# Semi-CVVR4 Data Science with Python

## Introduction

TFE 2019‐2020 6/05/2019 15:23 SUJET DU TFE 1. SEMI‐ CVVR4 Data Science with Python Section(s) concernée(s) ELEC, IG Description Context : With the explosion of the volume of data generated by computer devices, the internet of things, huge databases produced by human interaction on websites like google or facebook. Data science have become more and more important. One use case of Data science is in performance and energy monitoring. Modern CPUs have hundreds of sensors that can capture a variety of different events on the system. During my Ph.D. was developed multiple scripts to monitor energy consumption and performance of applications [1], and this generated a big volume of data. To analyze this data and have insights its necessary to do a lot of pre‐processing and use different visualization techniques. For that Python is an excellent choice mainly for its libraries for storing manipulating and visualizing data [2]. Objectives : The idea of this project is to build a python tool with a graphical interface to load and display the data collected from this energy monitor scripts. The tool should be able to do data analysis like extract and display important information, preprocessing, filtering, plotting and comparing data from different applications.

## Data management:

### A.Core libraries

#### 1.Numpy

Short for Numerical Python, [NumPy](https://www.numpy.org/) has been designed specifically for mathematical operations. It primarily supports multi-dimensional arrays and vectors for complex arithmetic operations. In addition to the data structures, the library has a rich set of functions to perform algebraic operations on the supported data types.

Another advantage of the library is its interoperability with other programming languages like C/C++, FORTRAN, and database management systems. Also, as the set of provided functions is precompiled, the computations are performed in an efficient manner.

When starting to deal with the scientific task in Python, one inevitably comes for help to Python’s SciPy Stack, which is a collection of software specifically designed for scientific computing in Python (do not confuse with SciPy library, which is part of this stack, and the community around this stack). This way we want to start with a look at it. However, the stack is pretty vast, there is more than a dozen of libraries in it, and we want to put a focal point on the core packages (particularly the most essential ones).

The most fundamental package, around which the scientific computation stack is built, is NumPy (stands for Numerical Python). It provides an abundance of useful features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which ameliorates performance and accordingly speeds up the execution.

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

**Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.

NumPy (Numerical Python) is the fundamental package for numerical computation in Python; it contains a powerful N-dimensional array object. It has around 18,000 comments on GitHub and an active community of 700 contributors. It’s a general-purpose array-processing package that provides high-performance multidimensional objects called arrays and tools for working with them. [NumPy](https://www.simplilearn.com/mathematical-computing-python-numpy-tutorial) also addresses the slowness problem partly by providing these multidimensional arrays as well as providing functions and operators that operate efficiently on these arrays.

Features:

* Provides fast, precompiled functions for numerical routines
* Array-oriented computing for better efficiency
* Supports an object-oriented approach
* Compact and faster computations with vectorization

Applications:

* Extensively used in data analysis
* Creates a powerful N-dimensional array
* Forms the base of other libraries, such as SciPy and scikit-learn
* Replacement of MATLAB when used with SciPy and matplotlib

From the video, you even learn how to create a simple array and change its shape using the arrange and reshape functions of NumPy.

#### 5. SciPy

Again you need to understand the difference between SciPy Stack and SciPy Library. SciPy contains modules for linear algebra, optimization, integration, and statistics. The main functionality of SciPy library is built upon NumPy, and its arrays thus make substantial use of NumPy. It provides efficient numerical routines as numerical integration, optimization, and many others via its specific submodules. The functions in all submodules of SciPy are well documented — another coin in its pot.

Based on NumPy, the Scientific Python library extends its capabilities by offering advanced operations such as integration, regression and probability to name a few. In order to use [SciPy](https://www.scipy.org/), we must install NumPy first, as it makes use of the underlying modules. What makes SciPy one of the extensively used libraries is the hierarchy in which the sub-modules are organized, and the manuals do an excellent job of explaining the meaning and usability of the exported modules.

SciPy (Scientific Python) is another free and open-source Python library extensively used in data science for high-level computations. SciPy has around 19,000 comments on GitHub and an active community of about 600 contributors. It’s widely used for scientific and technical computations because it extends NumPy and provides many user-friendly and efficient routines for scientific calculations.

Features:

* Collection of algorithms and functions built on the NumPy extension of Python
* High-level commands for data manipulation and visualization
* Multidimensional image processing with the SciPy.ndimage submodule
* Includes built-in functions for solving differential equations

Applications:

* Multidimensional image operations
* Solving differential equations and the Fourier transform
* Optimization algorithms
* Linear algebra

A simple demonstration of the functions of SciPy follows in the video of Python libraries for Data Science.

#### 6. Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

[Python Data Analysis Library](https://pandas.pydata.org/) is an open source library that helps organize data across various parameters, depending upon requirements. The variety of built-in data types like series, frames, and panels make Pandas a favorite library among Data Scientists. The tabular format of frames allow database-like add/delete operations on the data which makes grouping an easy task.

In addition, Panda provides a three dimensional panel data structure which helps in better visualization of the data types. The flexibility of the library supports multiple data formats including missing data.

Pandas (Python data analysis) is a must in the data science life cycle. It is the most popular and widely used Python library for data science, along with NumPy in matplotlib. With around 17,00 comments on GitHub and an active community of 1,200 contributors, it is heavily used for data analysis and cleaning. Pandas provide fast, flexible data structures, such as data frame CDs, which are designed to work with structured data very quickly and intuitively.

Features:

* Eloquent syntax and rich functionalities that gives you the freedom to deal with missing data
* Enables you to create your function and run it across a series of data
* High-level abstraction
* Contains high-level data structures and manipulation tools

Applications:

* General data wrangling and cleaning
* ETL (extract, transform, load) jobs for data transformation and data storage, as it has excellent support for loading CSV files into its data frame format
* Used in a variety of academic and commercial areas, including statistics, finance, and neuroscience
* Time-series-specific functionality, such as date range generation, moving window, linear regression, and date shifting.

Pandas provides features like:

* Dataset joining and merging
* Data Structure column deletion and insertion
* Data filtration
* Reshaping datasets
* DataFrame objects to manipulate data, and much more!

#### 7. PyBrain

#### 17.Statsmodels

The [StatsModels](http://www.statsmodels.org/stable/index.html) module allows users to perform statistical modelling on the data using the modelling and plotting support of the library. The models could be used for the purpose for forecasting across various domains. Model types supported include linear as well as regression models.

StatsModels also support time series analysis capabilities which are particularly poplar in the financial organizations to maintain the stock market information in a convenient format, for instance. Also, the models are fast enough to be used for the big data sets, making it an optimal choice for the same.

#### 27.PyOD

truggling with detecting outliers? You’re not alone. It’s a common problem among aspiring (and even established) data scientists. How do you define outliers in the first place?

Don’t worry, the PyOD library is here to your rescue.

PyOD is a comprehensive and scalable Python toolkit for detecting outlying objects. Outlier detection is basically identifying rare items or observations which are different significantly from the majority of data.

You can download pyOD by using the below code:

pip install pyod

How does PyOD work and how can you implement it on your own? Well, the below guide will answer all your PyOD questions

### B. Plotting

#### 9.Matplotlib

A part of the SciPy core package, [Matplotlib](https://matplotlib.org/) is used for the graphical representation of the processed data as per the user's requirements. We can generate various types of graphs including histograms, pie-charts, or a simple bar chart. It provides an object oriented MATLAB-like interface for users to perform desired operations on the data. An important feature of the library is its ability to offer customization to almost every available feature which makes the usage very flexible to the users.

[Matplotlib](https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial) has powerful yet beautiful visualizations. It’s a plotting library for Python with around 26,000 comments on GitHub and a very vibrant community of about 700 contributors. Because of the graphs and plots that it produces, it’s extensively used for data visualization. It also provides an object-oriented API, which can be used to embed those plots into applications.

Features:

* Usable as a MATLAB replacement, with the advantage of being free and open-source
* Supports dozens of backends and output types, which means you can use it regardless of which operating system you’re using or which output format you wish to use
* Pandas itself can be used as wrappers around MATLAB API to drive MATLAB like a cleaner
* Low memory consumption and better runtime behavior

Applications:

* Correlation analysis of variables
* Visualize 95 percent confidence intervals of the models
* Outlier detection using a scatter plot etc.
* Visualize the distribution of data to gain instant insights

The [Python Libraries for Data Science video](https://youtu.be/8OixQrWRiXo?t=701) demonstrates a straightforward plot to get a basic idea of the possibilities with Matplotlib.

Along with these libraries, data scientists are also leveraging the power of some other useful libraries:

* Similar to TensorFlow, Keras is another popular library that is used extensively for deep learning and neural network modules. Keras supports both the TensorFlow and Theano backends, so it is a good option if you don’t want to dive into the details of TensorFlow.
* Scikit-learn is a machine learning library that provides almost all the machine learning algorithms you might need. Scikit-learn is designed to be interpolated into NumPy and SciPy.
* Seabourn is another library for data visualization. It’s an enhancement of matplotlib, as it introduces additional plot types.

Histogram

|  |
| --- |
|  |
| %matplotlib inline | |
|  | |

|  |
| --- |
| import matplotlib.pyplot as plt |
|  |

|  |
| --- |
| from numpy.random import normal |
|  |

|  |
| --- |
| x = normal(size=100) |
|  |

|  |
| --- |
| plt.hist(x, bins=20) |
|  |

plt.show()

3D Graph

|  |
| --- |
|  |
| from matplotlib import cm | |
|  | |

|  |
| --- |
| from mpl\_toolkits.mplot3d import Axes3D |
|  |

|  |
| --- |
| import matplotlib.pyplot as plt |
|  |

|  |
| --- |
| import numpy as np |
|  |

|  |
| --- |
| fig = plt.figure() |
|  |

|  |
| --- |
| ax = fig.gca(projection='3d') |
|  |

|  |
| --- |
| X = np.arange(-10, 10, 0.1) |
|  |

|  |
| --- |
| Y = np.arange(-10, 10, 0.1) |
|  |

|  |
| --- |
| X, Y = np.meshgrid(X, Y) |
|  |

|  |
| --- |
| R = np.sqrt(X\*\*2 + Y\*\*2) |
|  |

|  |
| --- |
| Z = np.sin(R) |
|  |

|  |
| --- |
| surf = ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwarm) |
|  |

plt.show()

#### 11.Seaborn

Seaborn is essentially a higher-level API based on the matplotlib library. It contains more suitable default settings for processing charts. Also, there is a rich gallery of visualizations including some complex types like time series, jointplots, and violin diagrams.

The seaborn updates mostly cover bug fixes. However, there were improvements in compatibility between FacetGrid or PairGrid and enhanced interactive matplotlib backends, adding parameters and options to visualizations.

Seaborn is another plotting library based on matplotlib. It is a python library that provides high level interface for drawing attractive graphs. What matplotlib can do, Seaborn just does it in a more visually appealing manner.

Some of the features of Seaborn are:

* A dataset-oriented API for examining relationships between multiple variables
* Convenient views onto the overall structure of complex datasets
* Tools for choosing color palettes that reveal patterns in your data

You can install Seaborn using just one line of code:

pip install seaborn

#### 12.Bokeh

The [Bokeh](https://bokeh.pydata.org/en/latest/) library is a standalone library that enables users to plot the data using a web browser interface. Internally it uses JavaScript infrastructure, and therefore is independent of Matplotlib. An essential aspect of Bokeh library is its emphasis on widgets, which allows users to represent the data in various supported formats such as graphs, plots, and labels.

In addition, it also supports interactive visualizations via "callbacks", which allow you to hook in to Bokeh methods using JavaScript.

Bokeh is an interactive visualization library that targets modern web browsers for presentation. It provides elegant construction of versatile graphics for a large number of datasets.

Bokeh can be used to create interactive plots, dashboards and data applications. You’ll be pretty familiar with the installation process by now:

pip install bokeh

#### 13.Plotly

Primarily focused on 3D plotting, [Plotly](https://plot.ly/) can be integrated flawlessly with web applications and provides a number of useful APIs for languages to import. It uses data driven documents at its core for real time data representation, and users can configure it to process the graphics at server-side and send the results to the client or otherwise. We can also share the data with others over the platform, if required. There is also inter-operability between Plotly and Matplotlib data formats.

#### 19.PyQt

##### Advantages of using PyQt

1. Coding flexibility – GUI programming with Qt is designed around the concept of [signals and slots](https://pythonspot.com/pyqt5-signals-and-slots/) for establishing communication amongst objects. That permits flexibility when dealing with GUI events and results in a smoother codebase.
2. More than a framework – Qt uses a wide array of native platform APIs for the purpose of networking, database creation, and many more. It offers primary access to them via a unique API.
3. Various UI components – Qt offers several widgets, such as buttons or menus, all designed with a basic appearance across all supported platforms.
4. Various learning resources – because PyQt is one of the most used UI frameworks for Python, you can get easy access to a wide array of documentation.
5. Easy to master – PyQt comes with a user-friendly, straightforward API functionality, along with specific classes linked to Qt C++. This allows the user to use previous knowledge from either Qt or C++, making PyQt easy to understand.

##### Disadvantages of using PyQt

1. Lack of Python-specific documentation for classes in PyQt5
2. It requires a lot of time for understanding all the details of PyQt, meaning it is a quite steep learning curve

#### 31. Tkinter

##### Advantages of using Tkinter

1. Available out-of-charge for commercial usage.
2. It is featured in the underlying Python library.
3. Creating executables for Tkinter apps is more accessible since Tkinter is included in Python, and, as a consequence, it comes with no other dependencies.
4. Simple to understand and master, as Tkinter is a limited library with a simple API, being the primary choice for creating fast GUIs for Python scripts.

##### Disadvantages of using Tkinter

1. Tkinter does not include advanced widgets.
2. It has no similar tool as Qt Designer for Tkinter.
3. It doesn't have a native look and feel

#### What to choose?

Both [Tkinter](https://en.wikipedia.org/wiki/Tkinter) and [PyQt](https://riverbankcomputing.com/software/pyqt/intro/) are useful for designing acceptable GUI’s, but at the same time, they differ in terms of adaptability and functionality.

Mostly, Tkinter is all about writing GUI yourself, program your settings or functionality in the same script.

On the other hand, in PyQt, you separate GUI in a script, and use your Python knowledge from another script.

Instead of creating your own code for the user interface, *you can simply adopt the* [*Qt Designer*](https://pythonbasics.org/qt-designer-python/) *functions to develop your application*.

Therefore, let’s see what the main differences and advantages of PyQt vs. Tkinter are.

Anyhow, in most situations, the best solution is using PyQt, considering the advantages and disadvantages of both PyQt and Tkinter.

GUI programming with Qt is created around signals and slots for communication amongst objects. Thus, it allows flexibility, while it gets to the programmer access to a wide array of tools.

Tkinter can indeed be useful for those that want to design a fundamental and rapid GUIs for Python scripts, yet for a more advanced programming result, almost all programmers opt for the functionalities that come with PyQt.

They admit it is worth mastering the advanced knowledge of PyQt due to the professional programming results that come along.

Thus, when it comes to PyQt vs. Tkinter, it all depends on how much you want to learn and discover.

### C. Machine Learning

#### 4. PyTorch

What is PyTorch? Well, it’s a Python-based scientific computing package that can be used as:

* A replacement for NumPy to use the power of GPUs
* A deep learning research platform that provides maximum flexibility and speed

PyTorch offers the below features:

* **Hybrid Front-End**
* **Tools and libraries:** An active community of researchers and developers have built a rich ecosystem of tools and libraries for extending PyTorch and supporting development in areas from computer vision to reinforcement learning
* **Cloud support:** PyTorch is well supported on major cloud platforms, providing frictionless development and easy scaling through prebuilt images, large scale training on GPUs, ability to run models in a production scale environment, and more

#### 8.SciKit-Learn

Licensed under BSD, [Scikit-Learn](https://scikit-learn.org/stable/index.html) is an open source Machine Learning toolkit built on top of NumPy and SciPy. It features commonly used ML algorithms for preprocessing, classification, regression as well as clustering. The algorithms include [support vector machines](https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/), ridge regressions, [grid search](https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/), [k-means clustering](https://stackabuse.com/k-means-clustering-with-scikit-learn/), and many more.

Along with the algorithms, the kit also provides sample datasets to experiment with. The well documented APIs are easy to use for beginners, as well as advanced users. Due to its good performance across almost all platforms, it's popular for academic use as well as commercial purposes alike.

Image processing?

#### 20.OpenCV

Image processing

#### 21.Shogun

Implemented in C++, [Shogun](https://github.com/shogun-toolbox/shogun/) is an open source toolbox used for ML, providing a unified interface to multiple languages and platforms including Python. It focuses on scalable kernel methods to solve regression as well as classification problems.

The major focus during development was on bioinformatics, hence Shogun can scale to process over 10 million data samples while maintaining accuracy.

Like Pandas for data manipulation and matplotlib for visualization, scikit-learn is the Python leader for building models. There is just nothing else that compares to it.

In fact, scikit-learn is built on NumPy, SciPy and matplotlib. It is open source and accessible to everyone and reusable in various contexts.

Here’s how you can install it:

pip install scikit-learn

Scikit-learn supports different operations that are performed in machine learning like classification, regression, clustering, model selection, etc. You name it – and scikit-learn has a module for that.

### D. Deep learning

#### 2. Theano

[Theano](http://www.deeplearning.net/software/theano/) is a combination of a library and a compiler targeted towards solving complex mathematical equations in the DL area. It uses a multi-dimensional matrix using NumPy to perform the operations. Keeping performance in mind, Theano is very tightly coupled with NumPy and is precompiled, hence is platform independent and makes use of GPU as well. Along with these features, it also provides a unit testing framework for error detection and mitigation.

#### 3. Keras

[Keras](https://keras.io/) is a neural network library which is capable of execution on top of Google's TensorFlow or Microsoft's [CNTK](https://www.microsoft.com/en-us/cognitive-toolkit/) (Cognitive Toolkit). It is designed to be abstract in nature and acts more as a plugin for other deep learning libraries.

Keras can support standard, convolutional, as well as recurrent neural networks and provides distributed interfaces to the models on GPU clusters. Its easy-to-use interface is ideal for quick prototypes and their deployment on the supported platforms.

#### 10.Tensorflow

Primarily focused on neural networks, [TensorFlow](https://www.tensorflow.org/) is a Deep Learning library developed by Google engineers. The library is very extensible and supports numerous platforms, also including GPU support for better visualization. The classes of algorithms include classification, estimation models and differentiation to name a few.

Its rich API support makes it top choice for training neural networks and speech recognition using natural language processing.

[TensorFlow](https://www.simplilearn.com/introduction-to-tensorflow-tutorial) is a library for high-performance numerical computations with around 35,000 comments and a vibrant community of about 1,500 contributors. It’s used across various scientific fields. TensorFlow is a framework for defining and running computations that involve tensors, which are partially defined computational objects that eventually produce a value.

Features:

* Better computational graph visualizations
* Reduces error by 50 to 60 percent in neural machine learning
* Parallel computing to execute complex models
* Seamless library management backed by Google
* Quicker updates and frequent new releases to provide you with the latest features

TensorFlow is particularly useful for the following applications:

* Speech and image recognition
* Text-based applications
* Time-series analysis
* Video detection

The Python Libraries video takes you through an example of TensorFlow in action, reading handwritten digits by building a simple TensorFlow model.

Developed by Google, TensorFlow is a popular deep learning library that helps you build and train different models. It is an open source end-to-end platform. TensorFlow provides easy model building, robust machine learning production, and powerful experimentation tools and libraries.

TensorFlow provides multiple levels of abstraction for you to choose from according to your need. It is used for building and training models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

### E. Natural language processing

#### 14.NLTK

The [Natural Language Toolkit](http://www.nltk.org/) supports the commonly needed features for English language processing such as classification, tokenization, parsing and semantic analysis. After breaking the words into tokens using syntactical analysis, the kit forms a tree like structure using the language semantics and stores the data in its models. Supported on all major platforms, NLTK is an open source community maintained project. The applications are wide reaching, such as sentiment analysis and anti-spam engines.

#### 15.Gensim

A scalable, robust and platform independent library for NLP, [Gensim](https://radimrehurek.com/gensim/) uses NumPy and SciPy packages underneath. Short for 'Generate Similar', it is designed keeping large amount of data in memory and therefore is performance centric. It differs from other packages in the implementation as it uses data in a cascading manner as opposed to grouping it together.

Due to its efficiency, it's used widely across the domains such as healthcare and financial institutions.

#### 22.SpaCy

Another open source library targeted towards NLP, [SpaCy](https://spacy.io/) encompasses neural network models for various languages viz. English, German, French, Italian, and Dutch, among 30 other languages. Unlike other NLP libraries used primarily for academic purposes, SpaCy is focused on commercial usage.

It also provides extensions for machine learning as well as deep learning APIs. Some popular tech companies, like Airbnb and Quora, use SpaCy as a part of their platforms. What makes it stand out from other libraries is its ability to process documents rather than process data as multiple tokens.

It is a super useful and flexible Natural Language Processing (NLP) library and framework to clean text documents for model creation. SpaCy is fast as compared to other libraries which are used for similar tasks.

To install Spacy in Linux:

pip install -U spacy

python -m spacy download en

### F. Data collection

#### 16.Scrapy

Standing true to its name, [Scrapy](https://scrapy.org/download/) is an open source framework aimed at scraping through the data on the worldwide web. Initially designed to extract the data using exported functions, it has evolved into a framework that is used for designing web crawlers to parse through the web pages and store their data in a structured format. Following Python's object oriented and reusability philosophy, Scrapy is structured around a base class named Spider, and keeps adding layers of functionality, as required, around it.

#### 25. Selenium

#### 26.Beautiful Soup

### G.Data mining

#### 23.Orange

Along with machine learning support, the [Orange](http://orange.biolab.si/) toolkit also features visual analytical platform for interactive data mining. It's an open source package released under General Public License and is designed using C++ with Python wrappers on top.

The Orange package includes a set of widgets for visualization, classification, regression and evaluation of the datasets. The fields where Orange is often used range from DNA research and pharmaceutical domain analysis.

### H.Miscellaneous

#### 24.SymPy

While not directly used for data science and analytics, [SymPy](https://www.sympy.org/en/index.html) is a symbolic computation Python library targeted towards algebraic computations. Many data scientists use the library for intermediate mathematical analysis of their data, later to be consumed by other libraries, such as plotting or Machine Learning.

#### 18.Kivy

### I.Model interpretability

#### 28.Lime

LIME is an algorithm (and library) that can explain the predictions of any classifier or regressor. How does LIME do this? By approximating it locally with an interpretable model. Inspired from the paper [“Why Should I Trust You?”: Explaining the Predictions of Any Classifier”](https://arxiv.org/abs/1602.04938), this model interpreter can be used to generate explanations of any classification algorithm.

Installing LIME is this easy:

pip install lime

#### 29.H2O

I’m sure a lot of you will have heard of H2O.ai. They are market leaders in automated machine learning. But did you know they also have a model interpretability library in Python?

H2O’s driverless AI offers simple data visualization techniques for representing high-degree feature interactions and nonlinear model behavior. It provides Machine Learning Interpretability (MLI) through visualizations that clarify modeling results and the effect of features in a model.

### J.Deployment

#### 30.Flask

Flask is a web framework written in Python that is popularly used for deploying data science models. Flask has two components:

* **Werkzeug**: It is a utility library for the Python programming language
* **Jinja**: It is a template engine for Python

## Data

### Json

[JSON](https://www.json.org/), short for JavaScript Object Notation, is one of the most popular formats used in web development. This is the syntax that the JavaScript language uses to denote objects.

As a Python developer, you may notice that this looks eerily similar to a Python dictionary. There are several different solutions to working with JSON in Python, and more often than not this data is loaded into a dictionary.

This library gets the task of encoding and decoding JSON done in a fairly easy to use way. A lot of the other JSON libraries base their API off of this one and behave similarly.

One of the upsides about using the built in JSON module is that you don't have to install any third party libraries, allowing you to have minimal dependencies.

### simplejson

[simplejson](https://pypi.org/project/simplejson/) is a simple and fast JSON library that functions similarly to the built in module. A cool thing about simplejson is that it is externally maintained and regularly updated.

Many Python developers would suggest using simplejson in place of the stock json library for most cases because it is well maintained.

### UltraJSON

Like simplejson, [ujson](https://pypi.org/project/ujson/) is another community-maintained JSON library. This one, however, is written in C and designed to be [really fast](https://artem.krylysov.com/blog/2015/09/29/benchmark-python-json-libraries/). It lacks some of the more advanced features that the built in JSON library has, but really delivers on its promise, as it seems to be unmatched in terms of speed.

If you're dealing with really large datasets and JSON serialization is becoming an expensive task, then ujson is a great library to use.

The Requests library

These JSON serialization libraries are great, but often in the real world there is more context around why you have to deal with JSON data. One of the most common scenarios that requires decoding JSON would be when making HTTP requests to third party REST APIs.

The [requests](http://docs.python-requests.org/en/master/) library is the most popular Python tool for making HTTP requests, and it has a pretty awesome built in json() method on the response object that is returned when your HTTP request is finished. It's great to have a built in solution so you don't have to import more libraries for a simple task.

Another common scenario is that you are building a route on a web application and want to respond to requests with JSON data. [Flask](http://flask.pocoo.org/), a popular lightweight web framework for Python, has a built in [jsonify](http://flask.pocoo.org/docs/1.0/api/#flask.json.jsonify) function to handle serializing your data for you.

Per the Flask docs, the jsonify function takes data in the form of:

Single argument: Passed straight through to [dumps()](http://flask.pocoo.org/docs/1.0/api/#flask.json.dumps).

Multiple arguments: Converted to an array before being passed to [dumps()](http://flask.pocoo.org/docs/1.0/api/#flask.json.dumps).

Multiple keyword arguments: Converted to a dict before being passed to [dumps()](http://flask.pocoo.org/docs/1.0/api/#flask.json.dumps).

Both args and kwargs: Behavior undefined and will throw an exception.

This function wraps [dumps()](http://flask.pocoo.org/docs/1.0/api/#flask.json.dumps) to add a few enhancements that make life easier. It turns the JSON output into a [Response](http://flask.pocoo.org/docs/1.0/api/#flask.Response) object with the application/json mimetype.

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

###### JSON in Python

Python has a built-in package called json, which can be used to work with JSON data.

###### Parse JSON - Convert from JSON to Python

If you have a JSON string, you can parse it by using the json.loads() method.

You can convert Python objects of the following types, into JSON strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None
* When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

###### Format the Result

The example above prints a JSON string, but it is not very easy to read, with no indentations and line breaks.

The json.dumps() method has parameters to make it easier to read the result

You can also define the separators, default value is (", ", ": "), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values

###### Order the Result

The json.dumps() method has parameters to order the keys in the result

### Xml

## Annexes

### Matplotlib

#### Matplotlib.pyplot.figure

matplotlib.pyplot.figure(*num=None*, *figsize=None*, *dpi=None*, *facecolor=None*, *edgecolor=None*, *frameon=True*, *FigureClass=<class 'matplotlib.figure.Figure'>*, *clear=False*, *\*\*kwargs*)[[source]](https://matplotlib.org/3.1.1/_modules/matplotlib/pyplot.html#figure)

Create a new figure.

|  |  |
| --- | --- |
| **Parameters:** | **num** : integer or string, optional, default: None  If not provided, a new figure will be created, and the figure number will be incremented. The figure objects holds this number in a number attribute. If num is provided, and a figure with this id already exists, make it active, and returns a reference to it. If this figure does not exists, create it and returns it. If num is a string, the window title will be set to this figure's num.  **figsize** : (float, float), optional, default: None  width, height in inches. If not provided, defaults to [rcParams["figure.figsize"] = [6.4, 4.8]](https://matplotlib.org/3.1.1/tutorials/introductory/customizing.html?highlight=figure.figsize#a-sample-matplotlibrc-file) = [6.4, 4.8].  **dpi** : integer, optional, default: None  resolution of the figure. If not provided, defaults to [rcParams["figure.dpi"] = 100.0](https://matplotlib.org/3.1.1/tutorials/introductory/customizing.html?highlight=figure.dpi#a-sample-matplotlibrc-file) = 100.  **facecolor** : color spec  the background color. If not provided, defaults to [rcParams["figure.facecolor"] = 'white'](https://matplotlib.org/3.1.1/tutorials/introductory/customizing.html?highlight=figure.facecolor#a-sample-matplotlibrc-file) = 'w'.  **edgecolor** : color spec  the border color. If not provided, defaults to [rcParams["figure.edgecolor"] = 'white'](https://matplotlib.org/3.1.1/tutorials/introductory/customizing.html?highlight=figure.edgecolor#a-sample-matplotlibrc-file) = 'w'.  **frameon** : bool, optional, default: True  If False, suppress drawing the figure frame.  **FigureClass** : subclass of [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)  Optionally use a custom [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure) instance.  **clear** : bool, optional, default: False  If True and the figure already exists, then it is cleared. |
| **Returns:** | **figure** : [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)  The [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure) instance returned will also be passed to new\_figure\_manager in the backends, which allows to hook custom [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure) classes into the pyplot interface. Additional kwargs will be passed to the [Figure](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure) init function. |

Notes

If you are creating many figures, make sure you explicitly call [pyplot.close()](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.close.html#matplotlib.pyplot.close) on the figures you are not using, because this will enable pyplot to properly clean up the memory.

[rcParams](https://matplotlib.org/3.1.1/api/matplotlib_configuration_api.html#matplotlib.rcParams) defines the default values, which can be modified in the matplotlibrc file.

## Bibliographie

<https://bigdata-madesimple.com/top-20-python-libraries-for-data-science/>,08/11/2019

<http://xarray.pydata.org/en/stable/generated/xarray.Dataset.from_dict.html,19/11/2019>

<http://effbot.org/tkinterbook/canvas.htm,19/11/2019>

<https://docs.python.org/2/library/tkinter.html,19/11/2019>

http://effbot.org/tkinterbook//tkinter-hello-tkinter.htm,19/11/2019

### pyplot

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.figure.html,11/11/2019>

https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html,19/11/2019

### json

<https://docs.python.org/2/library/json.html>,19/11/2019

<http://json.org/example.html,19/11/2019>

https://www.w3schools.com/python/python\_json.asp,19/11/2019

### matplotlib

<https://pythonprogramming.net/how-to-embed-matplotlib-graph-tkinter-gui/,19/11/2019>

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.subplots.html,19/11/2019>

<https://www.programcreek.com/python/example/90071/matplotlib.backends.backend_tkagg.FigureCanvasTkAgg,19/11/2019>

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.figure.Figure.html,19/11/2019>

<https://stackoverflow.com/questions/6697259/interactive-matplotlib-plot-with-two-sliders,19/11/2019>

<https://stackoverflow.com/questions/30774281/update-matplotlib-plot-in-tkinter-gui,19/11/2019>

https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python?utm\_source=adwords\_ppc&utm\_campaignid=898687156&utm\_adgroupid=48947256715&utm\_device=c&utm\_keyword=&utm\_matchtype=b&utm\_network=g&utm\_adpostion=1t1&utm\_creative=332602034352&utm\_targetid=aud-299261629574:dsa-473406587955&utm\_loc\_interest\_ms=&utm\_loc\_physical\_ms=1001367&gclid=EAIaIQobChMIguXo94-85QIVh-R3Ch02xwTkEAAYASAAEgIkK\_D\_BwE,19/11/2019

### pujyter

<https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook,19/11/2019>

### numpy

<https://stackoverflow.com/questions/15579649/python-dict-to-numpy-structured-array,19/11/2019>

### pygithub

<https://sookocheff.com/post/tools/downloading-directories-of-code-from-github-using-the-github-api/,19/11/2019>

<https://github.com/PyGithub/PyGithub/issues/576,19/11/2019>

<https://pygithub.readthedocs.io/en/latest/,16/11/2019>

<https://pygithub.readthedocs.io/en/latest/introduction.html,19/11/2019>

<https://pypi.org/project/PyGithub/,19/11/2019>

https://stackoverflow.com/questions/38594717/how-do-i-push-new-files-to-github,19/11/2019

### pyQT

<https://www.tutorialspoint.com/pyqt/pyqt_hello_world.htm,19/11/2019>

### Panda

<https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/,19/11/2019>

<https://www.analyticsvidhya.com/blog/2015/07/11-steps-perform-data-analysis-pandas-python/,19/11/2019>

https://www.kaggle.com/residentmario/indexing-selecting-assigning,19/11/2019