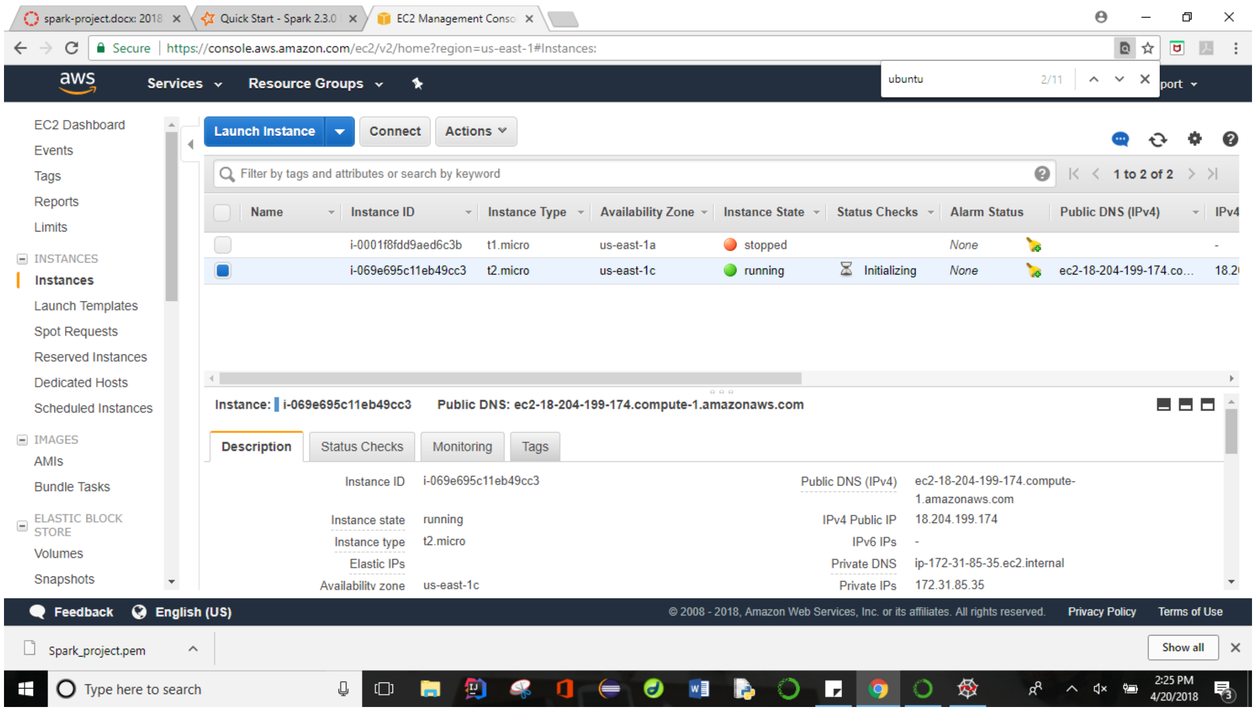
Cloud Data Storage

Project – Spark

Ishita Shah – 801034015

I have created an instance on AWS , My Instance - ec2-18-204-199-174.compute-1.amazonaws.com



https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4
☐		i-0001f8fdd9aed6c3b	t1.micro	us-east-1a	● stopped		None		-
☑		i-069e695c11eb49cc3	t2.micro	us-east-1c	● running	Initializing	None	ec2-18-204-199-174.co...	18.2

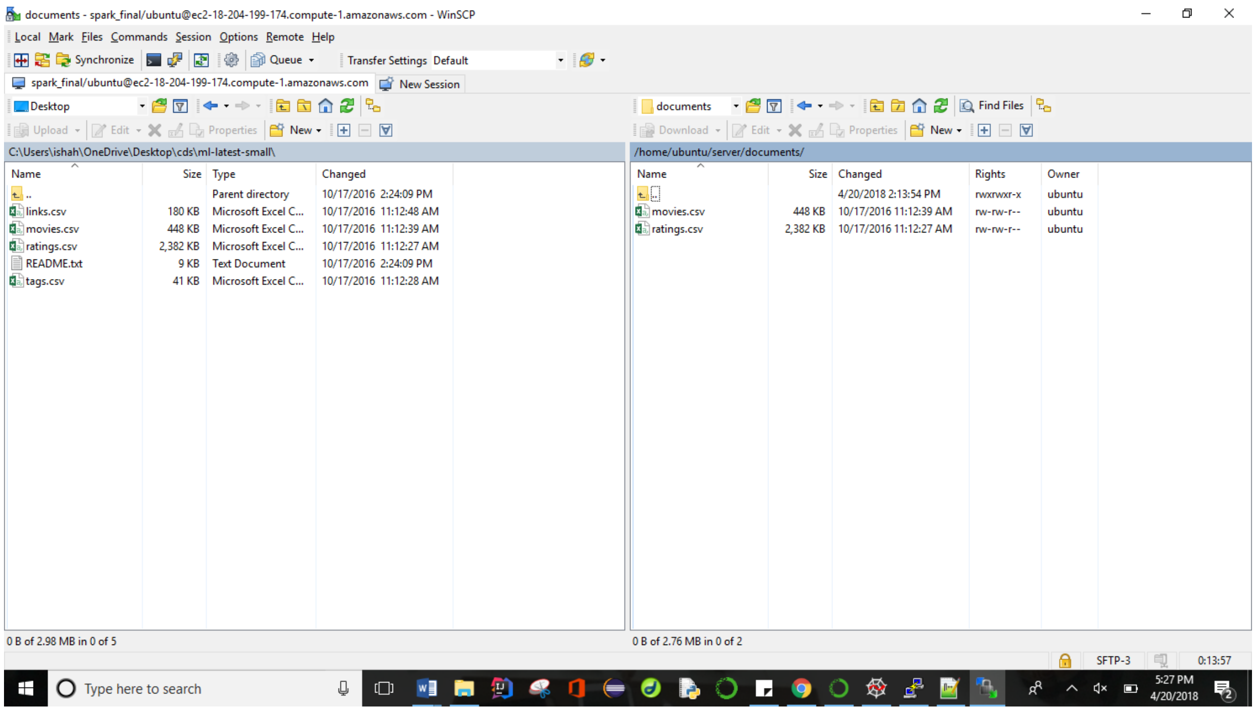
Instance: i-069e695c11eb49cc3   Public DNS: ec2-18-204-199-174.compute-1.amazonaws.com

Instance ID	i-069e695c11eb49cc3	Public DNS (IPv4)	ec2-18-204-199-174.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	18.204.199.174
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-85-35.ec2.internal
Availability zone	us-east-1c	Private IPs	172.31.85.35

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

mkdir server
ls // gave the list in server directory
cd server
wget http://apache.claz.org/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz // download spark
tar xvzf spark-2.3.0-bin-hadoop2.7.tgz

Converted the file on WinSCP.Choose small Data set, used movies.csv, ratings.csv



documents - spark_final/ubuntu@ec2-18-204-199-174.compute-1.amazonaws.com - WinSCP

Transfer Settings Default

spark_final/ubuntu@ec2-18-204-199-174.compute-1.amazonaws.com

Desktop

C:\Users\ishah\OneDrive\Desktop\cds\ml-latest-small\

Name	Size	Type	Changed
..		Parent directory	10/17/2016 2:24:09 PM
links.csv	180 KB	Microsoft Excel C...	10/17/2016 11:12:48 AM
movies.csv	448 KB	Microsoft Excel C...	10/17/2016 11:12:39 AM
ratings.csv	2,382 KB	Microsoft Excel C...	10/17/2016 11:12:27 AM
README.txt	9 KB	Text Document	10/17/2016 2:24:09 PM
tags.csv	41 KB	Microsoft Excel C...	10/17/2016 11:12:28 AM

0 B of 2.98 MB in 0 of 5

documents

/home/ubuntu/server/documents/

Name	Size	Changed	Rights	Owner
..		4/20/2018 2:13:54 PM	rwxrwxr-x	ubuntu
movies.csv	448 KB	10/17/2016 11:12:39 AM	rw-rw-r--	ubuntu
ratings.csv	2,382 KB	10/17/2016 11:12:27 AM	rw-rw-r--	ubuntu

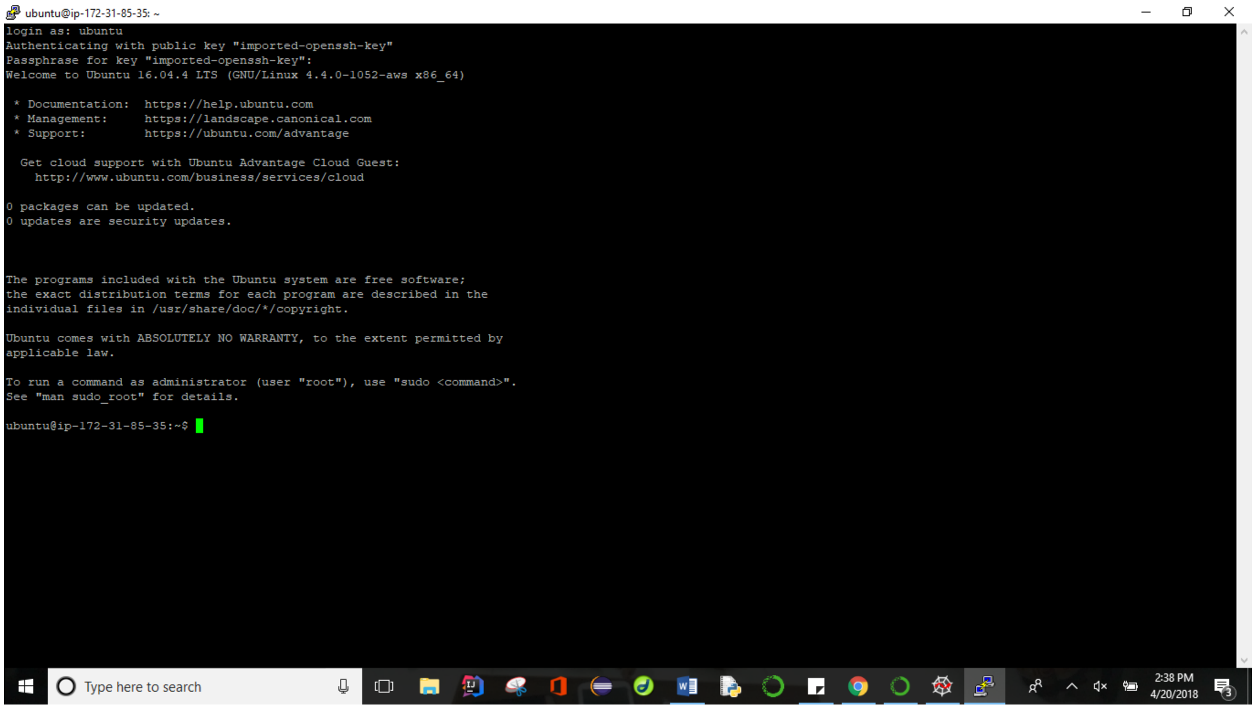
0 B of 2.76 MB in 0 of 2

SFTP-3 0:13:57

5:27 PM 4/20/2018

Converted the key document in Puttygen and uploaded in putty for the session.

Username – ubuntu ; Paraphrase - 123

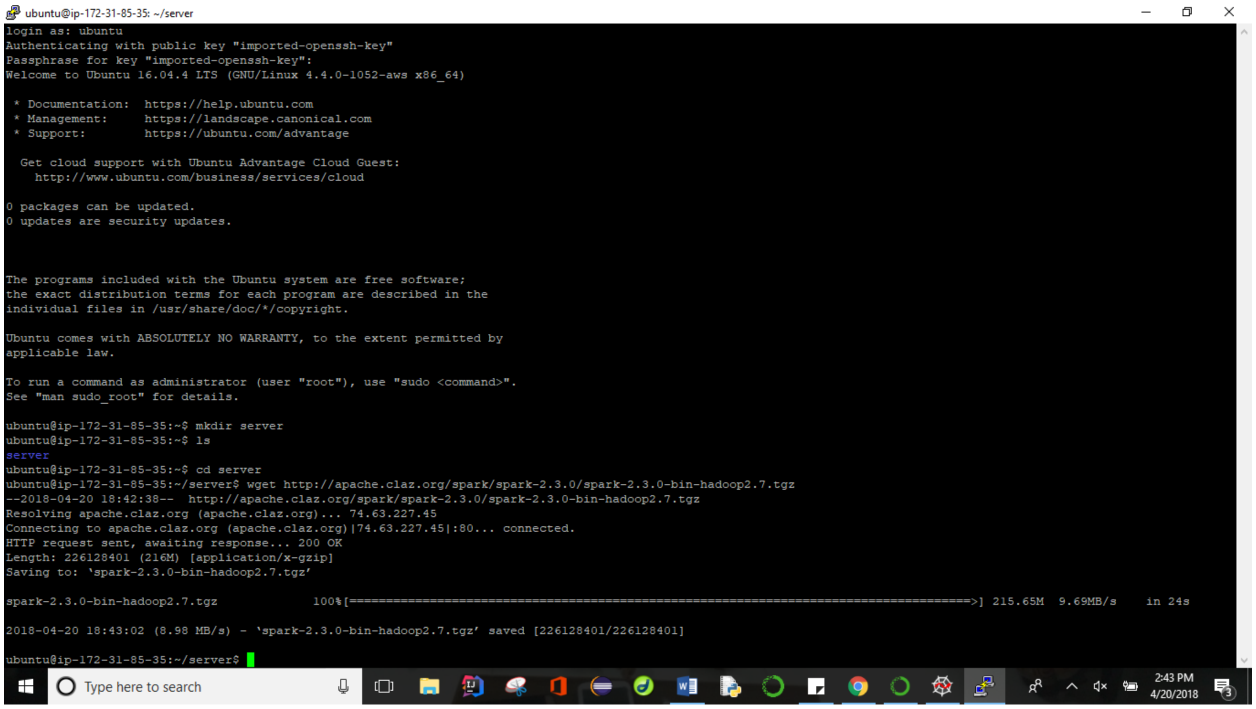


ubuntu@ip-172-31-85-35: ~

```
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1052-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-85-35:~$
```

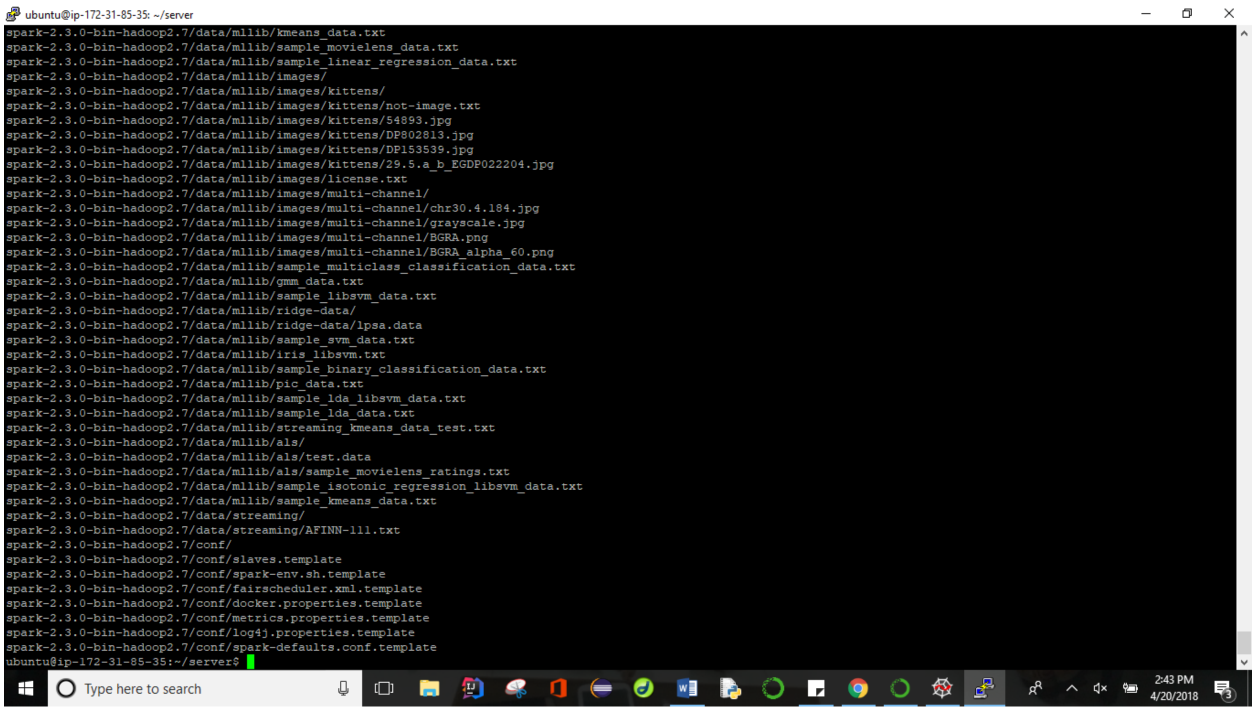
2:38 PM 4/20/2018

Installing Spark

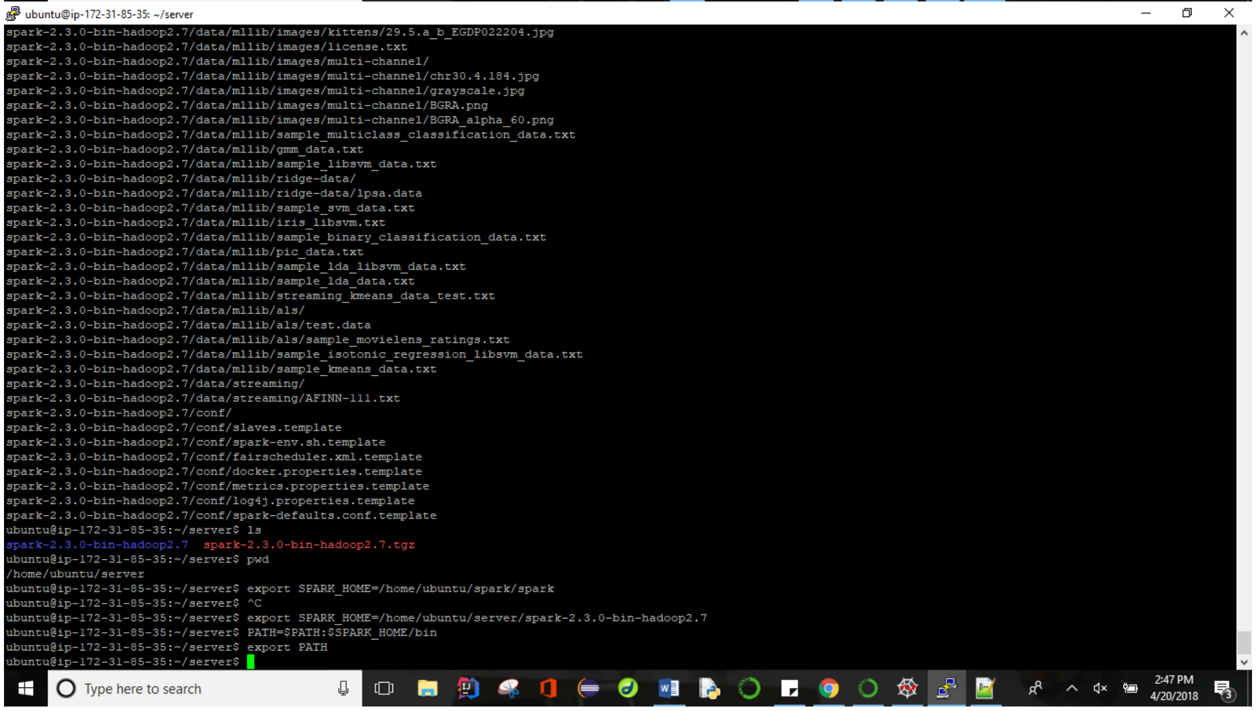


ubuntu@ip-172-31-85-35: ~/server

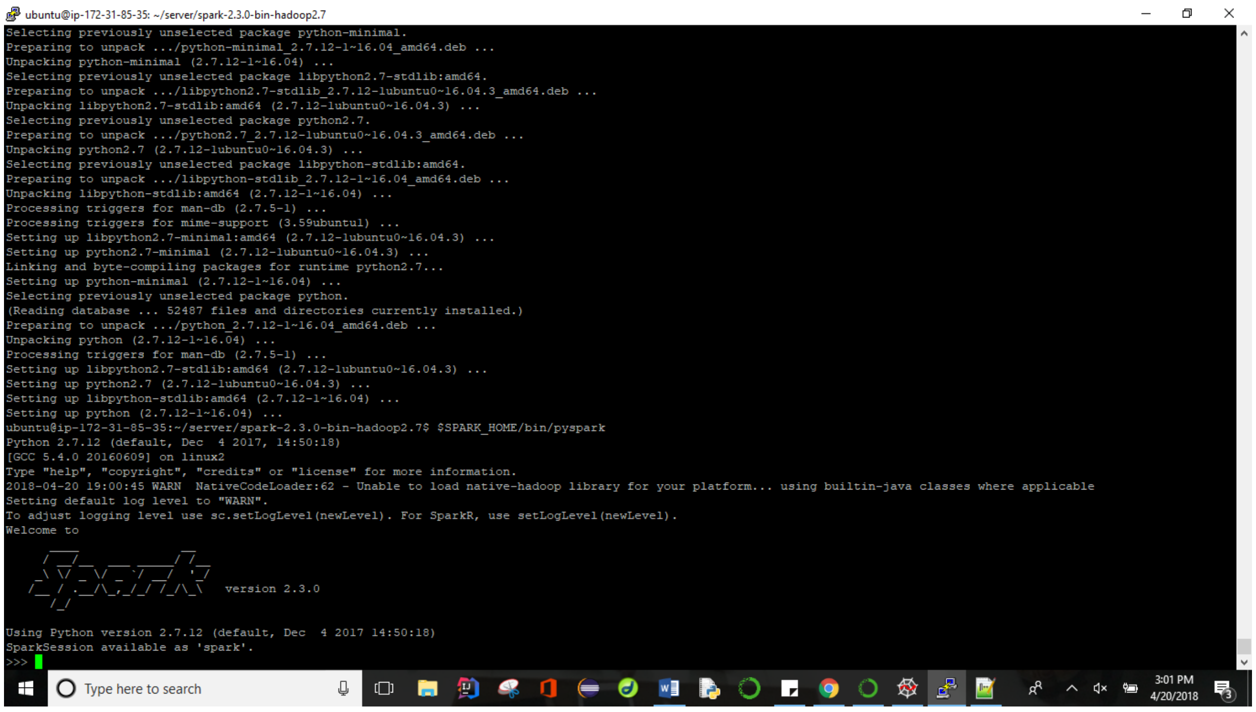
```
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1052-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-85-35:~$ mkdir server
ubuntu@ip-172-31-85-35:~$ ls
server
ubuntu@ip-172-31-85-35:~$ cd server
ubuntu@ip-172-31-85-35:~/server$ wget http://apache.claz.org/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz
--2018-04-20 18:42:38--  http://apache.claz.org/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz
Resolving apache.claz.org (apache.claz.org)... 74.63.227.45
Connecting to apache.claz.org (apache.claz.org)|74.63.227.45|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 226128401 (216M) [application/x-gzip]
Saving to: ‘spark-2.3.0-bin-hadoop2.7.tgz’

spark-2.3.0-bin-hadoop2.7.tgz             100%[=====================================================================================>] 215.65M  9.69MB/s    in 24s

2018-04-20 18:43:02 (8.98 MB/s) - ‘spark-2.3.0-bin-hadoop2.7.tgz’ saved [226128401/226128401]

ubuntu@ip-172-31-85-35:~/server$
```



```
spark-2.3.0-bin-hadoop2.7/data/mllib/kmeans_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_movielens_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_linear_regression_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/images/
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/not-image.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/54893.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/DP802813.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/DP153539.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/29.5.a_b_EGDP022204.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/license.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/chr30.4.184.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/grayscale.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/BGRA.png
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/BGRA_alpha_60.png
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_multiclass_classification_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/gmm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/ridge-data/
spark-2.3.0-bin-hadoop2.7/data/mllib/ridge-data/lpsa.data
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_svm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/iris_libsvm.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_binary_classification_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/pic_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_lda_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_lda_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/streaming_kmeans_data_test.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/als/
spark-2.3.0-bin-hadoop2.7/data/mllib/als/test.data
spark-2.3.0-bin-hadoop2.7/data/mllib/als/sample_movielens_ratings.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_isotonic_regression_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_kmeans_data.txt
spark-2.3.0-bin-hadoop2.7/data/streaming/
spark-2.3.0-bin-hadoop2.7/data/streaming/AFINN-111.txt
spark-2.3.0-bin-hadoop2.7/conf/
spark-2.3.0-bin-hadoop2.7/conf/slaves.template
spark-2.3.0-bin-hadoop2.7/conf/spark-env.sh.template
spark-2.3.0-bin-hadoop2.7/conf/fairscheduler.xml.template
spark-2.3.0-bin-hadoop2.7/conf/docker.properties.template
spark-2.3.0-bin-hadoop2.7/conf/metrics.properties.template
spark-2.3.0-bin-hadoop2.7/conf/log4j.properties.template
spark-2.3.0-bin-hadoop2.7/conf/spark-defaults.conf.template
ubuntu@ip-172-31-85-35:~/server$
```

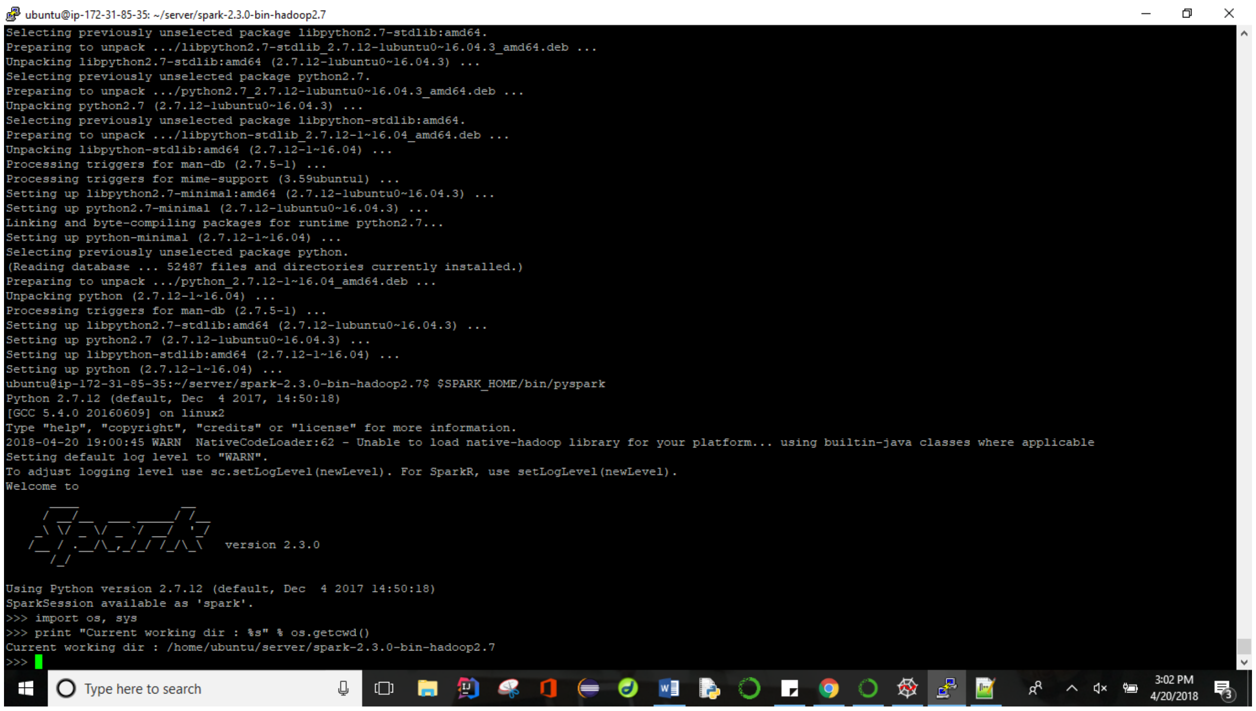


```
spark-2.3.0-bin-hadoop2.7/data/mllib/images/kittens/29.5.a_b_EGDP022204.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/license.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/chr30.4.184.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/grayscale.jpg
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/BGRA.png
spark-2.3.0-bin-hadoop2.7/data/mllib/images/multi-channel/BGRA_alpha_60.png
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_multiclass_classification_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/gmm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/ridge-data/
spark-2.3.0-bin-hadoop2.7/data/mllib/ridge-data/lpsa.data
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_svm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/iris_libsvm.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_binary_classification_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/pic_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_lda_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_lda_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/streaming_kmeans_data_test.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/als/
spark-2.3.0-bin-hadoop2.7/data/mllib/als/test.data
spark-2.3.0-bin-hadoop2.7/data/mllib/als/sample_movielens_ratings.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_isotonic_regression_libsvm_data.txt
spark-2.3.0-bin-hadoop2.7/data/mllib/sample_kmeans_data.txt
spark-2.3.0-bin-hadoop2.7/data/streaming/
spark-2.3.0-bin-hadoop2.7/data/streaming/AFINN-111.txt
spark-2.3.0-bin-hadoop2.7/conf/
spark-2.3.0-bin-hadoop2.7/conf/slaves.template
spark-2.3.0-bin-hadoop2.7/conf/spark-env.sh.template
spark-2.3.0-bin-hadoop2.7/conf/fairscheduler.xml.template
spark-2.3.0-bin-hadoop2.7/conf/docker.properties.template
spark-2.3.0-bin-hadoop2.7/conf/metrics.properties.template
spark-2.3.0-bin-hadoop2.7/conf/log4j.properties.template
spark-2.3.0-bin-hadoop2.7/conf/spark-defaults.conf.template
ubuntu@ip-172-31-85-35:~/server$ ls
spark-2.3.0-bin-hadoop2.7  spark-2.3.0-bin-hadoop2.7.tgz
ubuntu@ip-172-31-85-35:~/server$ pwd
/home/ubuntu/server
ubuntu@ip-172-31-85-35:~/server$ export SPARK_HOME=/home/ubuntu/spark/spark
ubuntu@ip-172-31-85-35:~/server$ ^C
ubuntu@ip-172-31-85-35:~/server$ export SPARK_HOME=/home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
ubuntu@ip-172-31-85-35:~/server$ PATH=$PATH:$SPARK_HOME/bin
ubuntu@ip-172-31-85-35:~/server$ export PATH
ubuntu@ip-172-31-85-35:~/server$
```



ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
Selecting previously unselected package python-minimal.
Preparing to unpack .../python-minimal_2.7.12-1~16.04_amd64.deb ...
Unpacking python-minimal (2.7.12-1~16.04) ...
Selecting previously unselected package libpython2.7-stdlib:amd64.
Preparing to unpack .../libpython2.7-stdlib_2.7.12-1ubuntu0~16.04.3_amd64.deb ...
Unpacking libpython2.7-stdlib:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../python2.7_2.7.12-1ubuntu0~16.04.3_amd64.deb ...
Unpacking python2.7 (2.7.12-1ubuntu0~16.04.3) ...
Selecting previously unselected package libpython-stdlib:amd64.
Preparing to unpack .../libpython-stdlib_2.7.12-1~16.04_amd64.deb ...
Unpacking libpython-stdlib:amd64 (2.7.12-1~16.04) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up libpython2.7-minimal:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Setting up python2.7-minimal (2.7.12-1ubuntu0~16.04.3) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python-minimal (2.7.12-1~16.04) ...
Selecting previously unselected package python.
(Reading database ... 52487 files and directories currently installed.)
Preparing to unpack .../python_2.7.12-1~16.04_amd64.deb ...
Unpacking python (2.7.12-1~16.04) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libpython2.7-stdlib:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Setting up python2.7 (2.7.12-1ubuntu0~16.04.3) ...
Setting up libpython-stdlib:amd64 (2.7.12-1~16.04) ...
Setting up python (2.7.12-1~16.04) ...
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 19:00:45 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> 
```

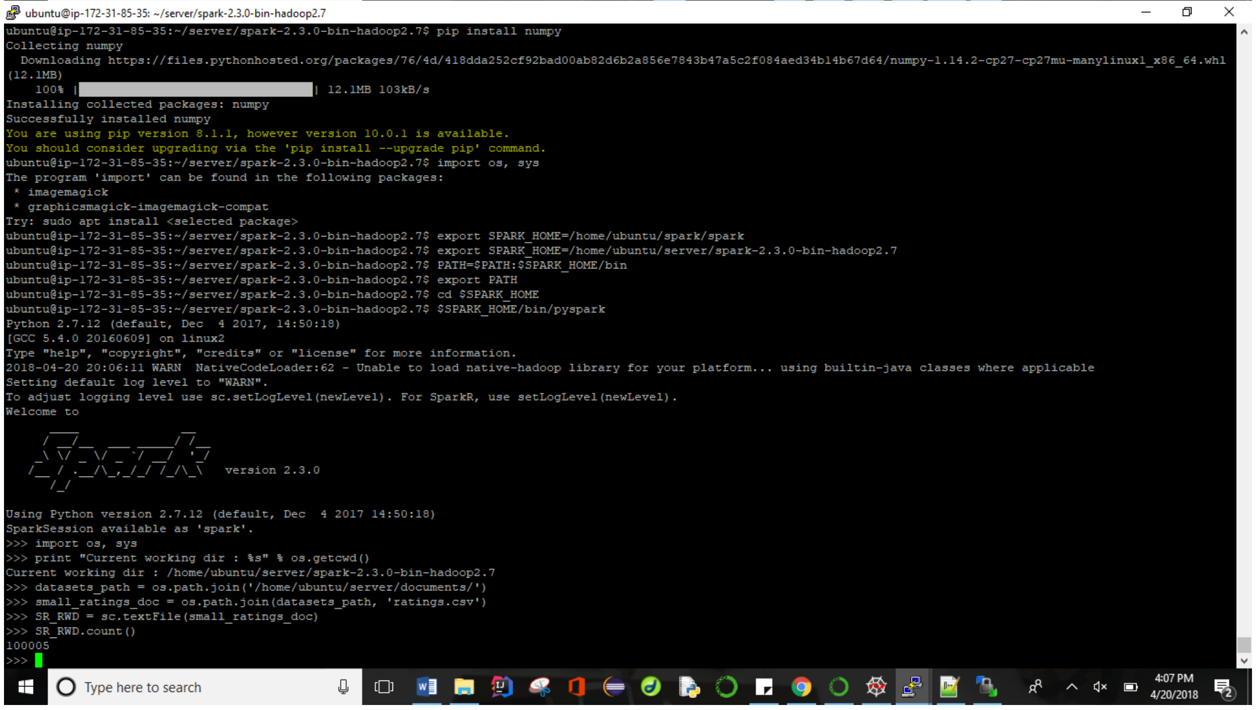


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
Selecting previously unselected package libpython2.7-stdlib:amd64.
Preparing to unpack .../libpython2.7-stdlib_2.7.12-1ubuntu0~16.04.3_amd64.deb ...
Unpacking libpython2.7-stdlib:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../python2.7_2.7.12-1ubuntu0~16.04.3_amd64.deb ...
Unpacking python2.7 (2.7.12-1ubuntu0~16.04.3) ...
Selecting previously unselected package libpython-stdlib:amd64.
Preparing to unpack .../libpython-stdlib_2.7.12-1~16.04_amd64.deb ...
Unpacking libpython-stdlib:amd64 (2.7.12-1~16.04) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up libpython2.7-minimal:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Setting up python2.7-minimal (2.7.12-1ubuntu0~16.04.3) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python-minimal (2.7.12-1~16.04) ...
Selecting previously unselected package python.
(Reading database ... 52487 files and directories currently installed.)
Preparing to unpack .../python_2.7.12-1~16.04_amd64.deb ...
Unpacking python (2.7.12-1~16.04) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libpython2.7-stdlib:amd64 (2.7.12-1ubuntu0~16.04.3) ...
Setting up python2.7 (2.7.12-1ubuntu0~16.04.3) ...
Setting up libpython-stdlib:amd64 (2.7.12-1~16.04) ...
Setting up python (2.7.12-1~16.04) ...
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 19:00:45 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>>
```

Type here to search

3:02 PM
4/20/2018

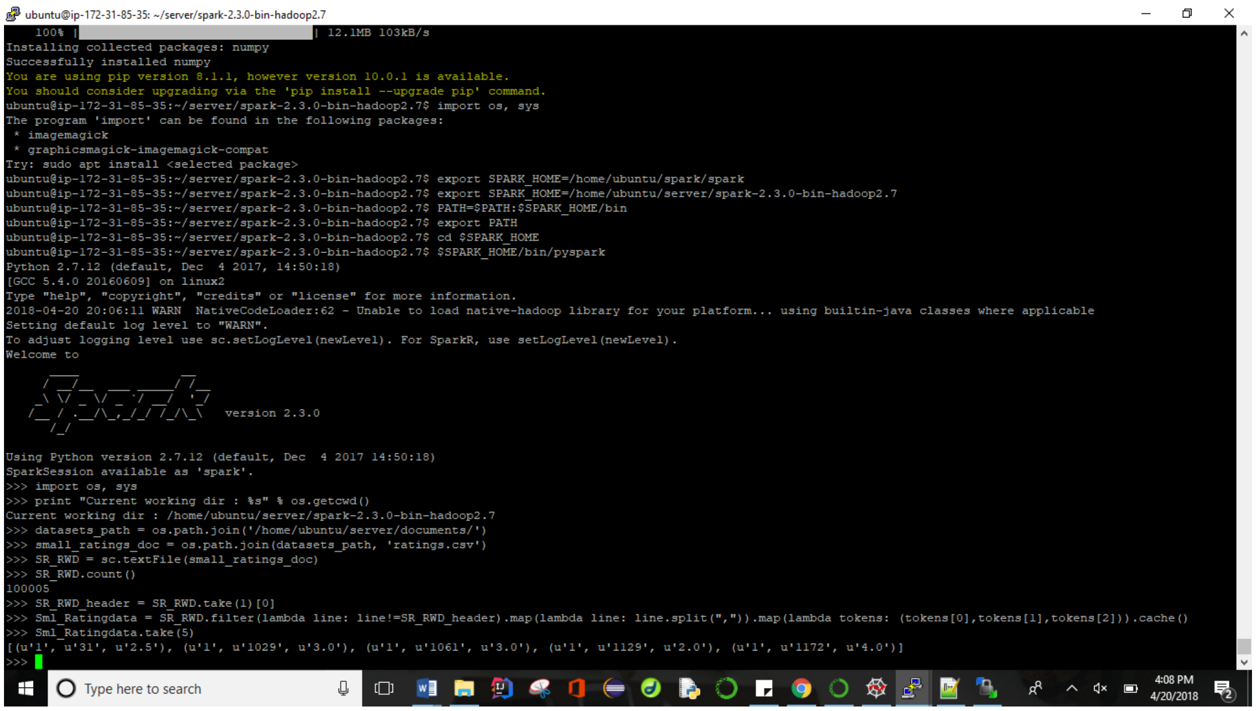


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/76/4d/418dda252cf92bad00ab82d6b2a856e7843b47a5c2f084aed34b14b67d64/numpy-1.14.2-cp27-cp27mu-manylinux1_x86_64.whl
(12.1MB)
    100% |████████████████████████████████| 12.1MB 103kB/s
Installing collected packages: numpy
Successfully installed numpy
You are using pip version 8.1.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ import os, sys
The program 'import' can be found in the following packages:
 * imagemagick
 * graphicsmagick-imagemagick-compat
Try: sudo apt install <selected package>
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/spark/spark
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ PATH=$PATH:$SPARK_HOME/bin
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export PATH
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ cd $SPARK_HOME
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 20:06:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>> datasets_path = os.path.join('/home/ubuntu/server/documents/')
>>> small_ratings_doc = os.path.join(datasets_path, 'ratings.csv')
>>> SR_RWD = sc.textFile(small_ratings_doc)
>>> SR_RWD.count()
100005
>>>
```

Type here to search

4:07 PM
4/20/2018

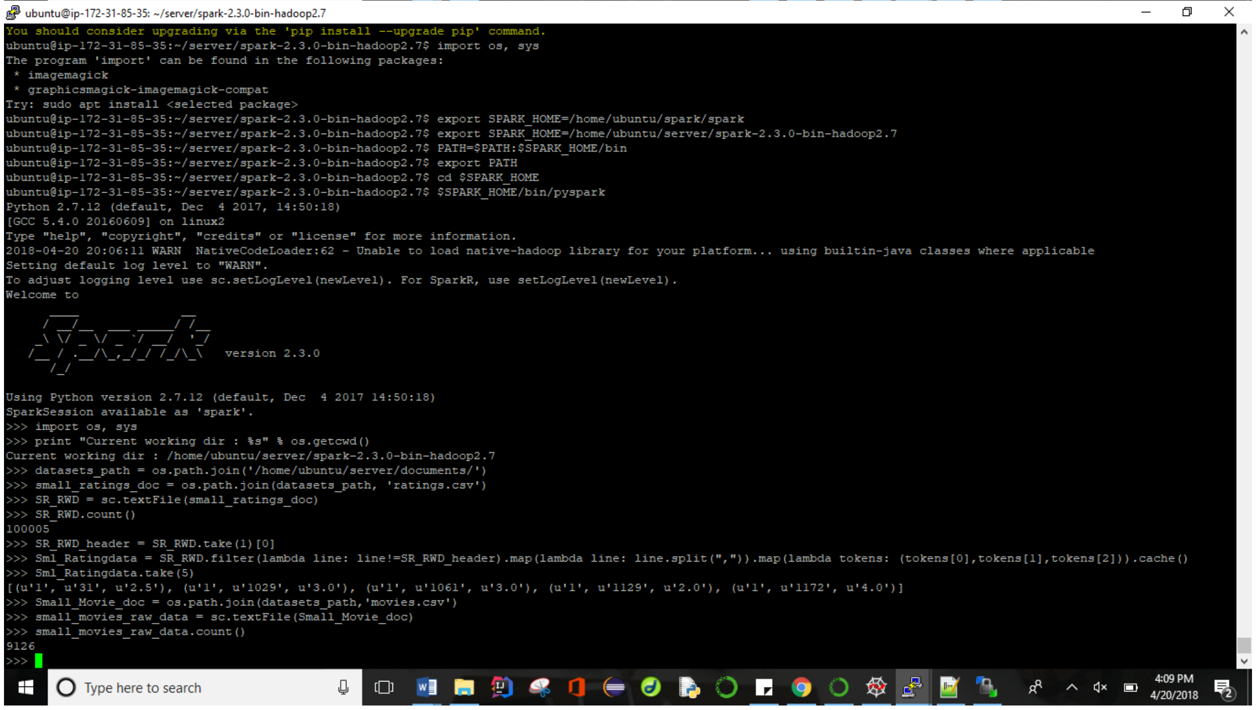


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
    100% |████████████████████████████████| 12.1MB 103kB/s
Installing collected packages: numpy
Successfully installed numpy
You are using pip version 8.1.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ import os, sys
The program 'import' can be found in the following packages:
 * imagemagick
 * graphicsmagick-imagemagick-compat
Try: sudo apt install <selected package>
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/spark/spark
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ PATH=$PATH:$SPARK_HOME/bin
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export PATH
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ cd $SPARK_HOME
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 20:06:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>> datasets_path = os.path.join('/home/ubuntu/server/documents/')
>>> small_ratings_doc = os.path.join(datasets_path, 'ratings.csv')
>>> SR_RWD = sc.textFile(small_ratings_doc)
>>> SR_RWD.count()
100005
>>> SR_RWD_header = SR_RWD.take(1)[0]
>>> Sml_Ratingdata = SR_RWD.filter(lambda line: line!=SR_RWD_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],tokens[1],tokens[2])).cache()
>>> Sml_Ratingdata.take(5)
[(u'1', u'31', u'2.5'), (u'1', u'1029', u'3.0'), (u'1', u'1061', u'3.0'), (u'1', u'1129', u'2.0'), (u'1', u'1172', u'4.0')]
>>>
```

Type here to search

4:08 PM
4/20/2018

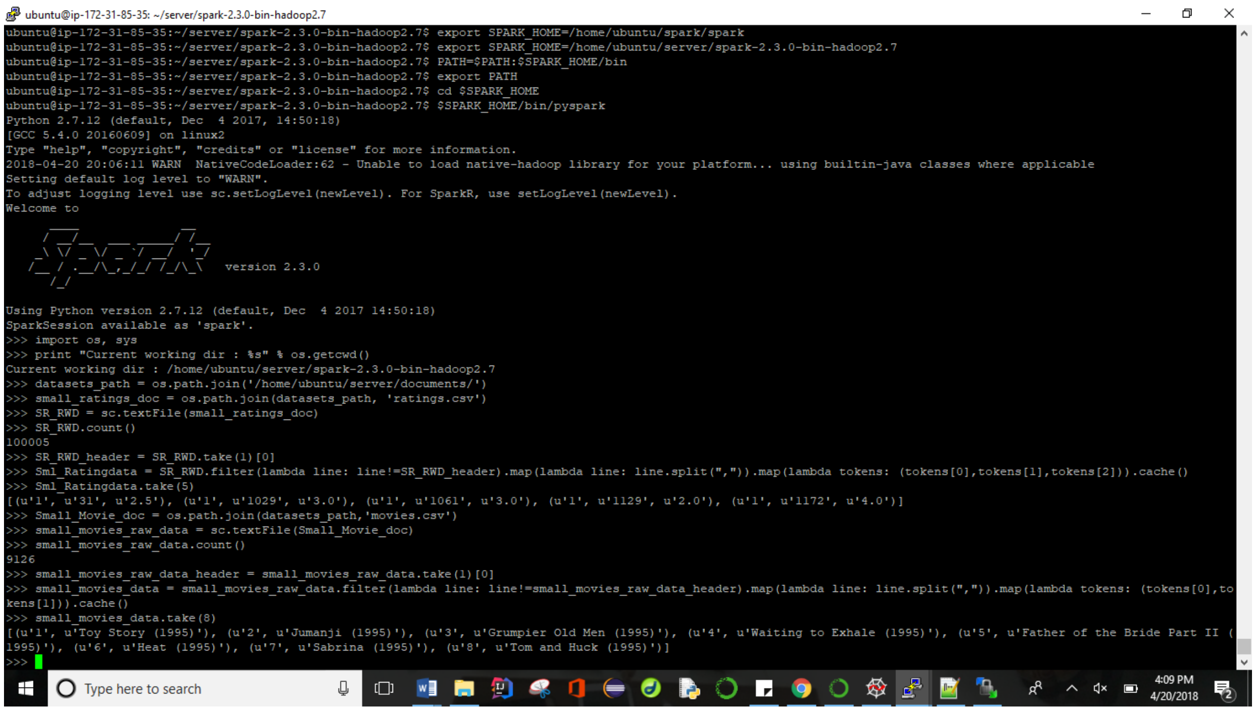


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

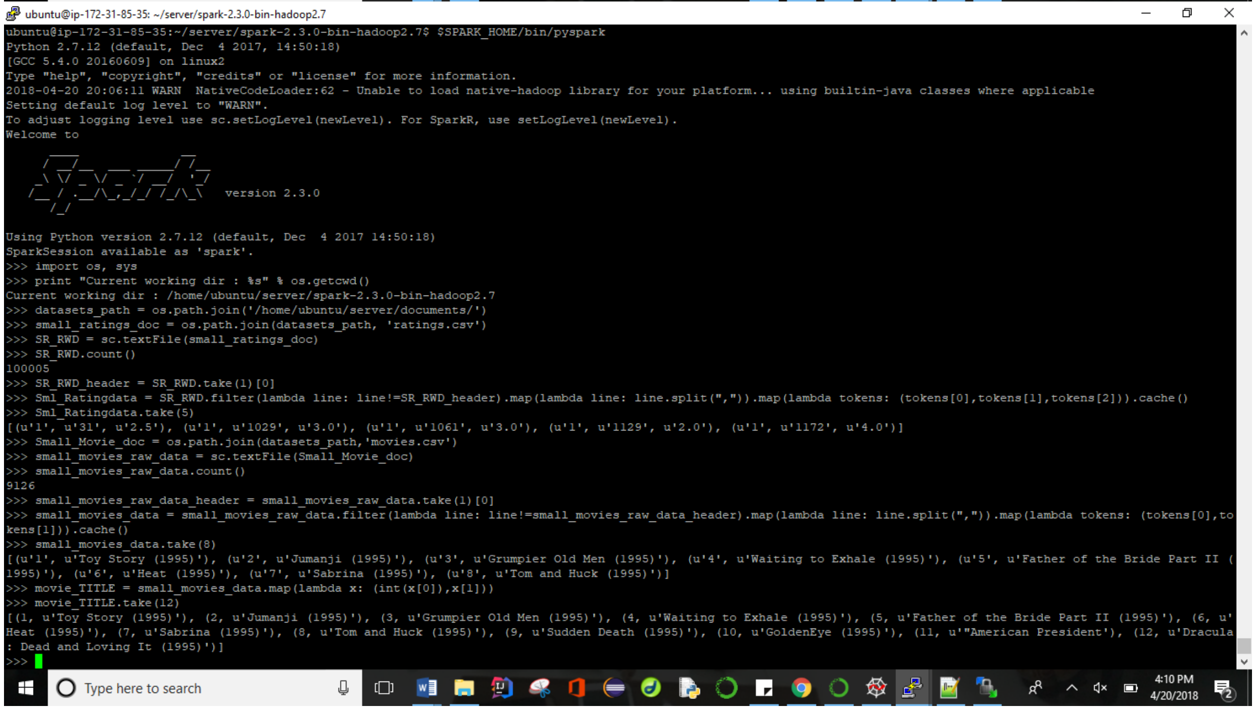
```
You should consider upgrading via the 'pip install --upgrade pip' command.
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ import os, sys
The program 'import' can be found in the following packages:
 * imagemagick
 * graphicsmagick-imagemagick-compat
Try: sudo apt install <selected package>
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/spark/spark
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ PATH=$PATH:$SPARK_HOME/bin
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export PATH
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ cd $SPARK_HOME
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 20:06:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>> datasets_path = os.path.join('/home/ubuntu/server/documents/')
>>> small_ratings_doc = os.path.join(datasets_path, 'ratings.csv')
>>> SR_RWD = sc.textFile(small_ratings_doc)
>>> SR_RWD.count()
100005
>>> SR_RWD_header = SR_RWD.take(1)[0]
>>> Sml_Ratingdata = SR_RWD.filter(lambda line: line!=SR_RWD_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],tokens[1],tokens[2])).cache()
>>> Sml_Ratingdata.take(5)
[(u'1', u'31', u'2.5'), (u'1', u'1029', u'3.0'), (u'1', u'1061', u'3.0'), (u'1', u'1129', u'2.0'), (u'1', u'1172', u'4.0')]
>>> Small_Movie_doc = os.path.join(datasets_path,'movies.csv')
>>> small_movies_raw_data = sc.textFile(Small_Movie_doc)
>>> small_movies_raw_data.count()
9126
>>>
```

Type here to search

4:09 PM
4/20/2018

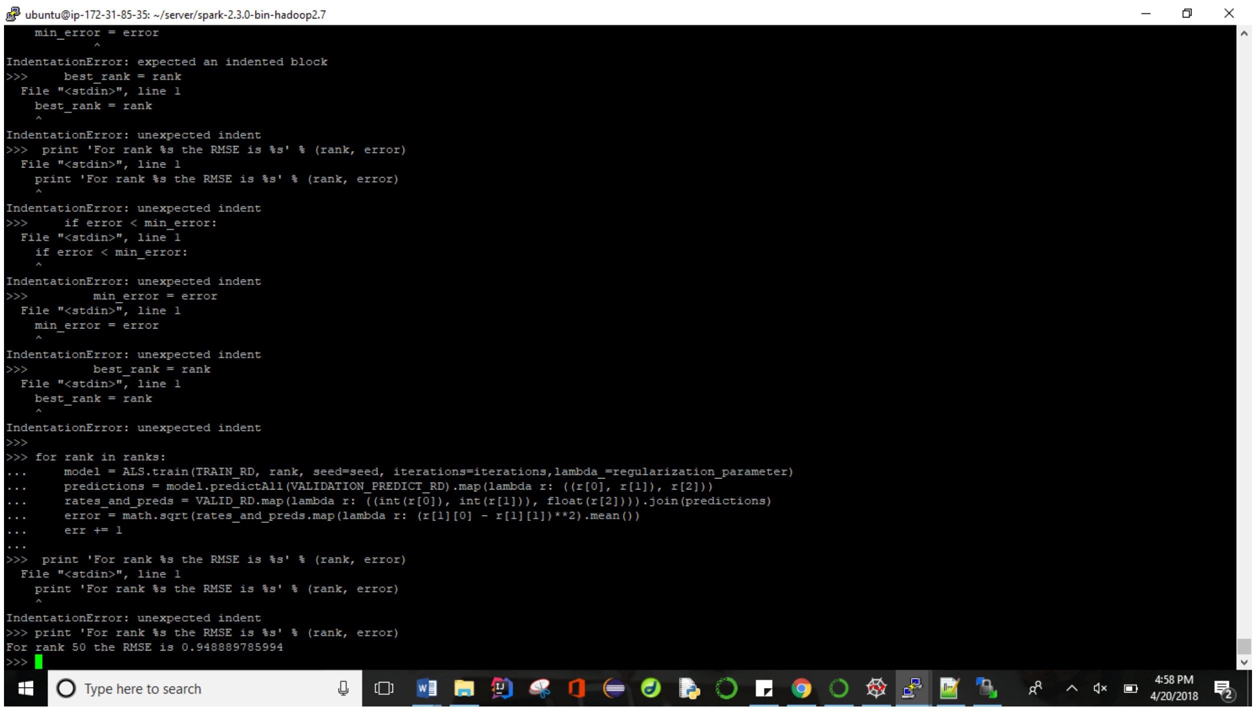

ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/spark/spark
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export SPARK_HOME=/home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ PATH=$PATH:$SPARK_HOME/bin
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ export PATH
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ cd $SPARK_HOME
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 20:06:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>> datasets_path = os.path.join('/home/ubuntu/server/documents/')
>>> small_ratings_doc = os.path.join(datasets_path, 'ratings.csv')
>>> SR_RWD = sc.textFile(small_ratings_doc)
>>> SR_RWD.count()
100005
>>> SR_RWD_header = SR_RWD.take(1)[0]
>>> Sml_Ratingdata = SR_RWD.filter(lambda line: line!=SR_RWD_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],tokens[1],tokens[2])).cache()
>>> Sml_Ratingdata.take(5)
[(u'1', u'31', u'2.5'), (u'1', u'1029', u'3.0'), (u'1', u'1061', u'3.0'), (u'1', u'1129', u'2.0'), (u'1', u'1172', u'4.0')]
>>> Small_Movie_doc = os.path.join(datasets_path,'movies.csv')
>>> small_movies_raw_data = sc.textFile(Small_Movie_doc)
>>> small_movies_raw_data.count()
9126
>>> small_movies_raw_data_header = small_movies_raw_data.take(1)[0]
>>> small_movies_data = small_movies_raw_data.filter(lambda line: line!=small_movies_raw_data_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],to
kens[1])).cache()
>>> small_movies_data.take(8)
[(u'1', u'Toy Story (1995)'), (u'2', u'Jumanji (1995)'), (u'3', u'Grumpier Old Men (1995)'), (u'4', u'Waiting to Exhale (1995)'), (u'5', u'Father of the Bride Part II (
1995)'), (u'6', u'Heat (1995)'), (u'7', u'Sabrina (1995)'), (u'8', u'Tom and Huck (1995)')]
>>>
```


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
ubuntu@ip-172-31-85-35:~/server/spark-2.3.0-bin-hadoop2.7$ $SPARK_HOME/bin/pyspark
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-04-20 20:06:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.12 (default, Dec  4 2017 14:50:18)
SparkSession available as 'spark'.
>>> import os, sys
>>> print "Current working dir : %s" % os.getcwd()
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7
>>> datasets_path = os.path.join('/home/ubuntu/server/documents/')
>>> small_ratings_doc = os.path.join(datasets_path, 'ratings.csv')
>>> SR_RWD = sc.textFile(small_ratings_doc)
>>> SR_RWD.count()
100005
>>> SR_RWD_header = SR_RWD.take(1)[0]
>>> Sml_Ratingdata = SR_RWD.filter(lambda line: line!=SR_RWD_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],tokens[1],tokens[2])).cache()
>>> Sml_Ratingdata.take(5)
[(u'1', u'31', u'2.5'), (u'1', u'1029', u'3.0'), (u'1', u'1061', u'3.0'), (u'1', u'1129', u'2.0'), (u'1', u'1172', u'4.0')]
>>> Small_Movie_doc = os.path.join(datasets_path,'movies.csv')
>>> small_movies_raw_data = sc.textFile(Small_Movie_doc)
>>> small_movies_raw_data.count()
9126
>>> small_movies_raw_data_header = small_movies_raw_data.take(1)[0]
>>> small_movies_data = small_movies_raw_data.filter(lambda line: line!=small_movies_raw_data_header).map(lambda line: line.split(",")).map(lambda tokens: (tokens[0],to
kens[1])).cache()
>>> small_movies_data.take(8)
[(u'1', u'Toy Story (1995)'), (u'2', u'Jumanji (1995)'), (u'3', u'Grumpier Old Men (1995)'), (u'4', u'Waiting to Exhale (1995)'), (u'5', u'Father of the Bride Part II (
1995)'), (u'6', u'Heat (1995)'), (u'7', u'Sabrina (1995)'), (u'8', u'Tom and Huck (1995)')]
```

```
>>> movie_TITLE = small_movies_data.map(lambda x: (int(x[0]),x[1]))
>>> movie_TITLE.take(12)
[(1, u'Toy Story (1995)'), (2, u'Jumanji (1995)'), (3, u'Grumpier Old Men (1995)'), (4, u'Waiting to Exhale (1995)'), (5, u'Father of the Bride Part II (1995)'), (6, u'
Heat (1995)'), (7, u'Sabrina (1995)'), (8, u'Tom and Huck (1995)'), (9, u'Sudden Death (1995)'), (10, u'GoldenEye (1995)'), (11, u'"American President'), (12, u'Dracula
: Dead and Loving It (1995)')]
>>>
```

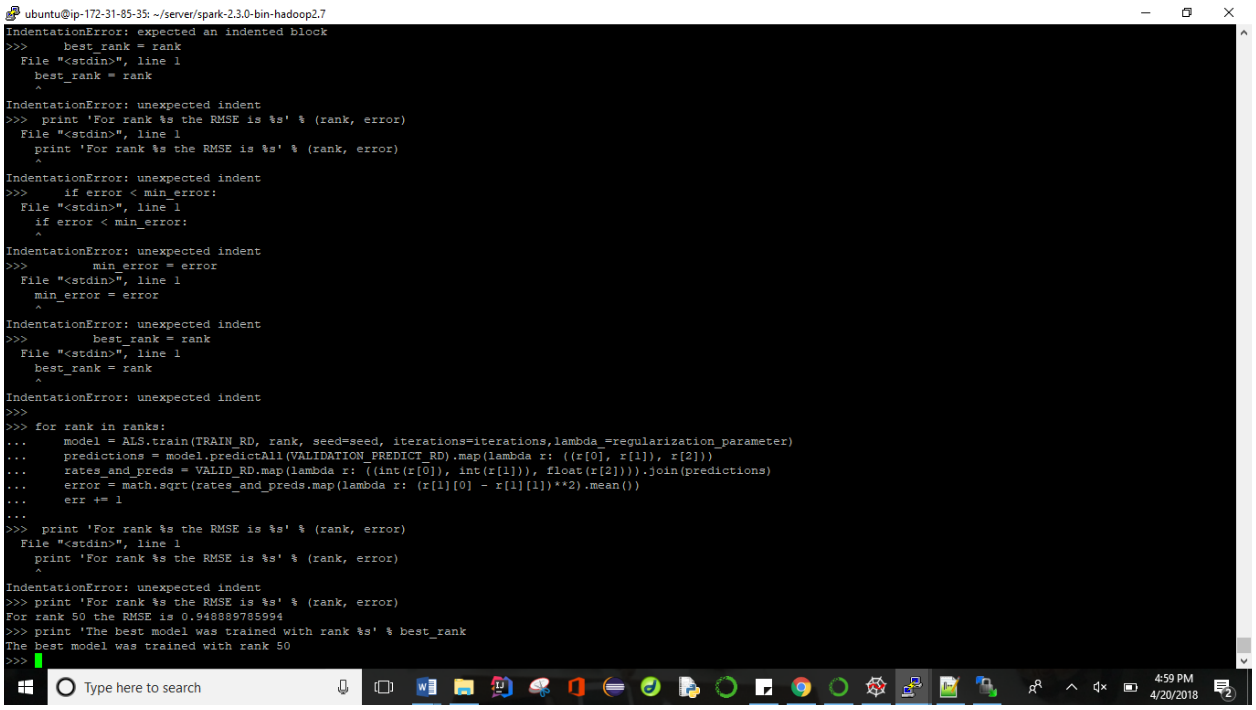


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
    min_error = error
            ^
IndentationError: expected an indented block
>>>     best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>>     if error < min_error:
  File "<stdin>", line 1
    if error < min_error:
    ^
IndentationError: unexpected indent
>>>         min_error = error
  File "<stdin>", line 1
    min_error = error
    ^
IndentationError: unexpected indent
>>>         best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>>
```

Type here to search

4:58 PM
4/20/2018

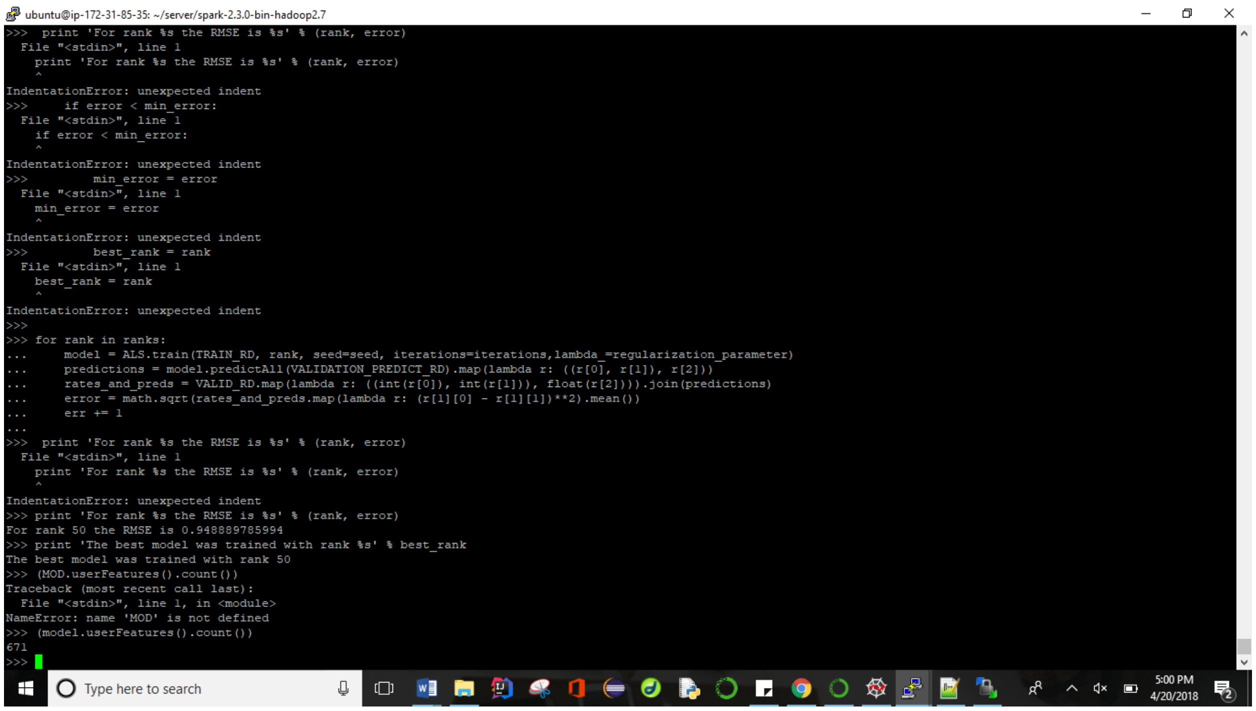


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
IndentationError: expected an indented block
>>>     best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>>     if error < min_error:
  File "<stdin>", line 1
    if error < min_error:
    ^
IndentationError: unexpected indent
>>>         min_error = error
  File "<stdin>", line 1
    min_error = error
    ^
IndentationError: unexpected indent
>>>         best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>> print 'The best model was trained with rank %s' % best_rank
The best model was trained with rank 50
>>>
```

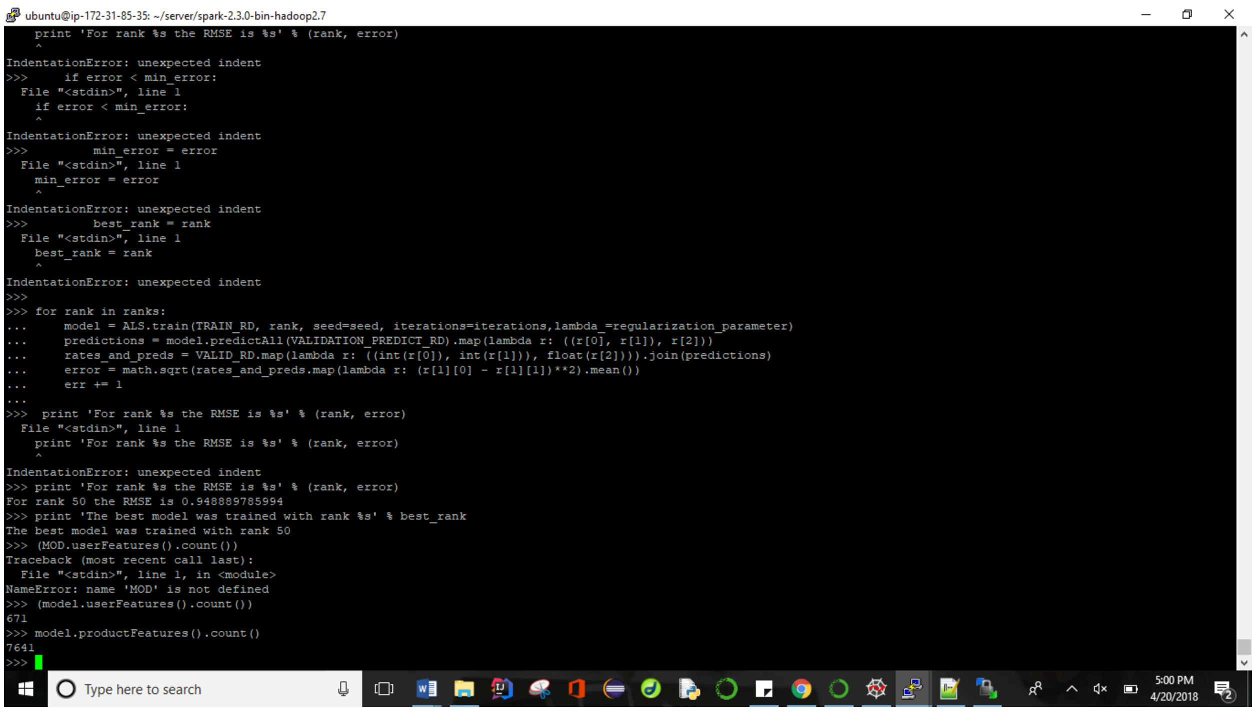
Type here to search

4:59 PM
4/20/2018



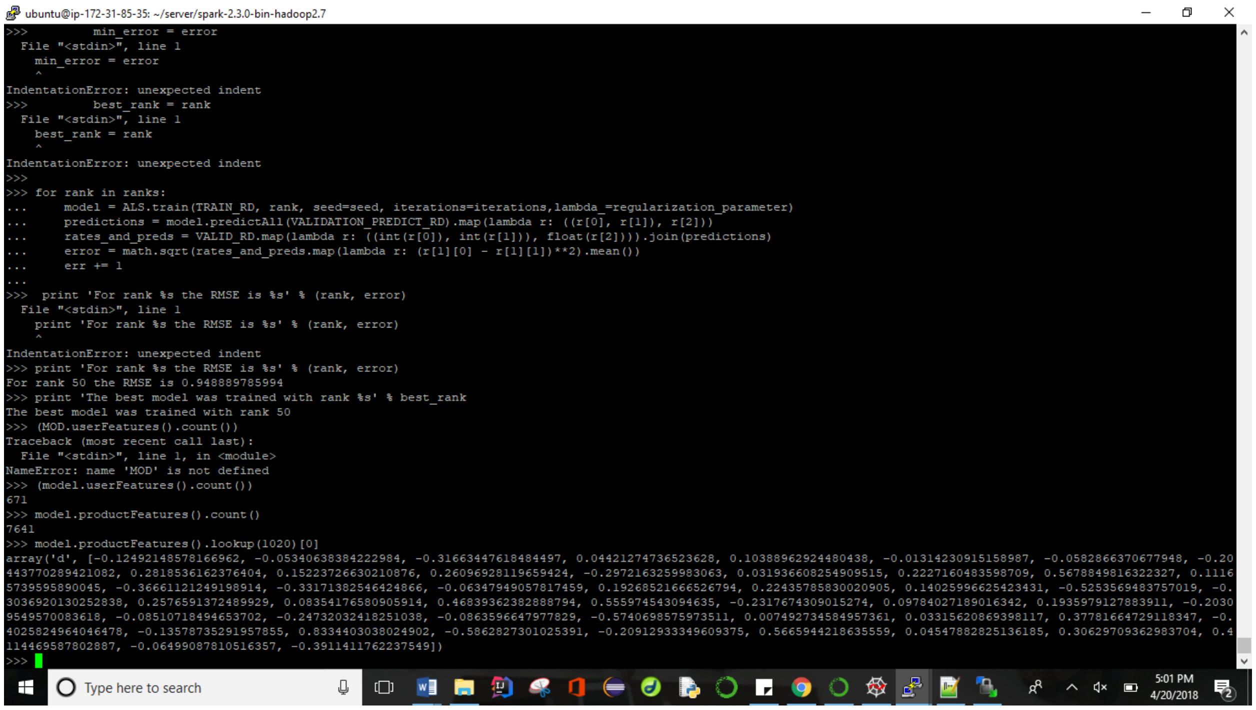
ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>>     if error < min_error:
  File "<stdin>", line 1
    if error < min_error:
    ^
IndentationError: unexpected indent
>>>         min_error = error
  File "<stdin>", line 1
    min_error = error
    ^
IndentationError: unexpected indent
>>>         best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>> print 'The best model was trained with rank %s' % best_rank
The best model was trained with rank 50
>>> (MOD.userFeatures().count())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'MOD' is not defined
>>> (model.userFeatures().count())
671
>>>
```

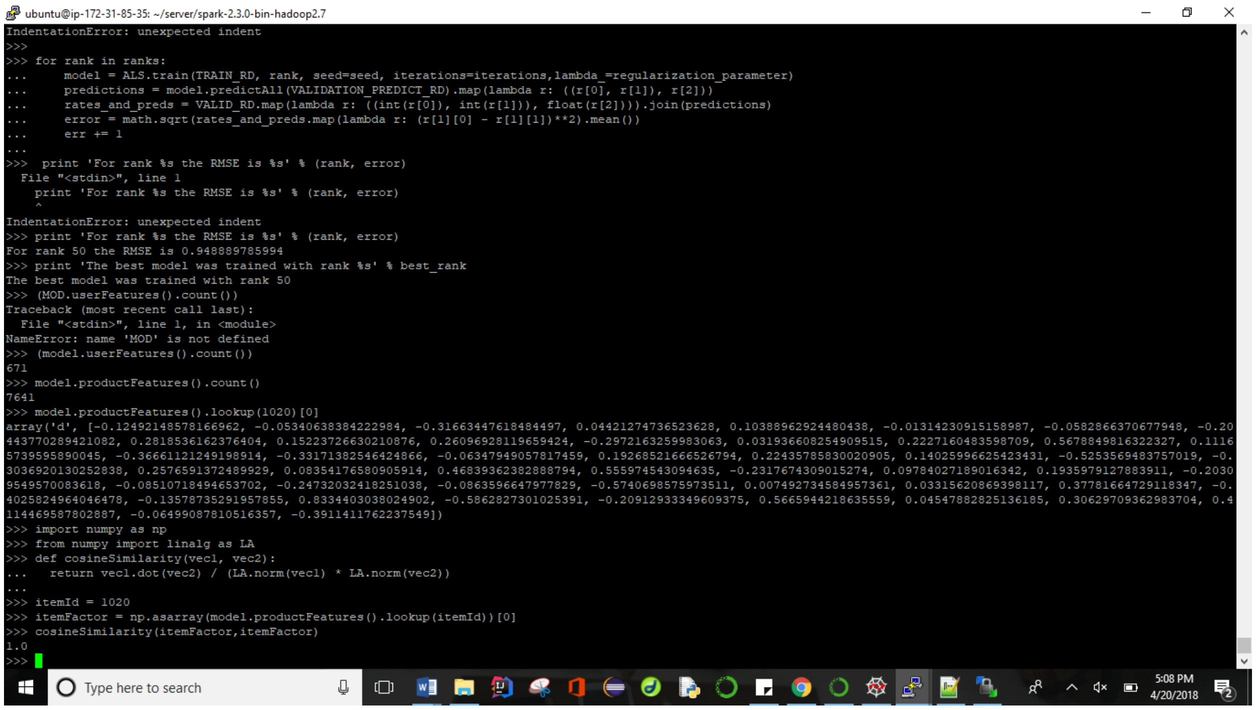


ubuntu@ip-172-31-85-35: ~/server/spark-2.3.0-bin-hadoop2.7

```
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>>     if error < min_error:
  File "<stdin>", line 1
    if error < min_error:
    ^
IndentationError: unexpected indent
>>>         min_error = error
  File "<stdin>", line 1
    min_error = error
    ^
IndentationError: unexpected indent
>>>         best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>> print 'The best model was trained with rank %s' % best_rank
The best model was trained with rank 50
>>> (MOD.userFeatures().count())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'MOD' is not defined
>>> (model.userFeatures().count())
671
>>> model.productFeatures().count()
7641
>>>
```

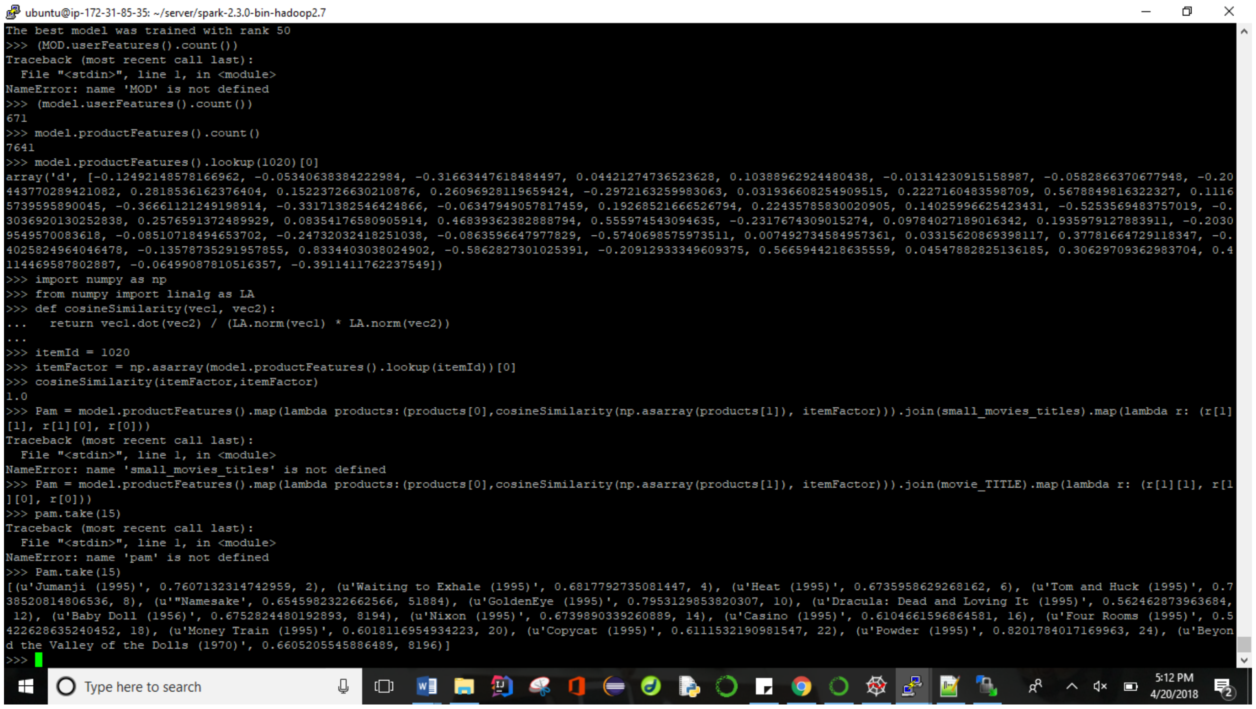


```
>>>         min_error = error
  File "<stdin>", line 1
    min_error = error
    ^
IndentationError: unexpected indent
>>>         best_rank = rank
  File "<stdin>", line 1
    best_rank = rank
    ^
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>> print 'The best model was trained with rank %s' % best_rank
The best model was trained with rank 50
>>> (MOD.userFeatures().count())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'MOD' is not defined
>>> (model.userFeatures().count())
671
>>> model.productFeatures().count()
7641
>>> model.productFeatures().lookup(1020)[0]
array('d', [-0.12492148578166962, -0.05340638384222984, -0.31663447618484497, 0.04421274736523628, 0.10388962924480438, -0.01314230915158987, -0.0582866370677948, -0.20
443770289421082, 0.2818536162376404, 0.15223726630210876, 0.26096928119659424, -0.2972163259983063, 0.031936608254909515, 0.2227160483598709, 0.5678849816322327, 0.1116
5739595890045, -0.36661121249198914, -0.33171382546424866, -0.06347949057817459, 0.19268521666526794, 0.22435785830020905, 0.14025996625423431, -0.5253569483757019, -0.
3036920130252838, 0.2576591372489929, 0.08354176580905914, 0.46839362382888794, 0.555974543094635, -0.2317674309015274, 0.09784027189016342, 0.1935979127883911, -0.2030
9549570083618, -0.08510718494653702, -0.24732032418251038, -0.08635966479777829, -0.5740698575973511, 0.007492734584957361, 0.03315620869398117, 0.37781664729118347, -0.
4025824964046478, -0.13578735291957855, 0.8334403038024902, -0.5862827301025391, -0.20912933349609375, 0.5665944218635559, 0.04547882825136185, 0.30629709362983704, 0.4
114469587802887, -0.06499087810516357, -0.3911411762237549])
>>>
```



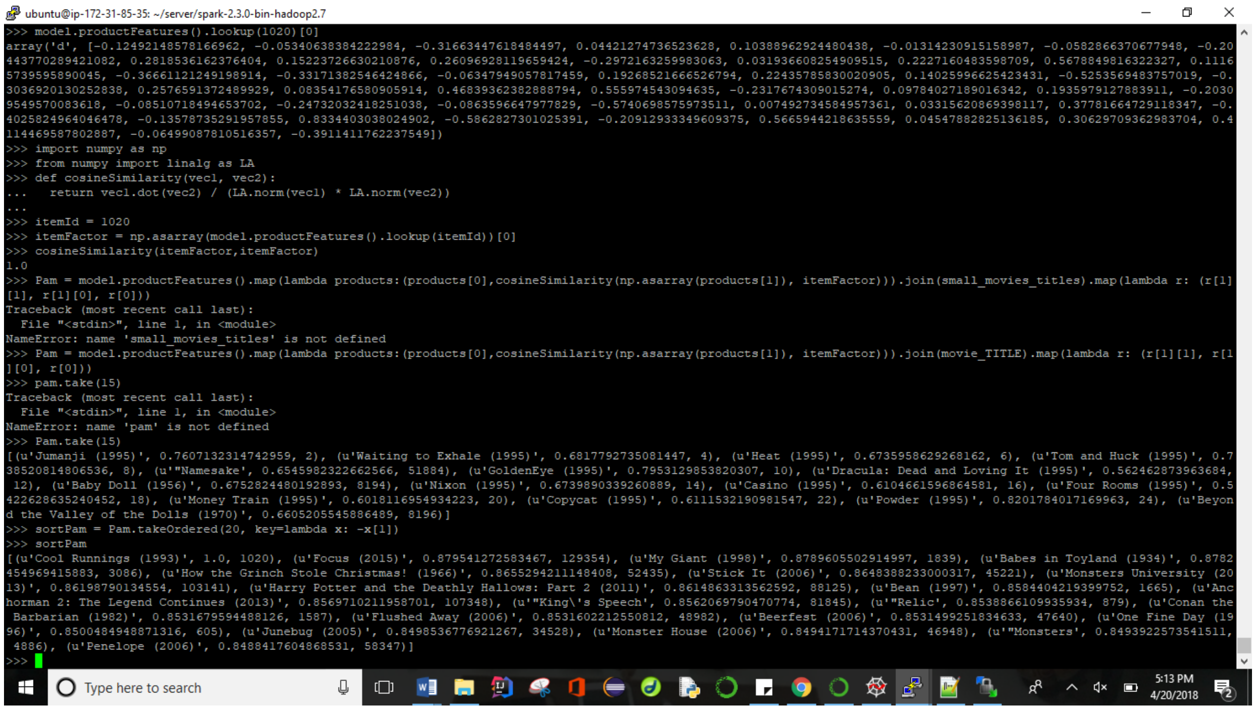
```
IndentationError: unexpected indent
>>>
>>> for rank in ranks:
...     model = ALS.train(TRAIN_RD, rank, seed=seed, iterations=iterations,lambda_=regularization_parameter)
...     predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
...     rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
...     error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
...     err += 1
...
>>>  print 'For rank %s the RMSE is %s' % (rank, error)
  File "<stdin>", line 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    ^
IndentationError: unexpected indent
>>> print 'For rank %s the RMSE is %s' % (rank, error)
For rank 50 the RMSE is 0.948889785994
>>> print 'The best model was trained with rank %s' % best_rank
The best model was trained with rank 50
>>> (MOD.userFeatures().count())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'MOD' is not defined
>>> (model.userFeatures().count())
671
>>> model.productFeatures().count()
7641
>>> model.productFeatures().lookup(1020)[0]
array('d', [-0.12492148578166962, -0.05340638384222984, -0.31663447618484497, 0.04421274736523628, 0.10388962924480438, -0.01314230915158987, -0.0582866370677948, -0.20
443770289421082, 0.2818536162376404, 0.15223726630210876, 0.26096928119659424, -0.2972163259983063, 0.031936608254909515, 0.2227160483598709, 0.5678849816322327, 0.1116
5739595890045, -0.36661121249198914, -0.33171382546424866, -0.06347949057817459, 0.19268521666526794, 0.22435785830020905, 0.14025996625423431, -0.5253569483757019, -0.
3036920130252838, 0.2576591372489929, 0.08354176580905914, 0.46839362382888794, 0.555974543094635, -0.2317674309015274, 0.09784027189016342, 0.1935979127883911, -0.2030
9549570083618, -0.08510718494653702, -0.24732032418251038, -0.08635966479777829, -0.5740698575973511, 0.007492734584957361, 0.03315620869398117, 0.37781664729118347, -0.
4025824964046478, -0.13578735291957855, 0.8334403038024902, -0.5862827301025391, -0.20912933349609375, 0.5665944218635559, 0.04547882825136185, 0.30629709362983704, 0.4
114469587802887, -0.06499087810516357, -0.3911411762237549])
```

```
>>> import numpy as np
>>> from numpy import linalg as LA
>>> def cosineSimilarity(vec1, vec2):
...   return vec1.dot(vec2) / (LA.norm(vec1) * LA.norm(vec2))
...
>>> itemId = 1020
>>> itemFactor = np.asarray(model.productFeatures().lookup(itemId))[0]
>>> cosineSimilarity(itemFactor,itemFactor)
1.0
>>>
```



```
The best model was trained with rank 50
>>> (MOD.userFeatures().count())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'MOD' is not defined
>>> (model.userFeatures().count())
671
>>> model.productFeatures().count()
7641
>>> model.productFeatures().lookup(1020)[0]
array('d', [-0.12492148578166962, -0.05340638384222984, -0.31663447618484497, 0.04421274736523628, 0.10388962924480438, -0.01314230915158987, -0.0582866370677948, -0.20
443770289421082, 0.2818536162376404, 0.15223726630210876, 0.26096928119659424, -0.2972163259983063, 0.031936608254909515, 0.2227160483598709, 0.5678849816322327, 0.1116
5739595890045, -0.3666112124919891 4, -0.33171382546424866, -0.06347949057817459, 0.19268521666526794, 0.22435785830020905, 0.14025996625423431, -0.5253569483757019, -0.
3036920130252838, 0.2576591372489929, 0.08354176580905914, 0.46839362382888794, 0.555974543094635, -0.2317674309015274, 0.09784027189016342, 0.1935979127883911, -0.2030
9549570083618, -0.08510718494653702, -0.24732032418251038, -0.0863596647977829, -0.5740698575973511, 0.007492734584957361, 0.03315620869398117, 0.37781664729118347, -0.
4025824964046478, -0.13578735291957855, 0.8334403038024902, -0.5862827301025391, -0.20912933349609375, 0.5665944218635559, 0.04547882825136185, 0.30629709362983704, 0.4
114469587802887, -0.06499087810516357, -0.3911411762237549])
>>> import numpy as np
>>> from numpy import linalg as LA
>>> def cosineSimilarity(vec1, vec2):
...   return vec1.dot(vec2) / (LA.norm(vec1) * LA.norm(vec2))
...
>>> itemId = 1020
>>> itemFactor = np.asarray(model.productFeatures().lookup(itemId))[0]
>>> cosineSimilarity(itemFactor,itemFactor)
1.0
>>> Pam = model.productFeatures().map(lambda products:(products[0],cosineSimilarity(np.asarray(products[1]), itemFactor))).join(small_movies_titles).map(lambda r: (r[1]
[1], r[1][0], r[0]))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'small_movies_titles' is not defined
>>> Pam = model.productFeatures().map(lambda products:(products[0],cosineSimilarity(np.asarray(products[1]), itemFactor))).join(movie_TITLE).map(lambda r: (r[1][1], r[1
][0], r[0]))
>>> pam.take(15)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pam' is not defined
>>> Pam.take(15)
[(u'Jumanji (1995)', 0.7607132314742959, 2), (u'Waiting to Exhale (1995)', 0.6817792735081447, 4), (u'Heat (1995)', 0.6735958629268162, 6), (u'Tom and Huck (1995)', 0.7
38520814806536, 8), (u'"Namesake', 0.6545982322662566, 51884), (u'GoldenEye (1995)', 0.7953129853820307, 10), (u'Dracula: Dead and Loving It (1995)', 0.562462873963684,
 12), (u'Baby Doll (1956)', 0.6752824480192893, 8194), (u'Nixon (1995)', 0.6739890339260889, 14), (u'Casino (1995)', 0.6104661596864581, 16), (u'Four Rooms (1995)', 0.5
422628635240452, 18), (u'Money Train (1995)', 0.6018116954934223, 20), (u'Copycat (1995)', 0.6111532190981547, 22), (u'Powder (1995)', 0.8201784017169963, 24), (u'Beyon
d the Valley of the Dolls (1970)', 0.6605205545886489, 8196)]
>>>
```

Gives Sorted Pam values for 15 movies



```
>>> model.productFeatures().lookup(1020)[0]
array('d', [-0.12492148578166962, -0.05340638384222984, -0.31663447618484497, 0.04421274736523628, 0.10388962924480438, -0.01314230915158987, -0.0582866370677948, -0.20
443770289421082, 0.2818536162376404, 0.15223726630210876, 0.26096928119659424, -0.2972163259983063, 0.031936608254909515, 0.2227160483598709, 0.5678849816322327, 0.1116
5739595890045, -0.3666112124919891 4, -0.33171382546424866, -0.06347949057817459, 0.19268521666526794, 0.22435785830020905, 0.14025996625423431, -0.5253569483757019, -0.
3036920130252838, 0.2576591372489929, 0.08354176580905914, 0.46839362382888794, 0.555974543094635, -0.2317674309015274, 0.09784027189016342, 0.1935979127883911, -0.2030
9549570083618, -0.08510718494653702, -0.24732032418251038, -0.0863596647977829, -0.5740698575973511, 0.007492734584957361, 0.03315620869398117, 0.37781664729118347, -0.
4025824964046478, -0.13578735291957855, 0.8334403038024902, -0.5862827301025391, -0.20912933349609375, 0.5665944218635559, 0.04547882825136185, 0.30629709362983704, 0.4
114469587802887, -0.06499087810516357, -0.3911411762237549])
>>> import numpy as np
>>> from numpy import linalg as LA
>>> def cosineSimilarity(vec1, vec2):
...   return vec1.dot(vec2) / (LA.norm(vec1) * LA.norm(vec2))
...
>>> itemId = 1020
>>> itemFactor = np.asarray(model.productFeatures().lookup(itemId))[0]
>>> cosineSimilarity(itemFactor,itemFactor)
1.0
>>> Pam = model.productFeatures().map(lambda products:(products[0],cosineSimilarity(np.asarray(products[1]), itemFactor))).join(small_movies_titles).map(lambda r: (r[1]
[1], r[1][0], r[0]))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'small_movies_titles' is not defined
>>> Pam = model.productFeatures().map(lambda products:(products[0],cosineSimilarity(np.asarray(products[1]), itemFactor))).join(movie_TITLE).map(lambda r: (r[1][1], r[1
][0], r[0]))
>>> pam.take(15)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pam' is not defined
>>> Pam.take(15)
[(u'Jumanji (1995)', 0.7607132314742959, 2), (u'Waiting to Exhale (1995)', 0.6817792735081447, 4), (u'Heat (1995)', 0.6735958629268162, 6), (u'Tom and Huck (1995)', 0.7
38520814806536, 8), (u'"Namesake', 0.6545982322662566, 51884), (u'GoldenEye (1995)', 0.7953129853820307, 10), (u'Dracula: Dead and Loving It (1995)', 0.562462873963684,
 12), (u'Baby Doll (1956)', 0.6752824480192893, 8194), (u'Nixon (1995)', 0.6739890339260889, 14), (u'Casino (1995)', 0.6104661596864581, 16), (u'Four Rooms (1995)', 0.5
422628635240452, 18), (u'Money Train (1995)', 0.6018116954934223, 20), (u'Copycat (1995)', 0.6111532190981547, 22), (u'Powder (1995)', 0.8201784017169963, 24), (u'Beyon
d the Valley of the Dolls (1970)', 0.6605205545886489, 8196)]
>>> sortPam = Pam.takeOrdered(20, key=lambda x: -x[1])
>>> sortPam
[(u'Cool Runnings (1993)', 1.0, 1020), (u'Focus (2015)', 0.879541272583467, 129354), (u'My Giant (1998)', 0.8789605502914997, 1839), (u'Babes in Toyland (1934)', 0.8782
454969415883, 3086), (u'How the Grinch Stole Christmas! (1966)', 0.8655294211148408, 52435), (u'Stick It (2006)', 0.8648388233000317, 45221), (u'Monsters University (20
13)', 0.86198790134554, 103141), (u'Harry Potter and the Deathly Hallows: Part 2 (2011)', 0.8614863313562592, 88125), (u'Bean (1997)', 0.8584404219399752, 1665), (u'Anc
horman 2: The Legend Continues (2013)', 0.8569710211958701, 107348), (u'"King\'s Speech', 0.8562069790470774, 81845), (u'"Relic', 0.8538866109935934, 879), (u'Conan the
 Barbarian (1982)', 0.8531679594488126, 1587), (u'Flushed Away (2006)', 0.8531602212550812, 48982), (u'Beerfest (2006)', 0.8531499251834633, 47640), (u'One Fine Day (19
96)', 0.8500484948871316, 605), (u'Junebug (2005)', 0.8498536776921267, 34528), (u'Monster House (2006)', 0.8494171714370431, 46948), (u'"Monsters', 0.8493922573541511,
 4886), (u'Penelope (2006)', 0.8488417604868531, 58347)]
>>>
```