Classical Machine Learning

Week 0

Plan

- Setting up your learning and programming environment

Getting started

- [Setting up your ML environment (Setup_NYU.ipynb)](Setup_NYU.ipynb)
    - [Choosing an ML environment (Choosing_an_ML_Environment_NYU.ipynb)](Choosing_an_ML_Environment_NYU.ipynb)
- [Quick intro to the tools (Getting_Started.ipynb)](Getting_Started.ipynb)

Week 1

Plan

- Motivate Machine Learning
- Introduce notation used throughout course
- Plan for initial lectures
  - *What*: Introduce, motivate a model
  - *How*: How to use a model: function signature, code (API)
  - *Why*: Mathematical basis -- enhance understanding and ability to improve results

- [Course Overview (Course_overview_NYU.ipynb)](#)
- [Machine Learning: Overview (ML_Overview.ipynb)](#)
- [Intro to Classical ML (Intro_Classical_ML.ipynb)](#)

Using an AI Assistant

AI Assistants are often very good at coding.

But using one to just "get the answer" deprives you of a valuable tool

- you can ask the Assistant *why* it chose to do something
- keep on asking
- treat it as a private tutor !

[Learning about KNN using an Assistant as a private tutor (https://www.perplexity.ai/search/using-python-and-sklearn-pleas-407oe3uzTXu1i9xEHVR2MQ)](https://www.perplexity.ai/search/using-python-and-sklearn-pleas-407oe3uzTXu1i9xEHVR2MQ)

Week 2

Recap of last week

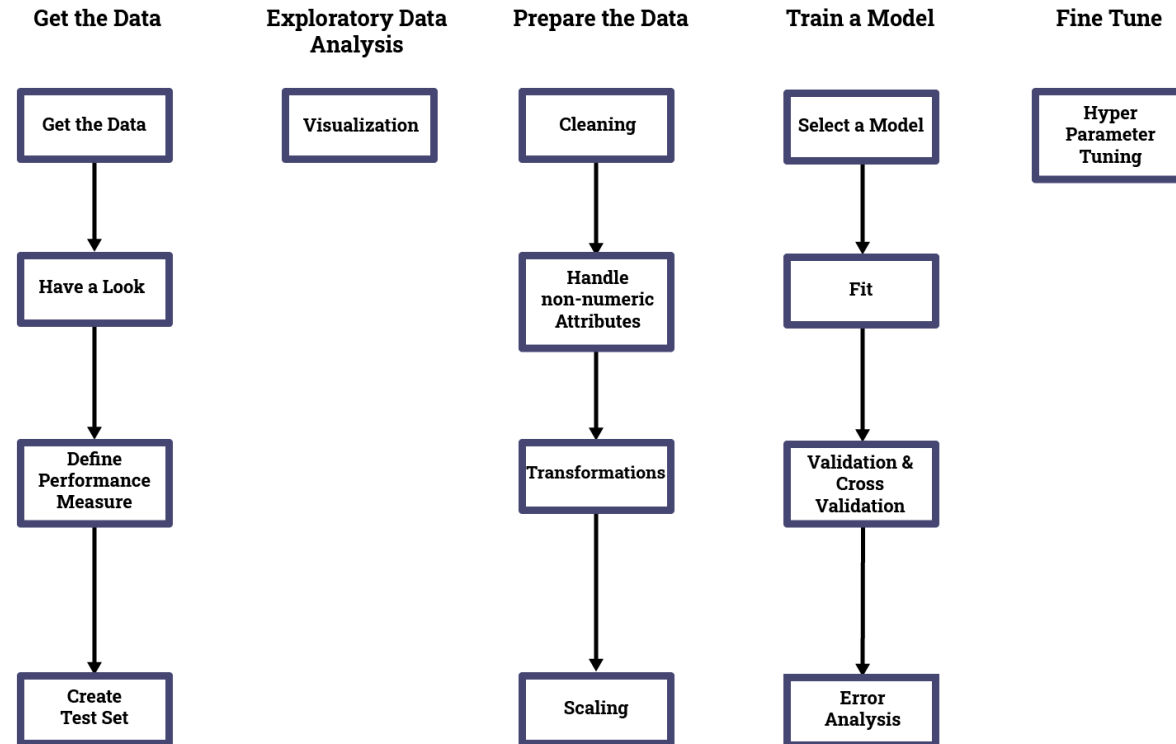- [Summary of Intro to Supervised Machine Learning (Intro_to_Supervised_Learning_Summary.ipynb)](#)

Plan

We will learn the Recipe for Machine Learning, a disciplined approach to solving problems in Machine Learning.

We will illustrate the Recipe while, at the same time, introducing a model for the Regression task: Linear Regression.

Our coverage of the Recipe will be rapid and shallow (we use an extremely simple example for illustration).

I highly recommend reviewing and understanding this [Geron notebook (external/handson-ml2/02_end_to_end_machine_learning_project.ipynb)](#) in order to acquire a more in-depth appreciation of the Recipe.

Recipe for Machine Learning

Get the Data	Exploratory Data Analysis	Prepare the Data	Train a Model	Fine Tune
Get the Data	Visualization	Cleaning	Select a Model	Hyper Parameter Tuning
↓		↓	↓	
Have a Look		Handle non-numeric Attributes	Fit	
↓		↓	↓	
Define Performance Measure		Transformations	Validation & Cross Validation	
↓		↓	↓	
Create Test Set		Scaling	Error Analysis	

Recipe, as illustrated by Linear Regression

[The Recipe for Machine Learning: Solving a Regression task (Recipe_via_Linear_Regression.ipynb)](Recipe_via_Linear_Regression.ipynb)

- A *process* for Machine Learning
  - Go through the methodical, multi-step process
    - Quick first pass, followed by Deeper Dives

Fitting a model: details

Recall: fitting a model (finding optimal value for the parameters) is found by minimizing a Loss function.

Let's examine a typical Loss function for Regression

- [Regression: Loss Function (Linear_Regression_Loss_Function.ipynb)](Linear_Regression_Loss_Function.ipynb)

Iterative training: when to stop

Increasing the number of parameters of a model improves in-sample fit (reduces Loss) but may compromise out-of-sample prediction (generalization).

We examine the issues of having too many/too few parameters.

- [When to stop iterating: Bias and Variance (Bias_and_Variance.ipynb)](Bias_and_Variance.ipynb)

Get the data: Fundamental Assumption of Machine Learning

- [Getting *good* training examples (Recipe_Training_data.ipynb)](Recipe_Training_data.ipynb)

Regression: final thoughts (for now)

- [Regression: coda (Regression_coda.ipynb)](Regression_coda.ipynb)

Deeper dives

- [Fine tuning techniques (Fine_tuning.ipynb)](Fine_tuning.ipynb)

Using an AI Assistant

[Learning about Linear Regression using an Assistant as private tutor (https://www.perplexity.ai/search/using-python-sklearn-and-matpl-vTYv7oGdRO6upR5L5OSirg)](https://www.perplexity.ai/search/using-python-sklearn-and-matpl-vTYv7oGdRO6upR5L5OSirg)

Additional Deep Learning resources

Here are some resources that I have found very useful.

Some of them are very nitty-gritty, deep-in-the-weeds (even the "introductory" courses)

- For example: let's make believe PyTorch (or Keras/TensorFlow) didn't exists; let's invent Deep Learning without it !
  - You will gain a deeper appreciation and understanding by re-inventing that which you take for granted

[Andrej Karpathy course: Neural Networks, Zero to Hero (https://karpathy.ai/zero-to-hero.html)](https://karpathy.ai/zero-to-hero.html)

- PyTorch
- Introductory, but at a very deep level of understanding
  - you will get very deep into the weeds (hand-coding gradients !) but develop a deeper appreciation

fast.ai

`fast.ai` is a web-site with free courses from Jeremy Howard.

- PyTorch
- Introductory and courses "for coders"
- Same courses offered every few years, but sufficiently different so as to make it worthwhile to repeat the course !
  - [Practical Deep Learning (https://course.fast.ai/)](https://course.fast.ai/)
  - [Stable diffusion (https://course.fast.ai/Lessons/part2.html)](https://course.fast.ai/Lessons/part2.html)
    - Very detailed, nitty-gritty details (like Karpathy) that will give you a deeper appreciation

[Stefan Jansen: Machine Learning for Trading (https://github.com/stefan-jansen/machine-learning-for-trading)](https://github.com/stefan-jansen/machine-learning-for-trading)

An excellent github repo with notebooks

- using Deep Learning for trading
- Keras
- many notebooks are cleaner implementations of published models

Assignments

Your assignments should follow the [Assignment Guidelines (assignments/Assignment_Guidelines.ipynb)](assignments/Assignment_Guidelines.ipynb)

```
In [1]: print("Done")
```

Done