

AI Introduction

AI Fact and Q&A

- Pattern Recognition, Computer Vision, Robotics, Data Mining were based on Mathematics, such as Linear Algebra, Statistics(probability) and Calculus.
- The theories have been super mature long before computer was invented or before today's powerful PC or super computer. Our imagination, vision or algorithms have been developed long before anything happened today. What we were lacking was the advancement of the material science, power CPU, memory, SSD storage, microchips...etc.
- Higher category of AI, supervised learning and unsupervised learning.
- Supervised learning rely not only by math by it was categorized more by tagging the right answer to train the model and let the model to do the prediction of the answer for the new questions/challenges. So there is a concept of training.
- Training: Tagging answers and helping the model training.
- Model: What is the model? Model is an artifact. Artifact is an item, and possibly a binary file.
- Where is the model? It is saved on your harddrive.
- How is the AI model working? It is loaded into the main memory of the server, such as if you run GPT 4.0 model, then the web request would be routed to the GPT 4.0 server, and it would preload the GPT 4.0 model into the memory. Vise versa this can be routed to GPT 5.0 model for instance. They are definitely on different servers. And the servers are not just single instances, it has a cluster of it for load balancing.
- How large are models? Models can be viewed in different stages. Initially the model is very small, and after the training it can be super large, like a few hundred gigabytes are not uncommon.
- Load Balancing: it is IT architecture term and it is a methodology for distributed computing, to lower the load or traffic to avoid hardware drainage or network constraint. If you would like to learn more about these. You would take more courses on DevOps and it will teach more about servers, software/IT infrastructure architecture. However, you would also need to take operating system and fully understand the Linux and how the operating system is working.

1/ Operating System theory class.

2/ Linux OS study and be an expert on it and understand linux kernel and C programming.

3/ Computer Architecture. Chips, motherboard, IO interfaces...etc.

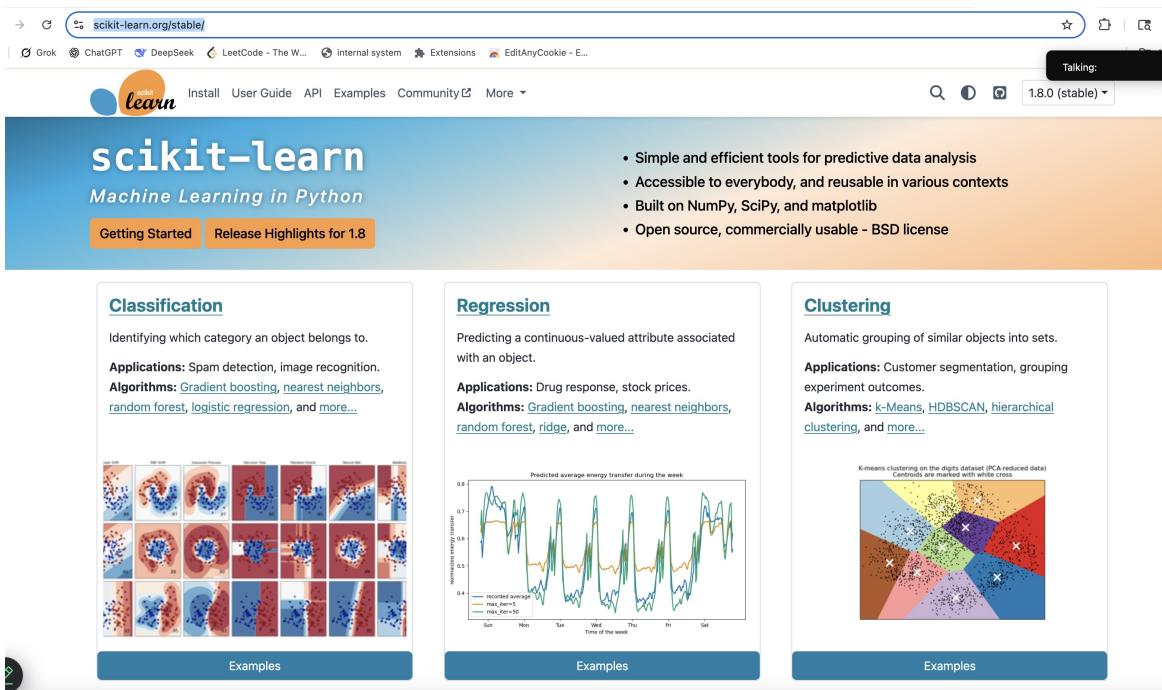
4/ DevOps, python programming, shell programming, computer networking, computer cybersecurity and hopefully if you want to be very good, you would need some lower level

programming language, such as C, Rust or Assembly. [IT/Computer expert]

- For data science or AI track, you need to learn math, python programming, python data structure, algorithms.
 - For devOps, please see above.
 - For pure computer programmer, still going to follow AI track and devOps track for a better career planning.
-
- Unsupervised learning is more leaning on the math part and it would use different algorithms to figure out grouping, clustering or any patterns without much human intervention. On the hand, the supervised learning would need us to tag the answers and train the model.
 - Where is the model being saved? It is similar to database files, such as index files, data files, logs...etc. Model, in theory, before training it is small, after the training, it grows larger.
 - How exactly we get the model? There are many different file types of model we defined by different frameworks. So in order to get these data types or artifact, we would have to write some source code to tailor to our need, and run the code to generate the specific file type, so called model. So it is model source code and we turn it into a binary format, which is consumable.
 - How do I prepare or do a self-training on the AI/ML if I want to be a geek or go ahead of others.

1/ Classic AI/ML (Pattern Recognition, Computer Vision, Robotics, Data Mining). This is prior to GPU and this is CPU centric. Programming API or libraries are the following,

<https://scikit-learn.org/stable/>



CPU centric, all math related algorithms in different classic AI methodologies,

scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

Calibration
Classification
Classifier comparison
Linear and Quadratic Discriminant Analysis with covariance ellipsoid
[Normal, Ledoit-Wolf and OAS Linear Discriminant Analysis for classification](#)
Plot classification probability
Recognizing hand-written digits
Clustering
Covariance estimation
Cross decomposition
Dataset examples
Decision Trees
Decomposition
Developing Estimators
Ensemble methods
Examples based on real world datasets
Feature Selection
Frozen Estimators
Gaussian Mixture Models
Gaussian Process for Machine

	accuracy	macro avg	weighted avg	
	0.97	0.97	0.97	899

We can also plot a [confusion matrix](#) of the true digit values and the predicted digit values.

```
disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")

plt.show()
```

Confusion Matrix

True label

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

Deep Learning

<https://pytorch.org/>

docs.pytorch.org/docs/stable/index.html

PyTorch

Learn Community Projects Docs Blogs & News About JOIN

v2.9.1 (stable) Home Install PyTorch User Guide Reference API Developer Notes Community Tutorials Search the docs ...

Features described in this documentation are classified by release status:

Stable (API-Stable): These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).

Unstable (API-Unstable): Encompasses all features that are under active development where APIs may change based on user feedback, requisite performance improvements or because coverage across operators is not yet complete. The APIs and performance characteristics of these features may change.

[Install PyTorch](#)

[User Guide](#)

[Pytorch Overview](#)

[Get Started](#)

[Learn the Basics](#)

[PyTorch Main Components](#)

[Developer Notes](#)

[Accelerator Integration](#)

On this page
Indices and tables

[Edit on GitHub](#)

[Show Source](#)

PyTorch Libraries

- [torchao](#)
- [torchrec](#)
- [torchft](#)
- [TorchCodec](#)
- [torchvision](#)
- [ExecuTorch](#)
- [PyTorch on XLA Devices](#)

<https://docs.pytorch.org/tutorials/> (1 day)

<https://docs.pytorch.org/tutorials/beginner/basics/intro.html> (a few hours)

<https://www.tutorialspoint.com/pytorch/index.htm> (1 week)

Compiling and running a ChatGPT LLM in local. (Practices)