Lesson 5

Python and its Libraries with Spark for Data Analysis

Python

- A general purpose, interpreted, interactive, object oriented and high level programming language
- Defines the basic data types, containers, lists, dictionaries, sets, tuples, functions and classes
- Expressive Programming statements

Python Libraries

- Extensive Python Standard Library
- Libraries for regular expressions
- Documentation generation
- Unit testing
- Web browsers
- Threading

Python Libraries

- Databases
- CGI
- Email
- Image manipulation

Python and Spark Binding

- Gives a strong combination of performance and features in the same bundle of codes
- Spark SQL binds with Python easily

Python and Spark Binding

- Spark SQL features together with Python help a programmer to build challenging applications for Big Data

Spark added Python API support for UDFs

- Functions take one row at a time That requires overhead (additional codes) for SerDe
- UDFs defined the UDFs in Java or Scala, and then invoked them from Python

Spark 2.3 Arrow Support to VUDFs and GVUDFs

- Supports to UDFs vectorized UDFs (VUDFs) vectorized UDFs (VUDFs)
- Spark and Apache Arrow facilitates VUDFs, which enables high performance Python UDFs for SerDe and data pipelines
- Provisions statistical functions

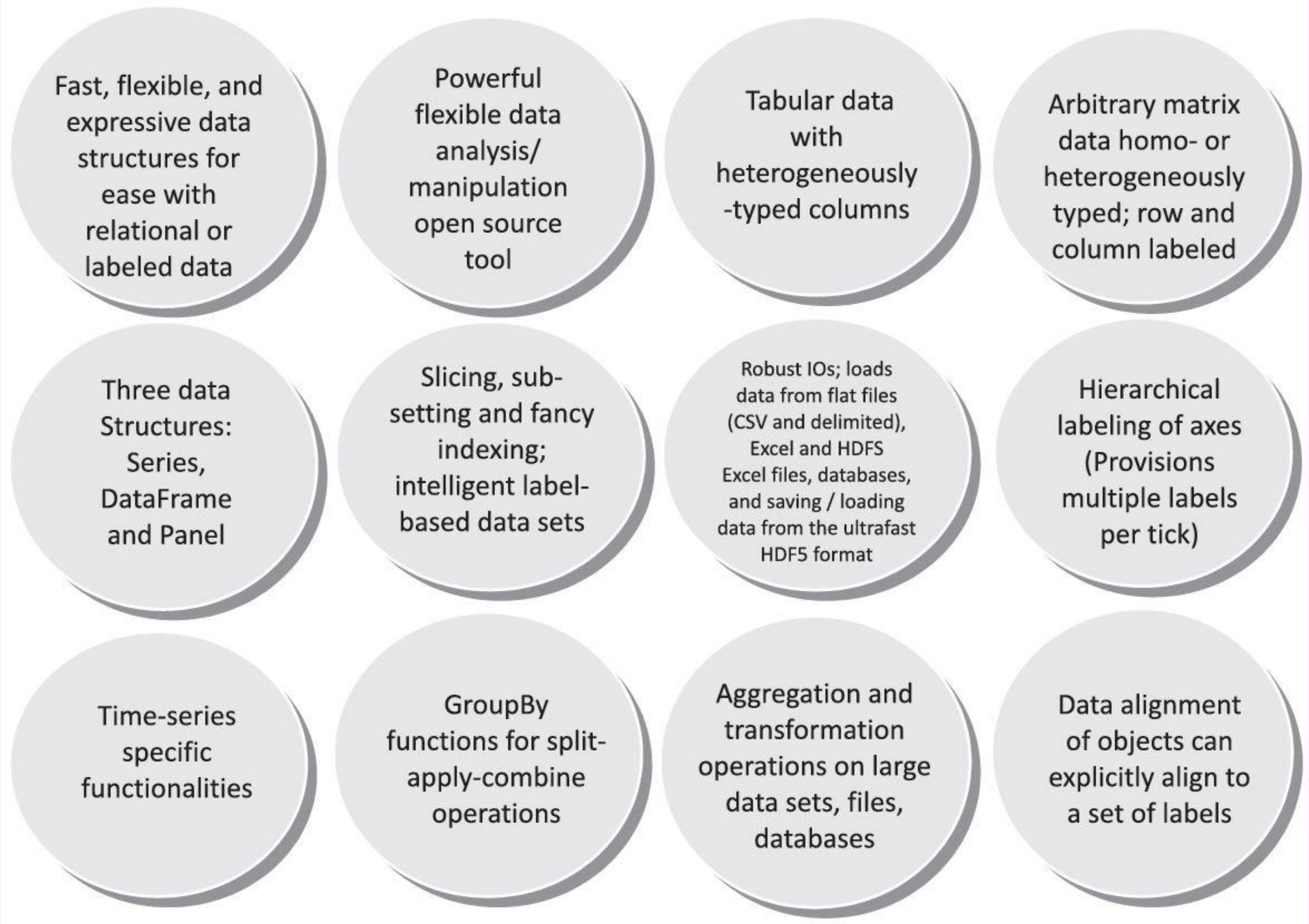
Python for data analysis and Plotting

- NumPy for numerical (Num) analysis
- SciPy scientific (Sci) computations
- Scikit-learn
- Pandas
- StatsModel
- matplotlib functions for plotting the mathematical functions

Python Pandas for Panel data (Grouped Vectors Data) Analytics

- An open source Python package, and consists of BSD-licensed library functions using the Panda (Panel Data)
- Pandas give high performance, easy-to-use data structures and data analysis tools

Figure 5.7 Main features of Panda for data analysis



Fast, flexible, and expressive data structures for ease with relational or labeled data

Powerful flexible data analysis/ manipulation open source tool

Tabular data with heterogeneously -typed columns

Arbitrary matrix data homo- or heterogeneously typed; row and column labeled

Three data Structures: Series, DataFrame and Panel

Slicing, sub-setting and fancy indexing; intelligent label-based data sets

Robust IOs; loads data from flat files (CSV and delimited), Excel and HDFS Excel files, databases, and saving / loading data from the ultrafast HDF5 format

Hierarchical labeling of axes (Provisions multiple labels per tick)

Time-series specific functionalities

GroupBy functions for split-apply-combine operations

Aggregation and transformation operations on large data sets, files, databases

Data alignment of objects can explicitly align to a set of labels

Support to VUDFs and GVUDFs

- Supports to UDFs vectorized UDFs (VUDFs) vectorized UDFs (VUDFs)
- Spark and Apache Arrow facilitates VUDFs, which enables high performance Python UDFs for SerDe and data pipelines

Stats Model and NumPy

- Provisions statistical functions
- NumPy includes (i) N-dimensional array objects and vector mathematics; (ii) linear algebraic functions, Fourier transform and random number functions; sophisticated (broadcasting) functions (iii) integration with C and Fortran codes

NumPy

- Table 5.5 examples of NumPy functions for data analysis problems
- NumPy provides multi-dimensional efficient containers of generic data and definitions of arbitrary data types.

NumPy

- Integrates easily with a wide variety of databases
- NumPy provides import, export (load/save) files,
- Creation of arrays
- Inspection of properties

NumPy

- Copying, sorting and reshaping, addition and removal of elements in the arrays, indexing, sub-setting and slicing of the arrays, scalar and vector mathematics (such as +, −, ×, ÷, power, sqr, sin, log, ceil – round up to nearest int, floor – round down up to the nearest int, round – round to nearest integer)

SciPy

- Adds on top of NumPy
- SciPy defines some useful functions for computing distances between a set of points
- Includes to MATLAB files and special functions, such as routines for numerical integration and optimization

User-Defined Functions (UDFs)

- The SQL registers the UDFs and calls them
- Exposes advanced functionality to SQL users
- User codes call UDFs into the SQL statements without writing the detailed codes

Example of Using UDFs

- Example 5.4 explains creation of a UDF, udfCostPlus() in pandas
- Table column puzzleCost creates using jigsaw_puzzle_info.txt from an RDD
- UDF gives the increased costs in the column, puzzle_cost_USD by 10%.

Vectorized User Defined Functions (VUDFs)

- Spark Arrow facilitates columnar in-memory analytics, which results in high performance of Python UDFs, SerDe and data pipelines
- Example 5.5 explains creation of a vectorized UDF (VUDF)

Creation of a vectorized UDF (VUDF)

- First define a pandas_UDFCostPlus for increasing cost puzzle_cost_USD of toys in puzzle_Costs RDD created from jigsaw_puzzle_info.txt,

VUDF Code Example

- def vectorized_plusTenPercent (v):
- return v4 + 0.1
- df.withColumn(‘v4’, vectorized_ plusTenPercent (df.v))

Grouped Vectorized UDFs (GVUDFs)

- Uses Panda library split-apply-combine pattern in data analysis
- Operates on all the data for a group, such as operate on all the data, “for each car showroom, compute yearly sales

Step 1 for GVUDF

1. Splits a Spark DataFrame into groups based on the conditions specified in the groupBy operator

Step 2 for GVUDF

2. Applies a vectorized user-defined function (pandas.DataFrame -> pandas.DataFrame) to each group

Steps 3 and 4 in GVUDF

3. Combines into new group
4. Returns the results as a new Spark DataFrame

Example

- Example 5.6 explains GVUDF for adding 10% in a cost of group of rows for toy products.

Summary

We learnt :

- Python integration with Spark
- Spark support to Python UDFs
- Spark Arrow for VUDFs and GVUDFs
- Panda analytics tools in Python

End of Lesson 5 on **Python and its Libraries with Spark for Data Analysis**