哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Big data technology and Practice

李春山

2020年9月22日

当今计算机领域的热点话题有哪些？

- 人工智能
- 大数据
- 互联网+；
- 中国制造2025
- 工业4.0
- 认知计算
- 物联网
- 云计算
- 移动互联网
- 智慧城市
- 智慧生活
- ……

Contents

Introduction

- **What big data means?**
- Digital computer technology started in 1940's, you may imagine how much data generated and stored around the world, since then.

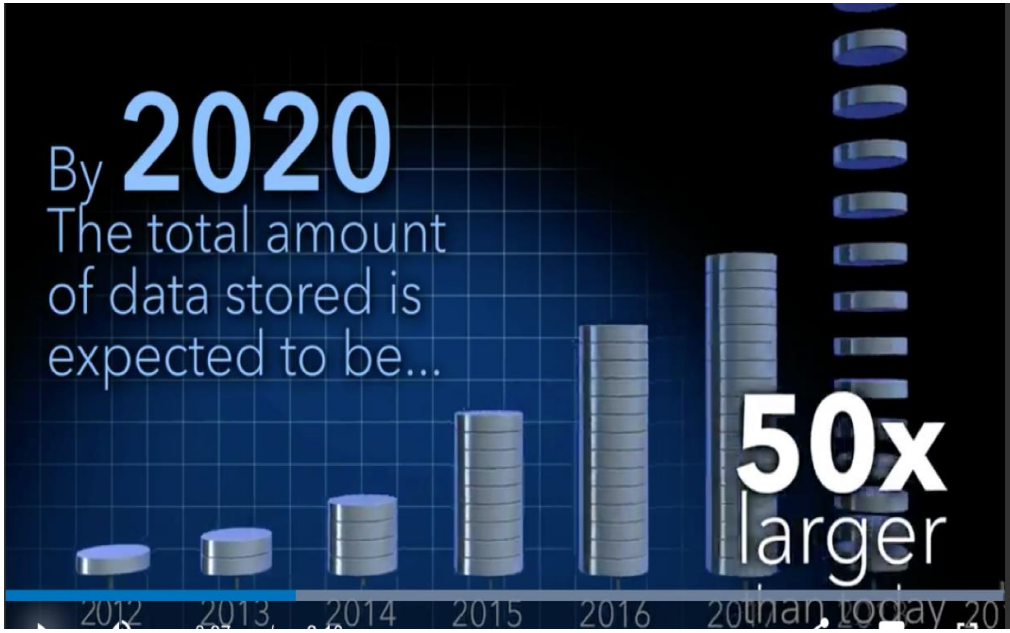
Multiples of bytes V·T·E

Decimal			Binary				
Value	Metric		Value	IEC		JEDEC	
1000	kB	kilobyte	1024	KiB	kibibyte	KB	kilobyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte	MB	megabyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte	GB	gigabyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte	–	
1000^5	PB	petabyte	1024^5	PiB	pebibyte	–	
1000^6	EB	exabyte	1024^6	EiB	exbibyte	–	
1000^7	ZB	**zettabyte**	1024^7	ZiB	zebibyte	–	
1000^8	YB	yottabyte	1024^8	YiB	yobibyte	–	

Orders of magnitude of data

Introduction

- **How much data is there?**
- In 2012, the amount of information stored worldwide exceeded 2.8 Zetabytes



- By 2020, 50 x larger!

Introduction

- An estimated 33% of information could be useful if appropriately tagged and analyzed.

- The amount actually analyzed is about 0.5%.

Introduction

- **What good is all of data?**
  - Data - as unorganized facts is in and of itself worthless.
  - Information - potentially valuable based on data.
  - Knowledge is what we understand based upon information
  - Wisdom is the effective use of knowledge in decision making, say AI.

Introduction

- The importance of big data doesn't revolve around how much data you have, but what you do with it.
- You can take data from any source and analyze it to find answers that enable
  - Cost reductions, Time reductions, New product development and optimized offerings, and Smart decision making

Introduction

- **When you combine big data with high-powered analytics, you can accomplish business-related tasks, such as:**
  - Determining root causes of failures, issues and defects in near-real time.
  - Generating coupons at the point of sale based on the customer's buying habits.
  - Recalculating entire risk portfolios in minutes.
  - Detecting fraudulent behavior before it affects your organization.

Big Data definition and Applications

- **Unlike traditional data, Big Data refers to heterogeneous formats: structured, unstructured and semi-structured data.**

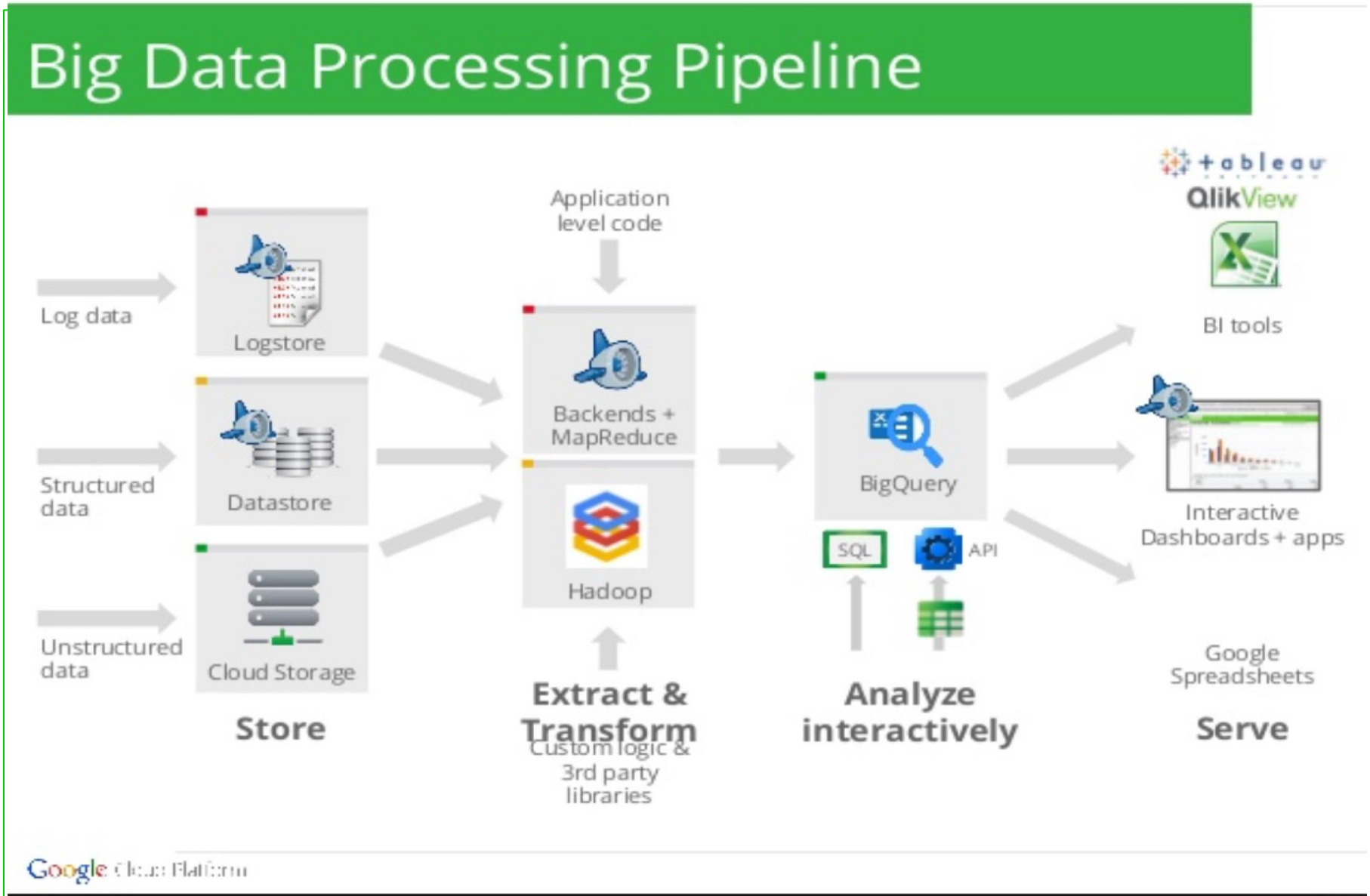
- **Big Data Analysis require powerful technologies and advanced algorithms due to its complex nature. So, the traditional static Business Intelligence tools can no longer be efficient in the case of Big Data applications.**

Big Data definition and Applications

- Most data scientists and experts define Big Data by the following three main characteristics (called the 3Vs):

- So many applications involving in Big data today and future, the course will demonstrate

Big Data Processing and Analytics -pipelines

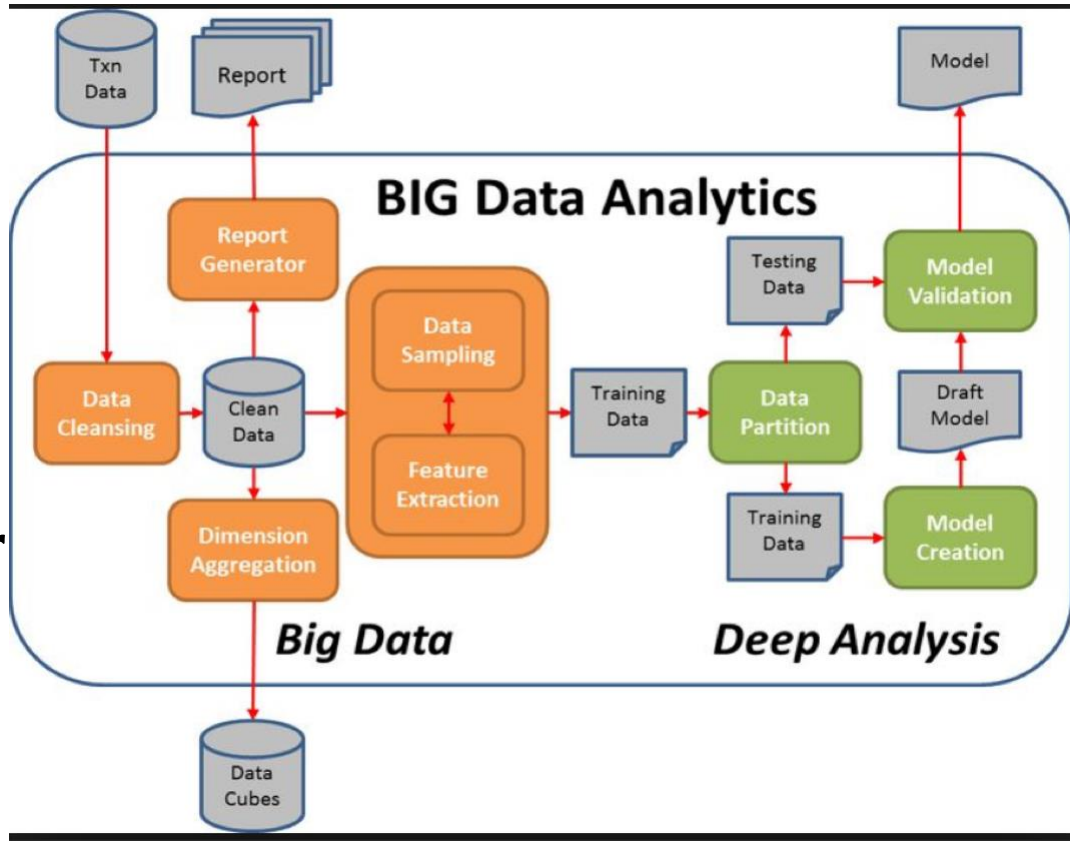


Data science

- **It is an interdisciplinary field about scientific methods, processes, and systems to extract knowledge or insights from data in various forms, similar to data mining.**

Big data analytics

- It is the process of examining large and varied data sets -- i.e., big data – to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful information that can help organizations make more informed business decisions.



Big data platforms, tools and software

- **Big data platform, an enterprise class IT platform, that enables organization in developing, deploying, operating and managing a big data infrastructure and environment, such as Hadoop clusters, Spark (and Flink) processing engine, Storm , Samza, and NoSQL databases,…, which may run on a cloud or cluster.**

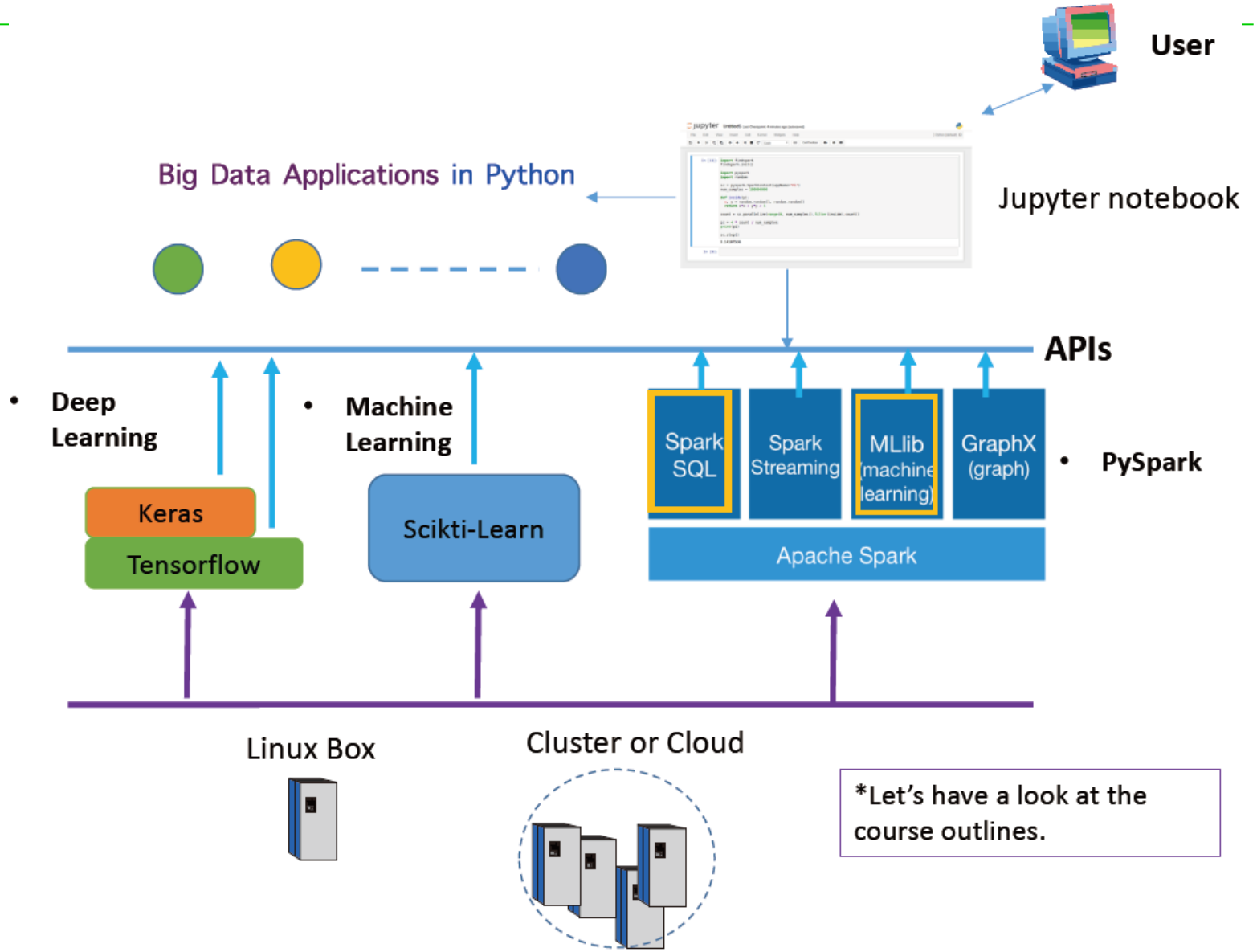
- **In this subject, Apache Spark will be used.**

Big data platforms, tools and software

- There are many tools or software for big data analysis, in this link, you may find “top 50” software from different companies with various purposes.

- Besides of Spark, , we will use the latest and popular tools, Scikit-Learn and Keras, (as well as Tensorflow) for machine learning and deep learning, including Pandas, NumPy, Matplotlib, PyTorch and OpenAI gym

The subject big picture



Contents

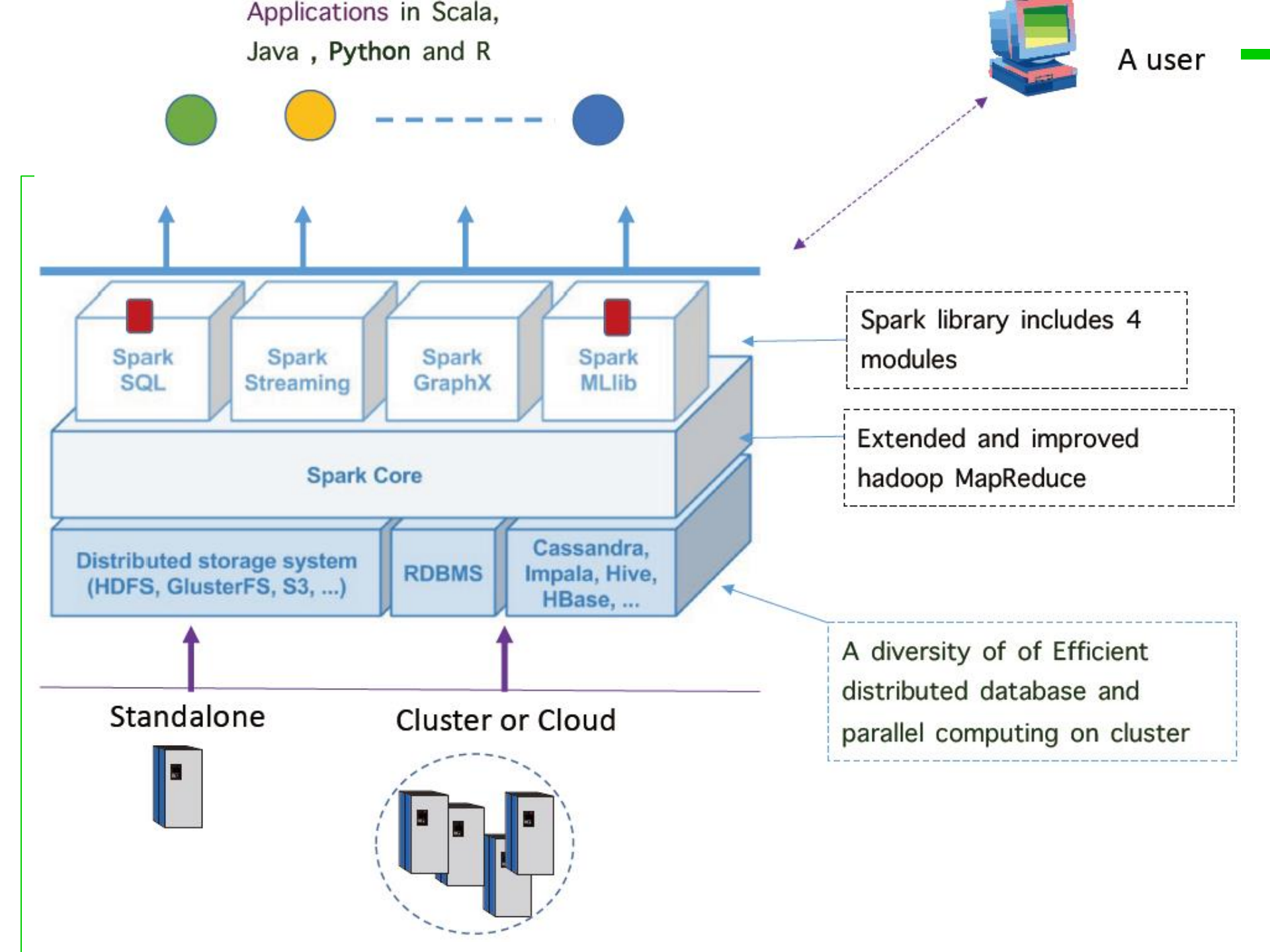
Chapter 2: PySpark

- **Why Spark and Python?**
- **Apache Spark is one of the most popular big data processing platform.**
- **Top companies like Google, Facebook, Netflix, Airbnb, Amazon, NASA, and more are all using Spark to solve their big data problems.**
- **We learn Spark from a user's point of view, particularly on how to use its powerful APIs of PySpark in Python for carrying out big data pre-processing and machine learning.**

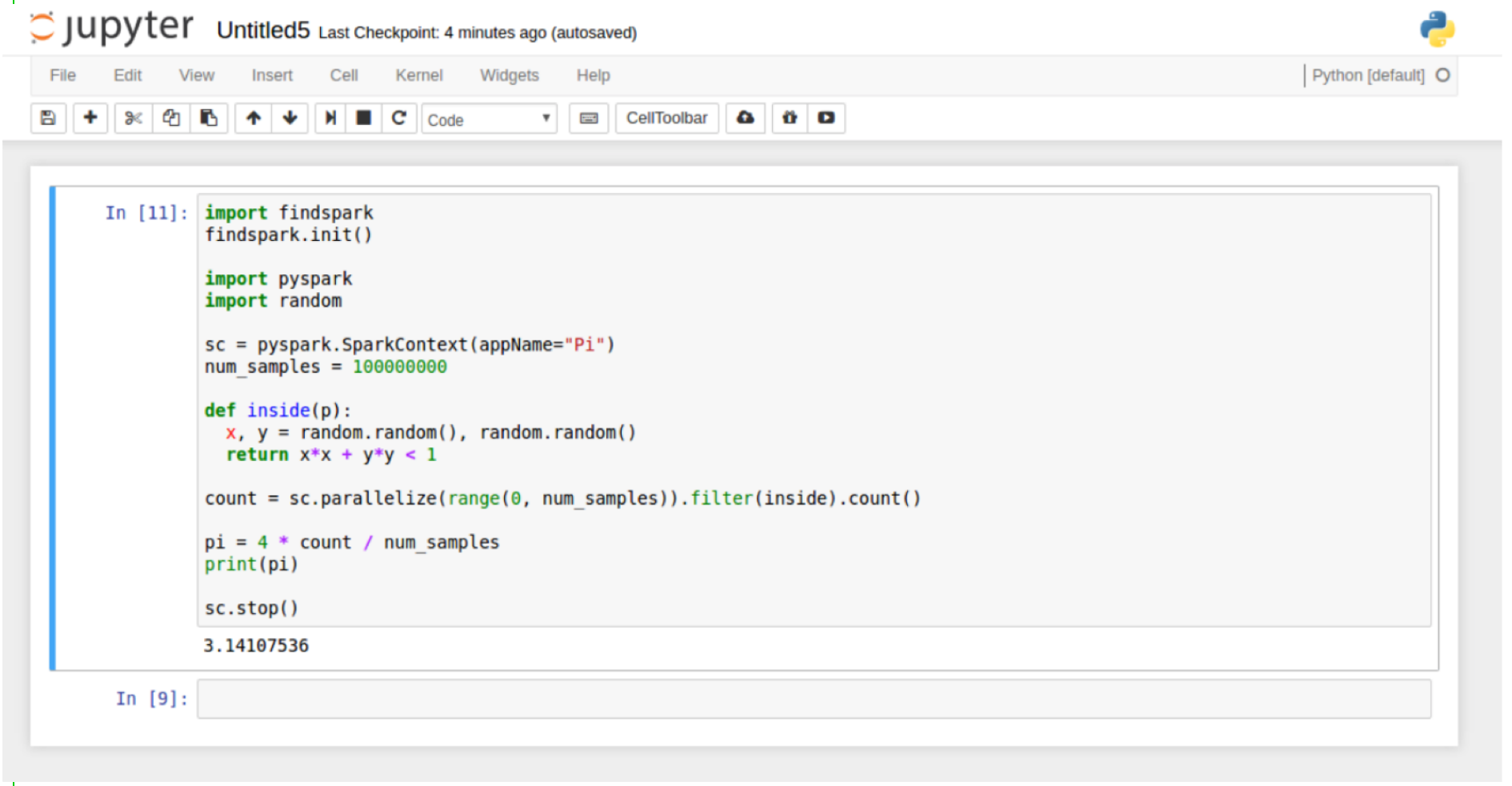
Why Spark and Python?

- Python for Spark is obviously slower than Scala. However, many developers, who love Python because it's flexible, robust, easy to learn, and benefits from all

- Importantly, the most of popular tools for machine learning and deep learning today use Python, of course in this subject


Applications in Scala,
Java , Python and R
A user
Spark SQL
Spark Streaming
Spark GraphX
Spark MLlib
Spark Core
Distributed storage system (HDFS, GlusterFS, S3, ...)
RDBMS
Cassandra, Impala, Hive, HBase, ...
Spark library includes 4 modules
Extended and improved hadoop MapReduce
A diversity of of Efficient distributed database and parallel computing on cluster
Standalone
Cluster or Cloud

Using Jupyter notebook with PySpark



Features of Jupyter notebook

- **As a web application, you can create and share documents.**
- **The documents (or notes) contain live code, equations, visualizations as well as text**
- **These documents are the ideal place to bring together an analysis description and its results as well as they can be executed for the data analysis in real time.**
- **It support multiple languages, such as Python, R and Scala.**
- **Its two main components are the kernels and a dashboard .**

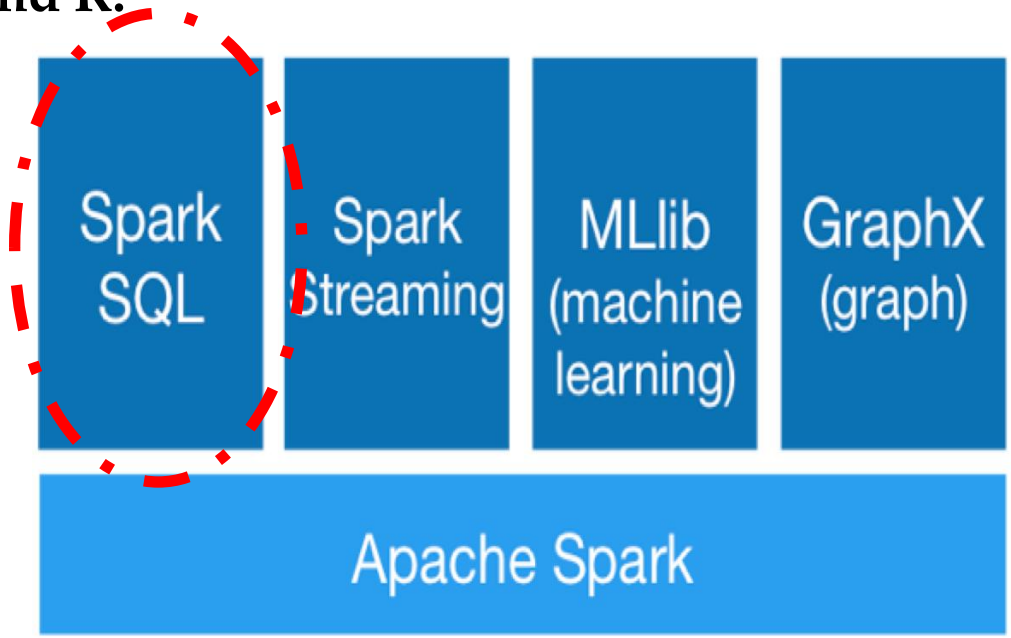
Features of Jupyter notebook

- **A kernel is a program that runs and introspects the user's code.**
- **The dashboard of the application not only shows (or reopen) you documents but also manage the kernels**
- **It is one of the ideal tools for data scientists.**
- **We will have a practical tutorial with Jupyter notebook in lab class.**

Contents

Chapter 3: Spark DataFrame and SQL

- Spark SQL is a Spark's module for working with structured data.
- It lets you query structured data inside Spark programs, using either SQL or DataFrame API.
- Usable in Java, Scala, Python and R.
- Uniform Data Access
- DataFrames and SQL API is provided for connecting to any data source in the same way.
- By the API, you may access a variety of data sources, including Hive, Avro, Parquet, ORC, JSON, and JDBC.
- You can even join data across
- these sources.



Spark DataFrame and SQL

- **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 individua data point (or instance)
- A simple example:

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|   GOOG|    Sam|200.0|
|   GOOG|Charlie|120.0|
|   GOOG|  Frank|340.0|
|   MSFT|   Tina|600.0|
|   MSFT|    Amy|124.0|
|   MSFT|Vanessa|243.0|
|     FB|   Carl|870.0|
|     FB|  Sarah|350.0|
|   APPL|   John|250.0|
|   APPL|  Linda|130.0|
|   APPL|   Mike|750.0|
|   APPL|  Chris|350.0|
+-------+-------+-----+
```

Spark DataFrame and SQL

- **The shift to DataFrames provides many advantages, such as:**
  - A much simpler syntax
  - Ability to use SQL directly in the dataframe
  - Operations are automatically distributed across RDD
- **As Spark DataFrames are able to input and output data from a wide variety of sources, we can use these DataFrames to apply various transformations on the data.**

Spark DataFrame and SQL

- At the end of the transformation calls, you can either show or collect the results to display or for some final processing and analysis.
- The main advantage to using Spark DataFrames vs others: it can handle data across many RDDs, huge data sets that would never fit on a single computer.

Spark DataFrame and SQL

pyspark.sql module

Module Context

Important classes of Spark SQL and DataFrames:

- **`pyspark.sql.SparkSession`** Main entry point for **`DataFrame`** and SQL functionality.
- **`pyspark.sql.DataFrame`** A distributed collection of data grouped into named columns.
- **`pyspark.sql.Column`** A column expression in a **`DataFrame`**.
- **`pyspark.sql.Row`** A row of data in a **`DataFrame`**.
- **`pyspark.sql.GroupedData`** Aggregation methods, returned by **`DataFrame.groupBy()`**.
- **`pyspark.sql.DataFrameNaFunctions`** Methods for handling missing data (null values).
- **`pyspark.sql.DataFrameStatFunctions`** Methods for statistics functionality.
- **`pyspark.sql.functions`** List of built-in functions available for **`DataFrame`**.
- **`pyspark.sql.types`** List of data types available.
- **`pyspark.sql.Window`** For working with window functions.

For example:

class pyspark.sql.**SparkSession**(*sparkContext*, *jsparkSession=None*) [source]

The entry point to programming Spark with the Dataset and DataFrame API.

A SparkSession can be used create **DataFrame**, register **DataFrame** as tables, execute SQL over tables, cache tables, and read parquet files. To create a SparkSession, use the following builder pattern:

```
>>> spark = SparkSession.builder \
...     .master("local") \
...     .appName("Word Count") \
...     .config("spark.some.config.option", "some-value") \
...     .getOrCreate()
```

builder

A class attribute having a **Builder** to construct **SparkSession** instances

builder

A class attribute having a **Builder** to construct **SparkSession** instances

class **Builder** [source]

Builder for **SparkSession**.

appName(*name*) [source]

Sets a name for the application, which will be shown in the Spark web UI.

If no application name is set, a randomly generated name will be used.

Parameters: **name** – an application name

New in version 2.0.

Parameters:
- **key** – a key name string for configuration property
- **value** – a value for configuration property
- **conf** – an instance of **SparkConf**

New in version 2.0.

- **from pyspark.sql import SparkSession**
- **spark = SparkSession.builder.appName("Basics").getOrCreate()**

Creating DataFrame and Using SQL

- We use Jupyter notebook for demonstrating all Python samples for editing, coding and running.
- I have organized them into a hierarchy of the teaching materials, “Bag Data”, as shown:
- Each time when you review them, you must start Jupyter notebook (better at your home directory), then use Web browser to show them

Creating DataFrame and Using SQL

- **How to use Spark DataFrame API, specifically pyspark.sql module, for creating DataFrame from a data source and using SQL queries directly with the dataframe?**

- **Let us look at an ipynb document:**
  - "Creating Spark DataFrame and Using SQL.ipynb", located in
  - …/big_data/Labs/Lab1/Spark_DataFrame

- **Its data source, people.json, is also in this directory.**

DataFrame Basic Operations

- For learning some basic operations with Spark DataFrames, let us open "DataFrame_Basic_Operations.ipynb", located in …/Lab1/Spark_DataFrame directory, where I also put a stock data file, from Apple.

GroupBy and Aggregate

- There are a variety of functions you can import from pyspark.sql.functions. You may have a look at them in lab class.
- One of DataFrame operations frequently in use is GroupBy and Aggregate methods.
- GroupBy allows you to group rows together based on some column value, for example, you could group together sales data by the day the sale occurred, or group repeat customer data based on the name of the customer.

GroupBy and Aggregate

- Once you've performed the GroupBy operation you can use an aggregate function off that data.
- An aggregate function aggregates multiple rows of data into a single output, such as taking the sum of inputs, or counting the number of inputs.

pyspark.sql.functions module

A collections of builtin functions

`pyspark.sql.functions.`**`abs`**(*col*)

Computes the absolute value.

New in version 1.3.

`pyspark.sql.functions.`**`acos`**(*col*)

Computes the cosine inverse of the given value; the returned angle is in the range0.0 through pi.

New in version 1.4.

`pyspark.sql.functions.`**`add_months`**(*start*, *months*) [source]

Returns the date that is *months* months after *start*

```
>>> df = spark.createDataFrame([('2015-04-08',)], ['d'])
>>> df.select(add_months(df.d, 1).alias('d')).collect()
[Row(d=datetime.date(2015, 5, 8))]
```

GroupBy and Aggregate

- **Let us look at an ipynb document: "GroupBy_and_Aggregate.ipynb", in the directory …/Lab1/Spark_DataFrame.**

- Its data source, sales_info.csv, is also in the same directory

Handle Data Missing

- **Data sources are often incomplete, "missing data", there are three basic options for filling in missing data:**
  - 1) Just keep the missing data points.
  - 2) Drop them missing data points (including the entire row)
  - 3) Fill them in with some other value.
- To see how to handle data missing o DataFrame, open "Missing_Data.ipynb",

Dealing with Dates and Timestamps

- **Frequently, big data involves Time and Date information, let's discuss some ways to deal with them!**
- **By using examples, let's see how to deal with dates and timestamps**
- **Open the notebook document: "Dates_and_Timestamps.ipynb" in the folder …/Lab1/Samples_Demo.**

Contents

Chapter 4: Pandas, Numby and Matplotlib

- **Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with data analysis**
- **It is well suited for many different kinds of data processing**
  - Tabular data, as in an SQL table or Excel spreadsheet
  - Ordered and unordered time series data.
  - Arbitrary matrix data with row and column labels
  - Any other form of observational / statistical data sets.
  - The data actually need not be labeled at all to be placed into a pandas data structure

Pandas

- **The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering**

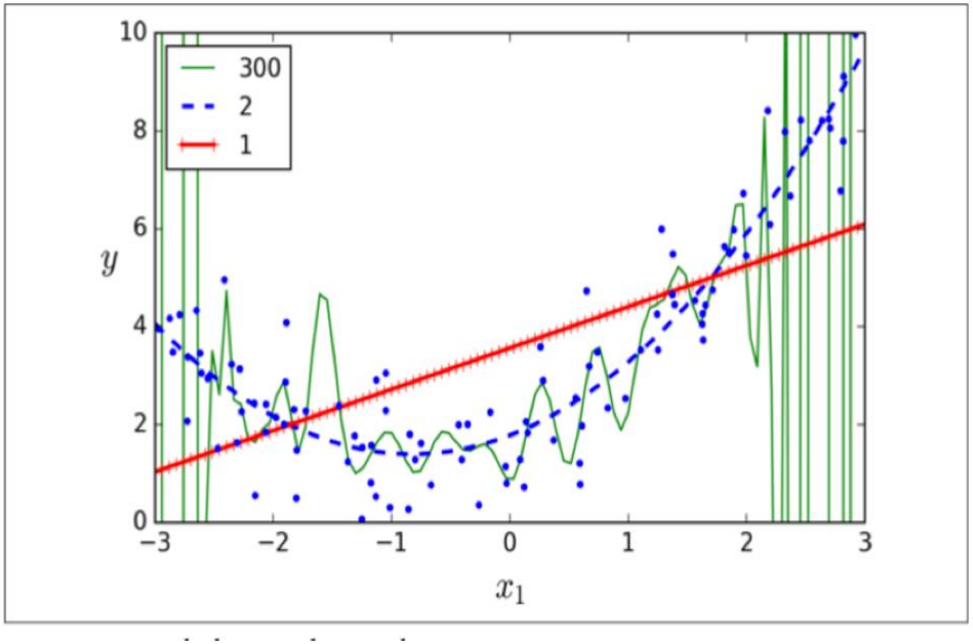
- **We will have a practice with Pandas (and Numby later) with Scikit-Learn.**
  - I will demo some basics about Pandas from this Pandas tutorial: https://pandas.pydata.org/pandas-docs/stable/tutorials.html
  - Let us look at an ipynb document: "Pandas Basics.ipynb" located in the directory …/Lab1/Pandas_and_Numby

NumPy

- **NumPy is the fundamental package for scientific computing in Python.**
- **It provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and a collection of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.**
- **In workshop, we will have a tutorial from** https://www.tutorialspoint.com/numpy/index.htm

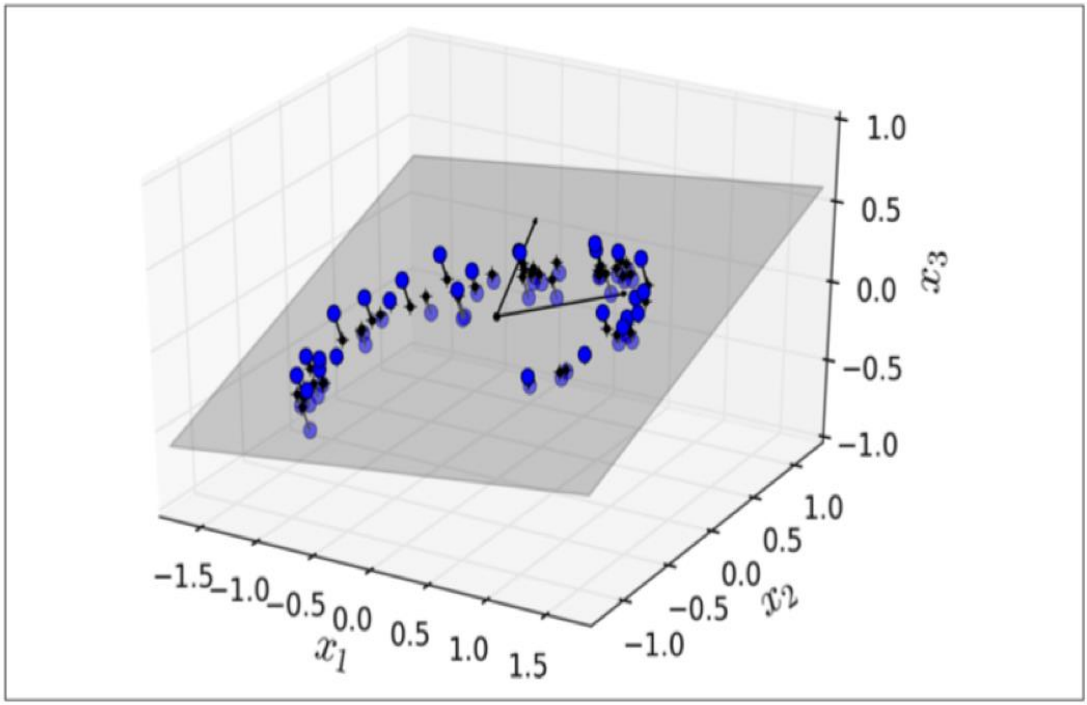
Matplotlib

- **It is a Python 2D plotting library for generating figures in a variety of formats and providing an interactive environments.**



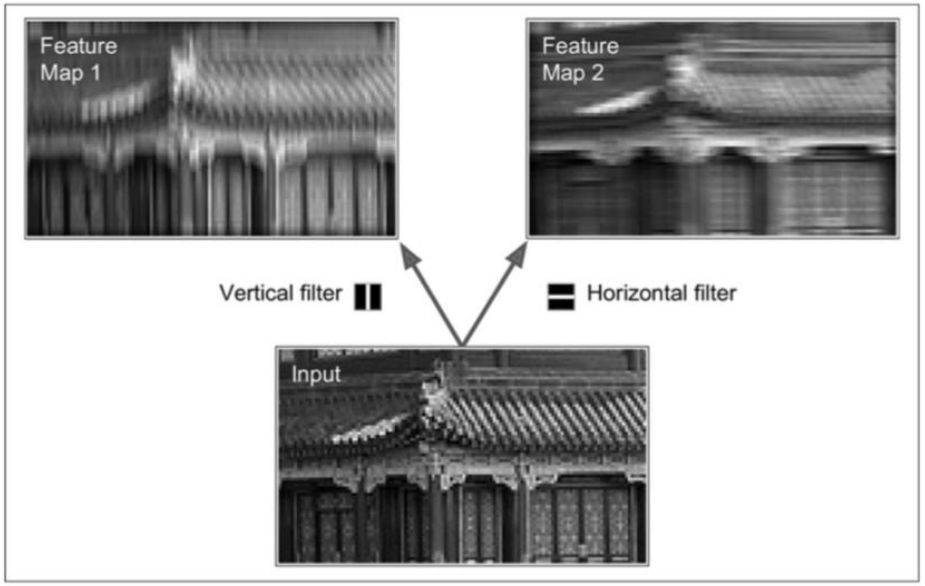
Matplotlib

- It can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



Matplotlib

- You can generate plots, histograms, power spectra, bar charts, error charts scatterplots, etc., with just a few lines of code



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2020年9月22日