EnjoyAlgorithms · Follow publication

Learning Tips For Popular Python Libraries In Machine Learning

7 min read · Feb 11, 2024

Ravish Kumar Follow

Listen Share

Python is considered the most used programming language in Machine Learning. But have you ever wondered why? And Why exactly is ML becoming so popular these days? It's all because of the ease in the usability of machine learning, which has become possible with the support of an enormous number of Python libraries. We can now build an entire ML model in just 10 to 15 lines of code, which would have taken hundreds and thousands of lines.

Why do we need libraries in Machine Learning?

Libraries package the reusable code modules and present them via functions that take user input and provide the desired output. Not only this, these libraries provide us with the best-optimized form of the functions by using concepts like multi-threading to make execution time extremely fast. Hence, knowledge of these libraries saves time and stops us from reinventing the wheel. It allows us to focus on more advanced parts of machine learning, like developing newer architecture or coining newer algorithms.

But one question should be coming into our minds: if so many libraries exist, what should we focus on while starting the ML journey?

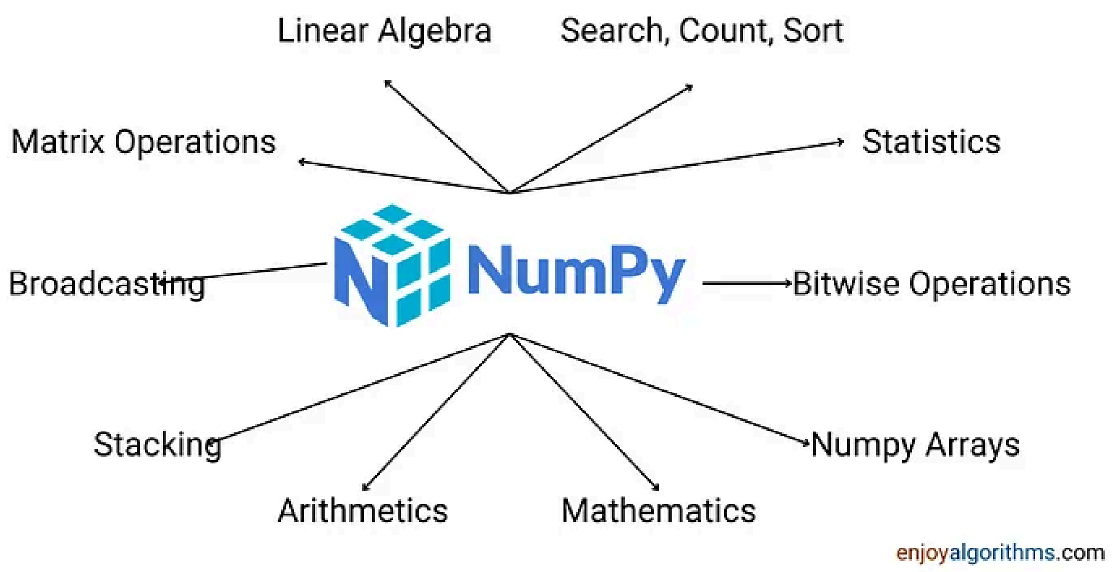
Which libraries shall we focus on while starting the ML journey?

Python community is vast, and that's why we have the support of many Python libraries consisting of all types of functions. But we only need to know some of them to start our Machine Learning journey. Some of the most popular libraries that we should practice before doing hands-on in ML are:

1. **Numpy** (Numerical-Python) to perform numerical calculations efficiently.
2. **Scipy (Scientific Python)** to optimize linear algebra, calculus, and image data-based calculations.
3. **Pandas** to manage the datasets efficiently and effectively.
4. **Matplotlib** to visualize the dataset or perform analysis.
5. **Scikit-learn** to build the end-to-end pipeline for Machine Learning.

Here, we will be discussing each of these libraries briefly and provide their resources to follow and learn.

Numpy (Numerical Python)



Numpy is a very popular open-source library in Machine Learning, and one can find its presence in almost all codes used in ML. The reason for that is that Numpy is used to perform numerical calculations. In ML, we narrow down all forms of datasets to numerical format and then feed them to our Machines to extract meaningful hidden patterns.

It is used to store and perform operations on multi-dimensional arrays or matrices. The capabilities and support it carries for various mathematical functions to perform wider operations ranging from simple linear algebra to estimating infinite Fourier series is way beyond any other libraries. Apart from this, it is swift in the computations compared with similar libraries and acts as a base for other libraries.

Tips to learn Numpy

For beginners who have started their ML journey, the best way to learn Numpy would be to:

- Learn critical operations like slicing, broadcasting, stacking, flattening, reshaping, and resizing numpy arrays.
- Create a dummy multi-dimensional array using Numpy and perform linear algebra operations like Matrix multiplication, addition, or subtraction.

- Try to compare the execution time between the "library-based solutions" and "self-implemented solutions" and observe the advantages of numpy.

Resources to follow and learn Numpy:

- Numpy original documentation
- Summary of Numpy's useful functions

Scipy (Scientific Python)

Scientific Python, popularly known as Scipy, is another open-source library built over the famous Numpy library. It is partially written in Python and partially in C language to boost the speed of mathematical computations further. It provides support for linear algebra, calculus, eigenvalue problems, statistics, and many more. Scipy is an advanced library and hence has some critical applications in Deep Learning or graphical data structures like Sparse matrix support.

One question should be coming into the mind: why use Scipy if Numpy is already there? Scipy is a full-fledged package comprising everything related to numerical data calculations like linear algebra, Fourier series, calculus, trigonometry, etc. If we require these numerical supports, then Scipy is better. Otherwise, for simple array formation, sorting indexing, and simple mathematical calculations, Numpy is more than enough.

Tips to learn Scipy

For beginners who have started their ML journey, the best way to learn Scipy would be to:

- The use of Scipy is mainly seen with Signal processing. Hence, in the case of audio datasets, it works the best. Therefore, load the audio datasets, perform Fourier Analysis, and extract insights.

- Waveforms like Sine, Cosine, and Tangent can easily be formed with the Scipy library. So, to practice ML on dummy algorithms to check whether the model is learning, one can check the pattern following capabilities of that model on the Sine or Cosine datasets.

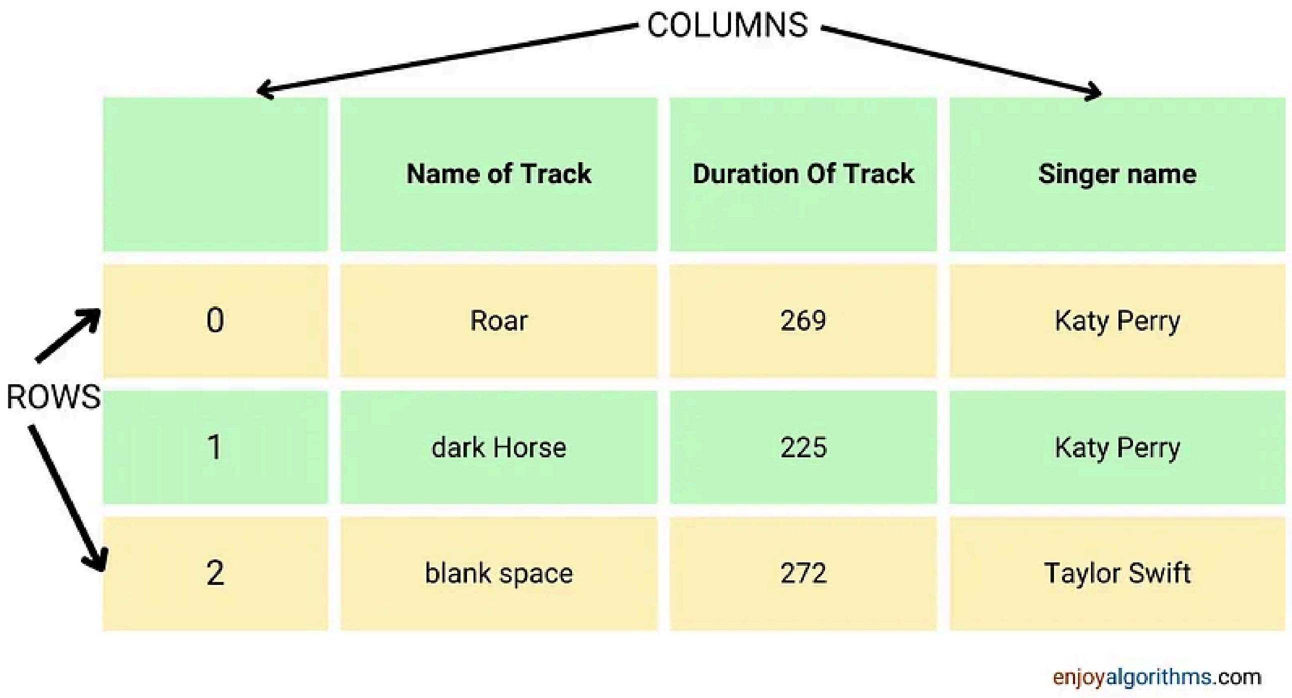
- Perform linear algebra calculations like matrix multiplication, finding determinants, eigenvalues, and eigenvectors using Scipy, and compare the execution performance with respect to the Numpy library.

Resources to follow and learn Scipy:

- Scipy official documentation page and Scipy Community.

- Scipy GitHub codebase

Pandas


COLUMNS
ROWS
enjoyalgorithms.com

	Name of Track	Duration Of Track	Singer name
0	Roar	269	Katy Perry
1	dark Horse	225	Katy Perry
2	blank space	272	Taylor Swift

Pandas is a Python library mainly focusing on Data Analyses. It is also an open-source library built over the Numpy library, which supports large varieties of datasets across all domains. Machine Learning beginners need datasets to practice their learnings to perform analysis or build ML models. Pandas library provides functions to a broader variety of datasets, using which they can easily practice.

It supports two types of data structures (Series for one-dimensional and DataFrame for multi-dimensional) to process data in either structured or semi-structured format.

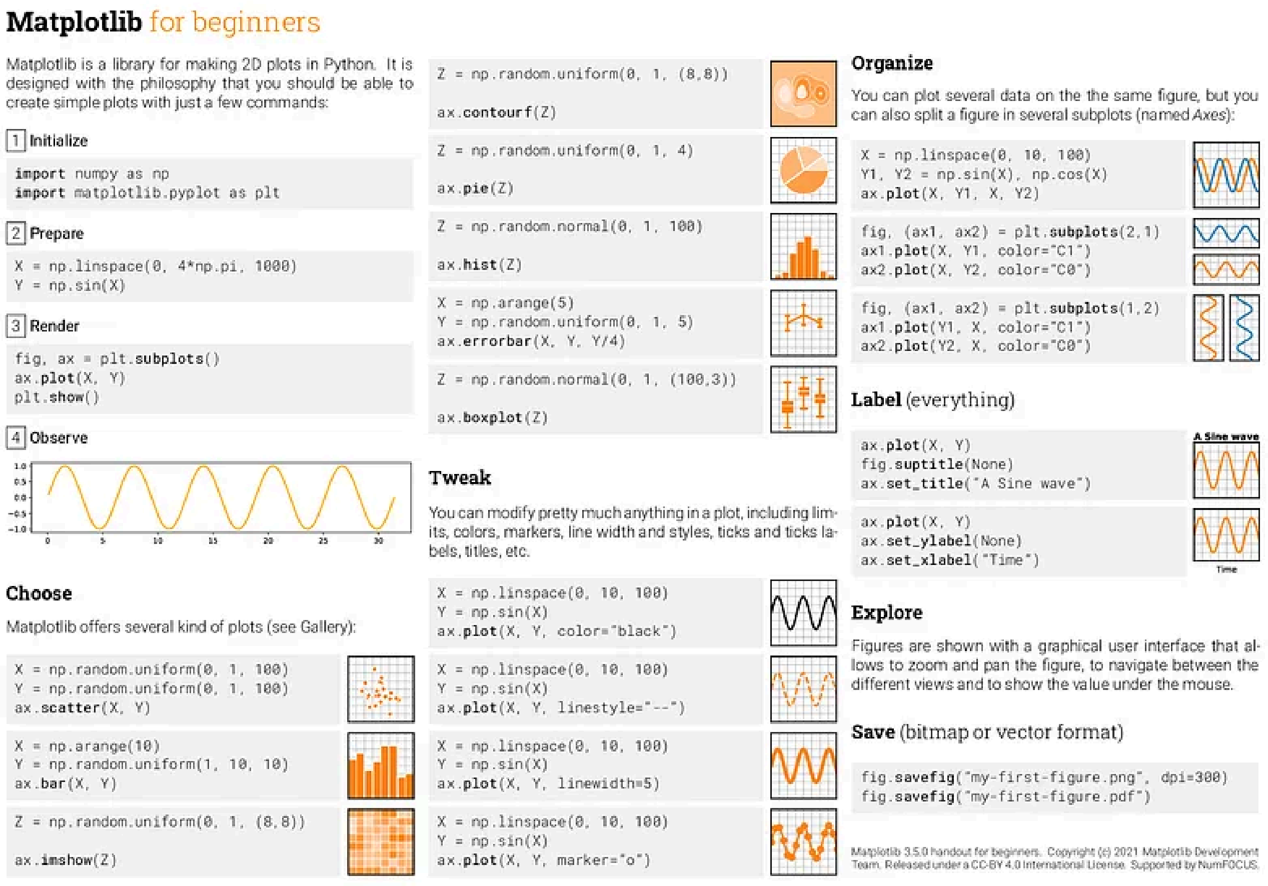
Tips to learn Pandas

- Explore the existing datasets in the Pandas library and play with basic functions like viewing the DataFrame, sorting it, preprocessing it, and then exporting it to files like CSV, JSON, or text files.
- Read data from different file extensions like CSV, excel sheets, JSON, and Text files, apply processing and data analysis techniques, and then save the processed data to that file.
- Use Pandas' built-in plot functions to see what support for data plotting it provides.

Resources to follow and learn Pandas:

- Pandas official documentation
- Book: Python for Data Analysis
- Summary of Pandas' useful functions

Matplotlib



Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

1 Initialize

```
import numpy as np
import matplotlib.pyplot as plt
```

2 Prepare

```
X = np.linspace(0, 4*np.pi, 1000)
Y = np.sin(X)
```

3 Render

```
fig, ax = plt.subplots()
ax.plot(X, Y)
plt.show()
```

4 Observe

Choose

Matplotlib offers several kind of plots (see Gallery):

```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```

```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```

```
Z = np.random.uniform(0, 1, (8,8))

ax.imshow(Z)
```

```
Z = np.random.uniform(0, 1, (8,8))

ax.contourf(Z)
```

```
Z = np.random.uniform(0, 1, 4)

ax.pie(Z)
```

```
Z = np.random.normal(0, 1, 100)

ax.hist(Z)
```

```
X = np.arange(5)
Y = np.random.uniform(0, 1, 5)
ax.errorbar(X, Y, Y/4)
```

```
Z = np.random.normal(0, 1, (100,3))

ax.boxplot(Z)
```

Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```

Organize

You can plot several data on the the same figure, but you can also split a figure in several subplots (named *Axes*):

```
X = np.linspace(0, 10, 100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, X, Y2)
```

```
fig, (ax1, ax2) = plt.subplots(2,1)
ax1.plot(X, Y1, color="C1")
ax2.plot(X, Y2, color="C0")
```

```
fig, (ax1, ax2) = plt.subplots(1,2)
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```

Label (everything)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```

```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```

Explore

Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```

Matplotlib 3.5.0 handout for beginners. Copyright (c) 2021 Matplotlib Development Team. Released under a CC-BY 4.0 International License. Supported by NumFOCUS.

Matplotlib is a popular Python library that plots curves for various data types, training, performance analysis, and visualization. This library helps make informed decisions about the data or processes required for better training. It is one of the most used libraries by data scientists or analysts working in this industry and acts as the base for other visualization libraries like Seaborn, which applies beautification to the matplotlib plots.

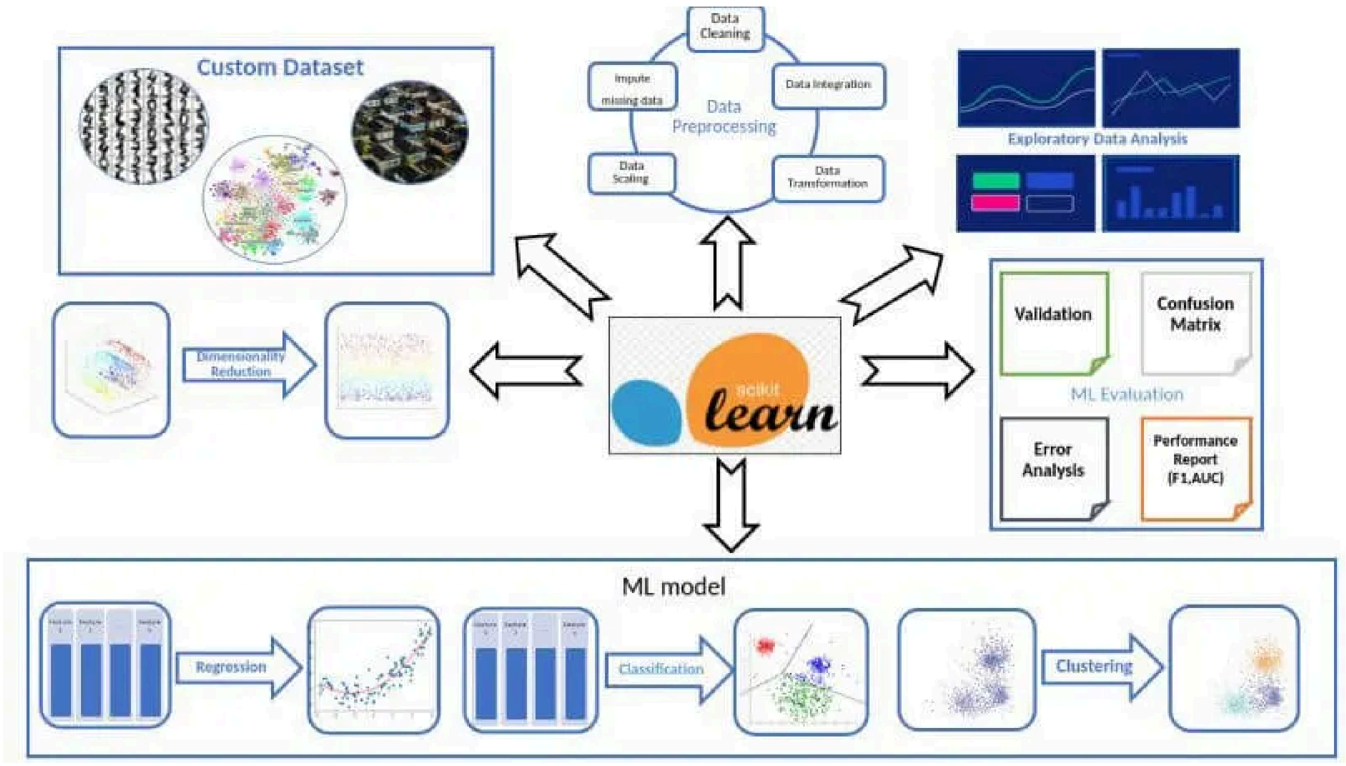
Tips to learn Matplotlib

- Start with simple plots of visualizing the relationship between two simple lists or arrays.
- Try various data analysis techniques on different datasets and plot curves like histograms, pie charts, bar graphs, scatter plots, and box plots. Please also understand the meaning of these plots because that will improve the graphical reading ability.
- Plot 3D scatter plots to visualize the higher dimensional datasets.

Resources to follow and learn Matplotlib:

- Matplotlib official documentation
- Introduction to Matplotlib with popular plots

Scikit-Learn



Scikit-learn is the one-stop library for many Machine Learning beginners and professionals. It provides a lot of functionalities, including dummy data to practice ML, preprocessing modules, a wide range of pre-defined ML algorithms, and support for training evaluation and deployment. Earlier libraries like Pandas and Numpy were not providing additional support mentioned before. In our list of required ML libraries, Scikit-Learn is the only library capable of doing all the things involved in building end-to-end ML pipelines.

It is an open-source library primarily built over Scipy, Numpy, and Matplotlib libraries, which gives it the additional advantage of speedy training and evaluation. It also provides the support to visualize the complete model development pipeline, which includes the stages like how the data will be received, processed, fed to the model, training occurs, evaluation happens, and results are published.

Tips to learn Scikit-Learn

- Scikit-Learn is a huge library, and learning all at once would not be preferable. Hence, using the functions when we need them while building model development pipelines would be better.

- Observe the difference between the execution time of ML algorithms when developed from scratch and Scikit-learn's inbuilt function. It will give a better understanding of how much these libraries help by making the execution fast.
- Use label encoding methods present in Scikit-learn to convert the non-numerical features into numerical values. This saves a lot of time in the case of large datasets.
- Try to load some datasets → preprocess them → use label encoders if required → normalize data features → play with dimensionality reduction techniques → try algorithms to train → visualize the performance of the trained model → Create the complete pipeline and visualize it.

Resources to follow and learn Matplotlib:

- Scikit-learn official documentation
- GitHub Codebase of Scikit-learn
- Important functions in Scikit-learn

Although more libraries help learn Machine Learning and will be used while developing specific models.

Conclusion

The complete knowledge of these five libraries, Numpy, Scipy, Pandas, Matplotlib, and Scikit-learn, will be considered a good starting point for beginners. We have prepared this list based on the requirements to build Machine Learning applications shown in this ML Course. Other libraries like TensorFlow, Keras, PyTorch, and Theano were good candidates for this list, but they are mainly used for developing Deep Learning algorithms. We hope you enjoyed the article.

Enjoy Learning!

16 Week Live Project-Based ML Course: Admissions Open

Machine Learning · Data Science · Python · Libraries · Data Visualization