

What is the max and min of the Volume column?

1 point

```
[19] 1 # Could have also used describe
2 from pyspark.sql.functions import max,min
```

```
+-----+
|max(Volume)|min(Volume)|
+-----+
| 80898100| 2094900|
```

```
[24] 1 df.filter("Close < 60").count()</pre>
      81
 [23] 1 df.filter(df['Close'] < 60).count()</pre>
      81
  [26] 1 from pyspark.sql.functions import count
        2 result = df.filter(df['Close'] < 60)</pre>
        3 result.select(count('Close')).show()
       +----+
       |count(Close)|
       +----+
                 81
       +----+
▼ What percentage of the time was the High greater than 80 dollars?
  In other words, (Number of Days High>80)/(Total Days in the dataset)
 [27] 1 # Many ways to do this
        2 (df.filter(df["High"]>80).count()*1.0/df.count())*100
       9.141494435612083
▼ What is the Pearson correlation between High and Volume?
  [28] 1 from pyspark.sql.functions import corr
        2 df.select(corr("High","Volume")).show()
       +----+
```

▼ How many days was the Close lower than 60 dollars?

What is the max High per year?

2 # 2015

|Year|max(High)|

|2015|90.970001| |2013|81.370003| |2014|88.089996| |2012|77.599998| |2016|75.190002|

```
[29] 1 from pyspark.sql.functions import year
2 yeardf = df.withColumn("Year",year(df["Date"]))
```

[30] 1 max_df = yeardf.groupBy('Year').max()

3 max_df.select('Year', 'max(High)').show()

▼ What is the max Close for each Calendar Month?

In other words, across all the years, what is the max Close price for Jan, Feb, Mar, etc... Your result will have a value for each of these months.



Check-Point-5:

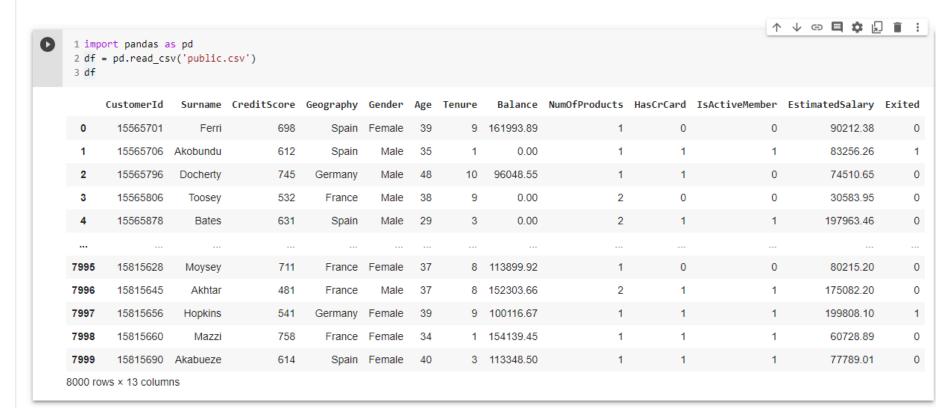
1 point



For homework 4:

PySpark ML lib warmup

Dataset Overview



Great ~! Check Point 6

Download your jupyter notebook and upload to new E3 (Lab.3 PySpark). (0.5 points)