# PROJECT PACKAGES

## PANDAS

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Pandas are well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet

Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) 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 panda's data structure

The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, Data Frame provides everything that R's data. Frame provides and much more. Pandas are built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas do well:

Easy handling of missing data (represented as Nan) in floating point as well as non-floating point data

Size mutability: columns can be **inserted and deleted** from Data Frame and higher dimensional objects

Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, Data Frame, etc. automatically align the data for you in computations

Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data

Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into Data Frame objects

Intelligent label-based **slicing**, **fancy indexing**, and **sub setting** of large data sets

Intuitive **merging** and **joining** data sets

Flexible **reshaping** and pivoting of data sets

**Hierarchical** labeling of axes (possible to have multiple labels per tick)

Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**

**Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. Pandas are the ideal tool for all of these tasks.

## NUMPY

NumPy is an open source library available in Python that aids in mathematical, scientific, engineering, and data science programming. NumPy is an incredible library to perform mathematical and statistical operations. It works perfectly well for multi-dimensional arrays and matrices multiplication

For any scientific project, NumPy is the tool to know. It has been built to work with the N-dimensional array, linear algebra, random number, Fourier transform, etc. It can be integrated to C/C++ and FORTRAN.

NumPy is a programming language that deals with multi-dimensional arrays and matrices. On top of the arrays and matrices, NumPy supports a large number of mathematical operations

Why use NumPy?

NumPy is memory efficiency, meaning it can handle the vast amount of data more accessible than any other library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, Tensor Flow and Scikit learn to use NumPy array to compute the matrix multiplication in the back end.

## TENSORFLOW

Tensor flow's name is directly derived from its core framework: **Tensor**. In Tensor flow, all the computations involve tensors. A tensor is a **vector** or **matrix** of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) **shape**. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In Tensor Flow, all the operations are conducted inside a **graph**. The graph is a set of computation that takes place successively. Each operation is called an **op node** and is connected to each other.

The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

In Machine Learning, models are feed with a list of objects called **feature vectors**. A feature vector can be of any data type. The feature vector will usually be the primary input to populate a tensor. These values will flow into an op node through the tensor and the result of this operation/computation will create a new tensor which in turn will be used in a new operation. All these operations can be viewed in the graph.

## KERAS

Keas is a high-level neural networks API, written in Python and capable of running on top of Tensor Flow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Supports both convolutional networks and recurrent networks, as well as combinations of the two.

Runs seamlessly on CPU and GPU.

## RE

A regular expression in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets or even documents.

While using the regular expression the first thing is to recognize is that everything is essentially a character, and we are writing patterns to match a specific sequence of characters also referred as

string. ASCII or Latin letters are those that are on your keyboards and Unicode is used to match the foreign text. It includes digits and punctuation and all special characters like $#@! %, etc.

For instance, a regular expression could tell a program to search for specific text from the string and then to print out the result accordingly. Expression can include

Text matching

Repetition

Branching

Pattern-composition etc.

In Python, a regular expression is denoted as RE (REs, regexes or regex pattern) are imported through **re module**. Python supports regular expression through libraries. In Python regular expression supports various things like **Modifiers, Identifiers, and White space characters**.

| **Identifiers** | **Modifiers** | **White space characters** | **Escape required** |
|---|---|---|---|
| \d= any number (a digit) | \d represents a digit.Ex: \d{1,5} it will declare digit between 1,5 like 424,444,545 etc. | \n = new line | . + * ? [] $ ^ () {} \| \ |
| \D= anything but a number (a non-digit) | + = matches 1 or more | \s= space | |
| \s = space (tab,space,newline etc.) | ? = matches 0 or 1 | \t =tab | |
| \S= anything but a space | * = 0 or more | \e = escape | |
| \w = letters ( Match | $ match end of a | \r = carriage return | |

| | | | |
|---|---|---|---|
| alphanumeric character, including "_") | string | | |
| \W =anything but letters ( Matches a non-alphanumeric character excluding "_") | ^ match start of a string | \f= form feed | |
| . = anything but letters (periods) | \| matches either or x/y | ---------- | |
| \b = any character except for new line | [] = range or "variance" | ---------- | |
| \. | {x} = this amount of preceding code | ---------- | |

## SKLEARN

Is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

The functionality that scikit-learn provides include:

**Regression**, including Linear and Logistic Regression

**Classification**, including K-Nearest Neighbors

**Clustering**, including K-Means and K-Means++

**Model selection**

**Preprocessing**, including Min-Max Normalization

# MATPLOTLIB

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also. Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

**Types of Plots:**


**Syntax:** **from** matplotlib **import** pyplot as plt