

Classical Machine Learning

Week 0

Plan

- Setting up your learning and programming environment

Getting started

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

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 the Landscape of ML (<https://www.perplexity.ai/search/i-am-interested-in-the-landsca-yO63NWfSGS8iHR5nyQYVA>)

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

Week 2 (early start in Week 1)

We began covering the **Recipe**, as illustrated by **Linear Regression**

[The Recipe for Machine Learning: Solving a Regression task
\(Recipe via Linear Regression.ipynb\)](#)

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

Week 2

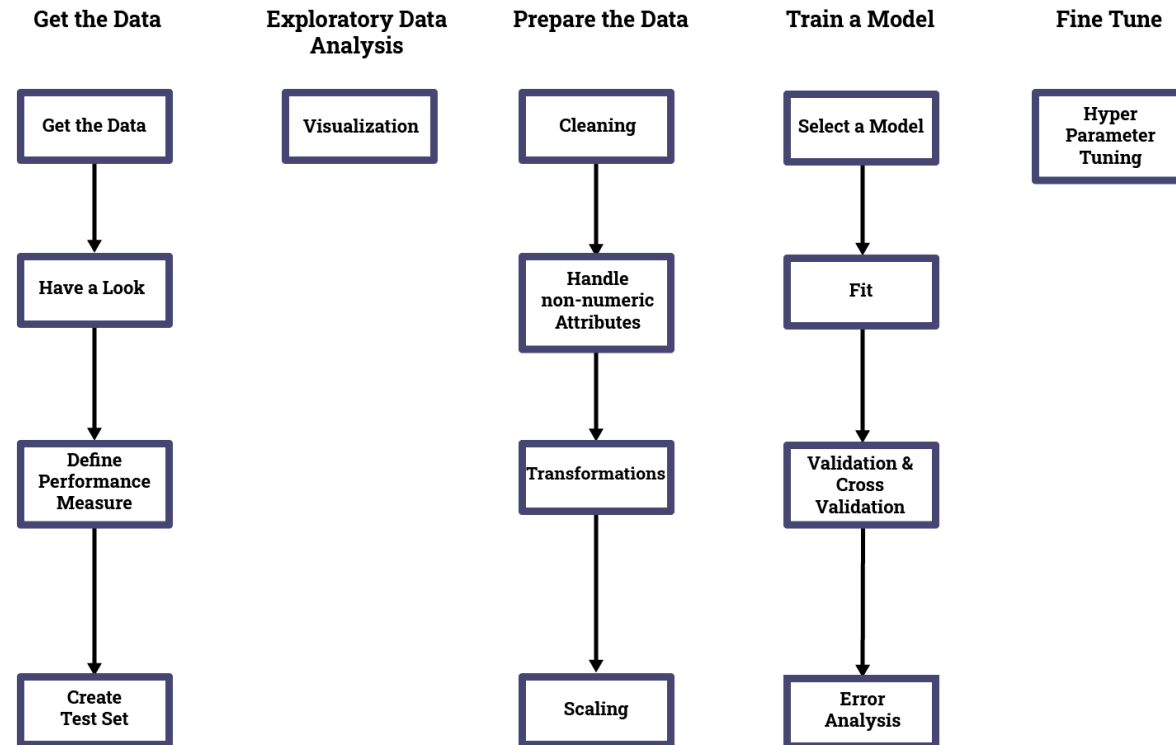
Plan

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, as illustrated by Linear Regression

[The Recipe for Machine Learning: Solving a Regression task \(continued\).](#)
([Recipe via Linear Regression.ipynb#Create-a-test-set](#)).

- 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 \(Linear Regression Loss Function.ipynb\)](#)

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 \(Bias and Variance.ipynb\)](#)

Get the data: Fundamental Assumption of Machine Learning

- [Getting good training examples \(Recipe Training_data.ipynb\)](#)

Regression: final thoughts (for now)

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

Deeper dives

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

Recipe "Prepare the Data" step: Transformations

We discuss the importance of adding *synthetic* features to our Linear Regression example

- and *preview* the *mechanical* process of creating these features via *Transformations*

Transformations

- [Prepare Data: Intro to Transformations \(Prepare_data_Overview.ipynb\)](#)

Validation

Our test dataset can be used only once, yet

- we have an iterative process for developing models
- each iteration requires a proxy for out of sample data to use in the Performance Metric

The solution: create a proxy for out of sample that is a *subset* of the training data.

- [Validation and Cross-Validation \(Recipe via Linear Regression.ipynb#Validation-and-Cross-Validation\)](#)
- [Avoiding cheating in Cross-Validation \(Prepare_data Overview.ipynb#Using-pipelines-to-avoid-cheating-in-cross-validation\)](#)

Week 3 (early start in Week 1)

Classification intro

- [Classification: Overview \(Classification Overview.ipynb\)](#)
- [Classification and Categorical Variables \(Classification Notebook Overview.ipynb\)](#)
 - [linked notebook \(Classification and Non Numerical Data.ipynb\)](#)

Categorical variables (contained as subsections of Classification and Categorical Variables)

We examine the proper treatment of categorical variables (target or feature).

Along the way, we run into a subtle difficulty: the Dummy Variable Trap.

- [Classification and Categorical Variables: Categorical Variables \(Classification Notebook Overview.ipynb#Categorical-variables\)](#)
 - [Categorical variables, One Hot Encoding \(OHE\) \(Categorical Variables.ipynb\)](#)

Week 3

Classification Task

Plan

- We introduce a model for the Classification task: Logistic Regression
- How to deal with Categorical (non-numeric) variables
 - classification target
 - features

Classification intro

- [Classification: Overview \(Classification Overview.ipynb\)](#) **Covered last week**
- [Classification and Categorical Variables \(continued\) \(Classification and Non Numerical Data.ipynb#Recipe-Step-B:-Exploratory-Data-Analysis-\(EDA\)\)](#)
 - [linked notebook \(Classification and Non Numerical Data.ipynb\)](#)

Categorical variables (contained as subsections of Classification and Categorical Variables)

We examine the proper treatment of categorical variables (target or feature).

Along the way, we run into a subtle difficulty: the Dummy Variable Trap.

- [Classification and Categorical Variables: Categorical Variables \(Classification_Notebook_Overview.ipynb#Categorical-variables\)](#)
 - [Categorical variables, One Hot Encoding \(OHE\) \(Categorical_Variables.ipynb\)](#)

Multinomial Classification

We generalize Binary Classification into classification into more than two classes.

- [Multinomial Classification \(Multinomial_Classification.ipynb\)](#)

Classification and Categorical variables wrapup

- [Classification Loss Function \(Classification_Loss_Function.ipynb\)](#)
- [Baseline model for Classification \(Classification_Baseline_Model.ipynb\)](#)
- [OHE issue: Dummy variable trap \(Dummy_Variable_Trap.ipynb\)](#)

Classification: final thoughts (for now)

- [Classification: coda \(Classification_coda.ipynb\)](#)

Plan Deeper dives

Week 4

Good news

- You now know two main tasks in Supervised Learning
 - Regression, Classification
- You now know how to use virtually every model in `sklearn`
 - Consistent API
 - `fit`, `transform`, `predict`
- You survived the "sprint" to get you up and running with ML

Time to re-visit, in more depth, several important topics

Transformations: the "why"

Part of becoming a better Data Scientist is transforming raw features into more useful synthetic features.

We focus on the necessity (the "why"): transforming raw data into something that tells a story.

We will then discuss the [mechanics \(Prepare data Overview.ipynb\)](#) (how to use `sklearn` to implement transformation Pipelines) of Transformations.

- [Becoming a successful Data Scientist \(Becoming_a_successful_Data_Scientist.ipynb\)](#)
- [Transformations: overview \(Transformations_Overview.ipynb\)](#)
 - linked notebooks:
 - [Transformations: adding a missing feature \(Transformations_Missing_Features.ipynb\)](#)

Transformations: the "how"

Having hopefully motivated the use of transformations in theory

- we turn to the *mechanical* process of creating these features via *Transformations* in *sklearn*

Transformations

- [Prepare Data: Intro to Transformations \(Prepare_data_Overview.ipynb\)](#)

Transformations: Avoiding cheating when using Cross-Validation

Our test dataset can be used only once, yet

- we have an iterative process for developing models
- each iteration requires a proxy for out of sample data to use in the Performance Metric

The solution: create a proxy for out of sample that is a *subset* of the training data.

- [Validation and Cross-Validation \(Recipe via Linear Regression.ipynb#Validation-and-Cross-Validation\)](#) (Covered in week 1)

Error Analysis

- We explain Error Analysis for the Classification Task, with a detailed example
- How Training Loss can be improved
- [Error Analysis \(Error_Analysis_Overview.ipynb\)](#)
 - [linked notebook \(Error_Analysis.ipynb\)](#)
 - Summary statistics
 - Conditional statistics
 - [Worked example \(Error_Analysis_MNIST.ipynb\)](#)
- [Loss Analysis: Using training loss to improve models \(Training_Loss.ipynb\)](#)

Imbalanced data

- [Imbalanced data \(Imbalanced_Data.ipynb\)](#)

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 exist; 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>)

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](#)).

Regression

- Assignment notebook: [Using Machine Learning for Hedging \(assignments/Regression%20task/Using_Machine_Learning_for_Hedging.ipynb\)](#)
- Data
 - There is an archive file containing the data
 - You can find it
 - Under the course page: Content --> Data --> Assignments --> Regression task
 - You won't be able to view the file in the browser, but you **will** be able to Download it
 - You should unzip this archive into the *the same directory* as the assignment notebook
 - The end result is that the directory should contain
 - The assignment notebook and a helper file
 - A directory named Data

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

Done

