

Learning Apache Spark with Python

CONTENTS

1	Preface	3
1.1	About	3
1.2	Motivation for this tutorial	4
1.3	Copyright notice and license info	4
1.4	Acknowledgement	5
1.5	Feedback and suggestions	5
2	Why Spark with Python ?	7
2.1	Why Spark?	7
2.2	Why Spark with Python (PySpark)?	8
3	Configure Running Platform	11
3.1	Run on Databricks Community Cloud	11
3.2	Configure Spark on Mac and Ubuntu	16
3.3	Configure Spark on Windows	19
3.4	PySpark With Text Editor or IDE	19
3.5	PySparkling Water: Spark + H2O	26
3.6	Set up Spark on Cloud	27
3.7	PySpark on Colaboratory	28
3.8	Demo Code in this Section	28
4	An Introduction to Apache Spark	31
4.1	Core Concepts	31
4.2	Spark Components	32
4.3	Architecture	34
4.4	How Spark Works?	34
5	Programming with RDDs	35
5.1	Create RDD	35
5.2	Spark Operations	39
5.3	<code>rdd.DataFrame</code> vs <code>pd.DataFrame</code>	41
6	Statistics and Linear Algebra Preliminaries	59
6.1	Notations	59
6.2	Linear Algebra Preliminaries	59
6.3	Measurement Formula	61

6.4	Confusion Matrix	62
6.5	Statistical Tests	63
7	Data Exploration	65
7.1	Univariate Analysis	65
7.2	Multivariate Analysis	79
8	Data Manipulation: Features	87
8.1	Feature Extraction	87
8.2	Feature Transform	97
8.3	Feature Selection	116
8.4	Unbalanced data: Undersampling	117
9	Regression	119
9.1	Linear Regression	119
9.2	Generalized linear regression	133
9.3	Decision tree Regression	142
9.4	Random Forest Regression	150
9.5	Gradient-boosted tree regression	158
10	Regularization	167
10.1	Ordinary least squares regression	167
10.2	Ridge regression	168
10.3	Least Absolute Shrinkage and Selection Operator (LASSO)	168
10.4	Elastic net	168
11	Classification	169
11.1	Binomial logistic regression	169
11.2	Multinomial logistic regression	181
11.3	Decision tree Classification	194
11.4	Random forest Classification	206
11.5	Gradient-boosted tree Classification	216
11.6	XGBoost: Gradient-boosted tree Classification	217
11.7	Naive Bayes Classification	219
12	Clustering	233
12.1	K-Means Model	233
13	RFM Analysis	247
13.1	RFM Analysis Methodology	248
13.2	Demo	250
13.3	Extension	256
14	Text Mining	263
14.1	Text Collection	263
14.2	Text Preprocessing	271
14.3	Text Classification	274
14.4	Sentiment analysis	280
14.5	N-grams and Correlations	287

14.6	Topic Model: Latent Dirichlet Allocation	287
15	Social Network Analysis	305
15.1	Introduction	306
15.2	Co-occurrence Network	306
15.3	Appendix: matrix multiplication in PySpark	311
15.4	Correlation Network	312
16	ALS: Stock Portfolio Recommendations	313
16.1	Recommender systems	314
16.2	Alternating Least Squares	315
16.3	Demo	315
17	Monte Carlo Simulation	323
17.1	Simulating Casino Win	324
17.2	Simulating a Random Walk	324
18	Markov Chain Monte Carlo	335
18.1	Metropolis algorithm	336
18.2	A Toy Example of Metropolis	336
18.3	Demos	337
19	Neural Network	345
19.1	Feedforward Neural Network	345
20	Automation for Cloudera Distribution Hadoop	349
20.1	Automation Pipeline	349
20.2	Data Clean and Manipulation Automation	349
20.3	ML Pipeline Automation	352
20.4	Save and Load PipelineModel	353
20.5	Ingest Results Back into Hadoop	353
21	Wrap PySpark Package	355
21.1	Package Wrapper	355
21.2	Pacakge Publishing on PyPI	357
22	PySpark Data Audit Library	359
22.1	Install with <code>pip</code>	359
22.2	Install from Repo	359
22.3	Uninstall	359
22.4	Test	360
22.5	Auditing on Big Dataset	361
23	Zeppelin to jupyter notebook	371
23.1	How to Install	371
23.2	Converting Demos	372
24	My Cheat Sheet	377

25 JDBC Connection	381
25.1 JDBC Driver	381
25.2 JDBC read	381
25.3 JDBC write	383
25.4 JDBC temp_view	383
26 Databricks Tips	385
26.1 Display samples	385
26.2 Auto files download	385
26.3 delta format	387
26.4 mlflow	387
27 PySpark API	391
27.1 Stat API	391
27.2 Regression API	397
27.3 Classification API	416
27.4 Clustering API	436
27.5 Recommendation API	451
27.6 Pipeline API	456
27.7 Tuning API	458
27.8 Evaluation API	463
28 Main Reference	469
Bibliography	471
Python Module Index	473
Index	475

WHY SPARK WITH PYTHON ?

Chinese proverb

Sharpening the knife longer can make it easier to hack the firewood – old Chinese proverb

I want to answer this question from the following two parts:

2.1 Why Spark?

I think the following four main reasons from [Apache Spark™](#) official website are good enough to convince you to use Spark.

1. Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Apache Spark has an advanced DAG execution engine that supports acyclic data flow and in-memory computing.

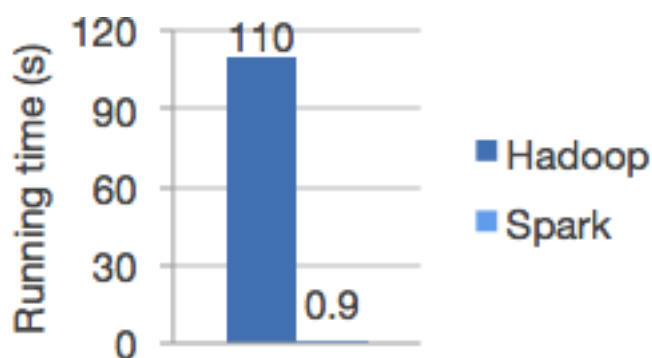


Fig. 1: Logistic regression in Hadoop and Spark

2. Ease of Use

Write applications quickly in Java, Scala, Python, R.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it interactively from the Scala, Python and R shells.

3. Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.

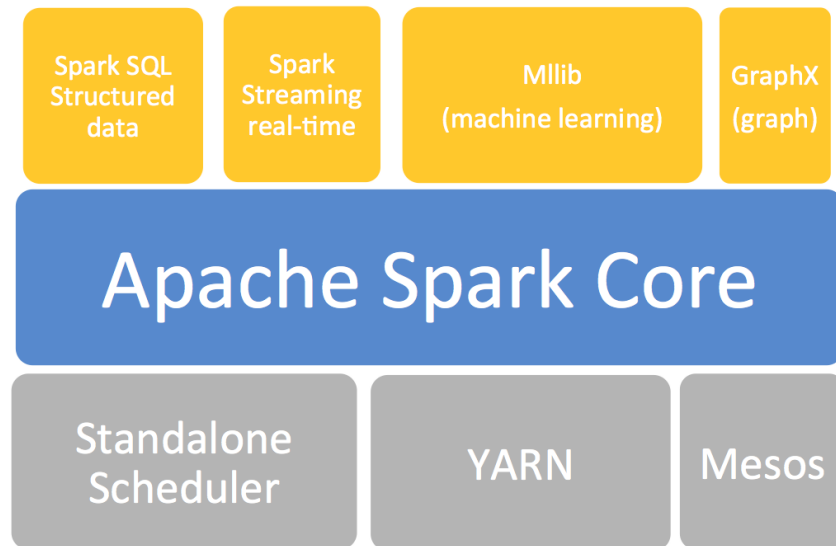


Fig. 2: The Spark stack

4. Runs Everywhere

Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

2.2 Why Spark with Python (PySpark)?

No matter you like it or not, Python has been one of the most popular programming languages.



Fig. 3: The Spark platform

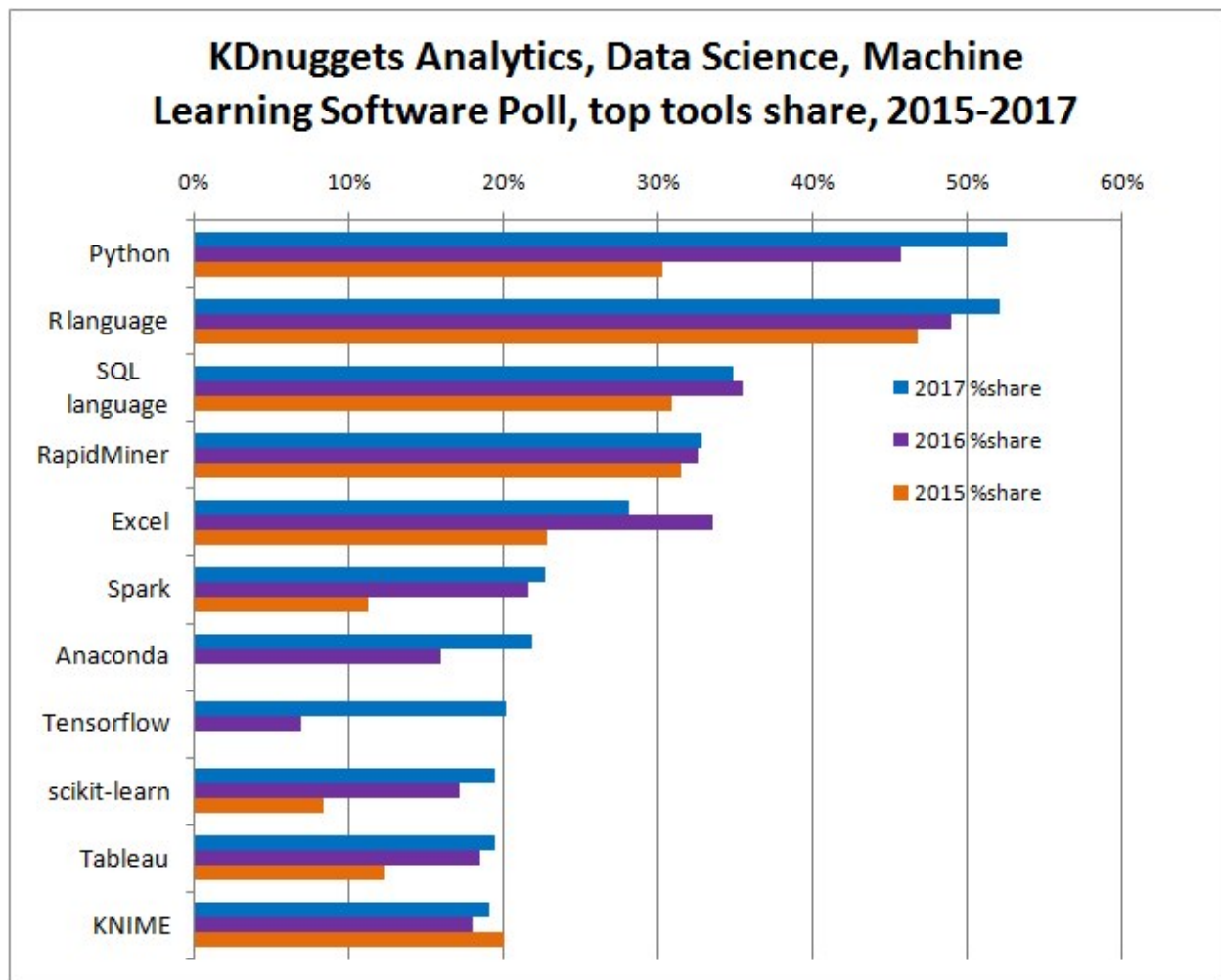


Fig. 4: KDnuggets Analytics/Data Science 2017 Software Poll from [kdnuggets](http://kdnuggets.com).

CONFIGURE RUNNING PLATFORM

Chinese proverb

Good tools are prerequisite to the successful execution of a job. – old Chinese proverb

A good programming platform can save you lots of troubles and time. Herein I will only present how to install my favorite programming platform and only show the easiest way which I know to set it up on Linux system. If you want to install on the other operator system, you can Google it. In this section, you may learn how to set up Pyspark on the corresponding programming platform and package.

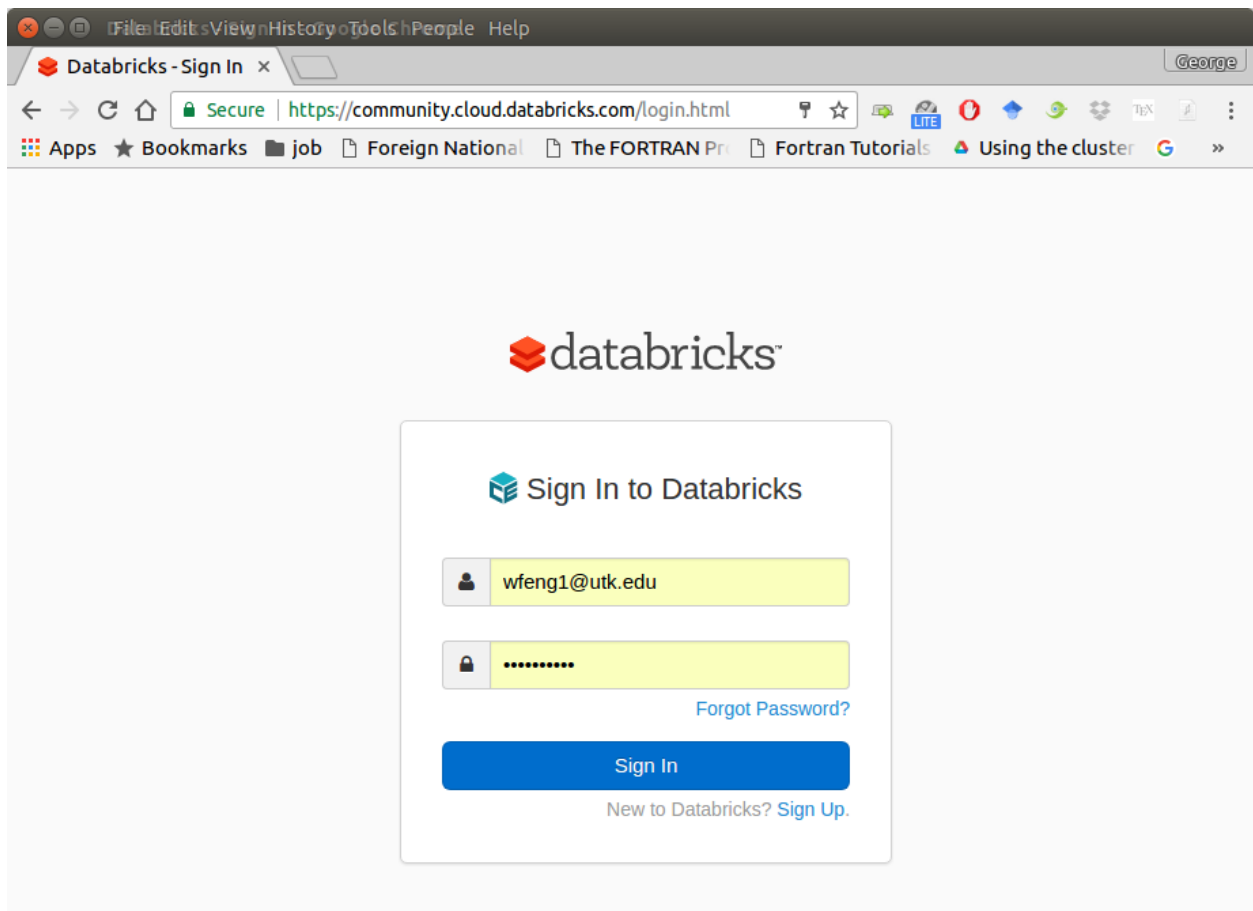
3.1 Run on Databricks Community Cloud

If you don't have any experience with Linux or Unix operator system, I would love to recommend you to use Spark on Databricks Community Cloud. Since you do not need to setup the Spark and it's totally **free** for Community Edition. Please follow the steps listed below.

1. Sign up a account at: <https://community.cloud.databricks.com/login.html>
2. Sign in with your account, then you can creat your cluster(machine), table(dataset) and notebook(code).
3. Create your cluster where your code will run
4. Import your dataset

Note: You need to save the path which appears at Uploaded to DBFS: /File-Store/tables/05rmhuqv1489687378010/. Since we will use this path to load the dataset.

5. Create your notebook



File Edit View History Tools People Help

Databricks

Secure | <https://community.cloud.databricks.com/?o=4622560542654492>

Apps Bookmarks job Foreign National The FORTRAN Pr Fortran Tutorials Using the cluster Attachments

Upgrade ? @George

Community Edition (2.45)

Welcome to databricks™

Featured Notebooks

- [Introduction to Apache Spark on Databricks](#)
- [Databricks for Data Scientists](#)
- [Introduction to Structured Streaming](#)

New

- [Notebook](#)
- [Job](#)
- [Cluster](#)
- [Table](#)
- [Library](#)

Documentation

- [Databricks Guide](#)
- [Python, R, Scala, SQL](#)
- [Importing Data](#)

Open Recent

- [Databricks for Data Scientists](#)
- [linearRegression](#)

What's new?

- Automatic termination of clusters
- Autoscaling local storage with EBS volumes (beta)

[Latest release notes](#)

[Send Feedback](#)

File Edit View History Tools People Help

Create Cluster - Databricks

Secure | <https://community.cloud.databricks.com/?o=4622560542654492#create/cluster>

Apps Bookmarks job Foreign National The FORTRAN Pr Fortran Tutorials Using the cluster Attachments

Upgrade ? @George

Create Cluster

New Cluster

[Cancel](#) [Create Cluster](#)

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name

Databricks Runtime Version

Instance
Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).

AWS **Spark**

Availability Zone

[Send Feedback](#)

