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: Introduction

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: Regression task

Plan

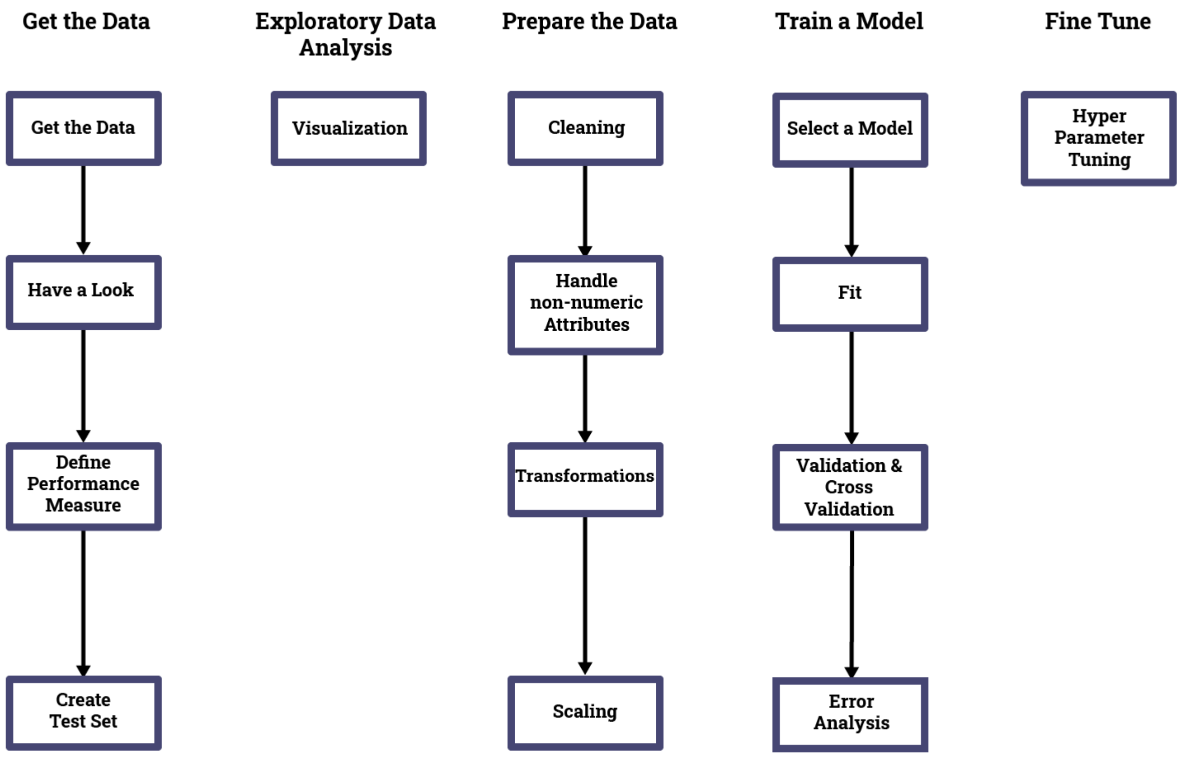
We will learn the Recipe for Machine Learning, a disciplined approach to solving problems in Machine Learning.

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



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

Non-feature dimensions

In response to questions about Assignment 1,

- we will clarify the limitations in our ability to handle *timeseries* data with our current tools.

Non-feature dimensions: preview (Non-feature_dimensions_preview.ipynb)

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)

Error Analysis

We can only improve our model's out of sample Performance Metric

- by diagnosing the in-sample errors
- that is the goal of the Error Analysis step of the Recipe
- We explain Error Analysis for the Classification Task, with a detailed example
- How Training Loss can be improved

The conversion of a probability (e.g., model output) to a Class (categorical variable) for Classification

- often involves the comparison of a probability to a threshold

- we show how varying the threshold changes the conditional Performance Metric for Classification

    - the threshold is a hyper-parameter, thus this is a kind of Fine-Tuning

- Error Analysis (Error_Analysis_Overview.ipynb)

    - linked notebook (Error_Analysis.ipynb)
        - Summary statistics
        - Conditional statistics
    - Worked example (Error_Analysis_MNIST.ipynb)**Deferred**

- Loss Analysis: Using training loss to improve models (Training_Loss.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)

Week 4: Transformations

Plan

Now you know how to create models. What happens if the Performance Metric for your model is disappointing ?

The first step is recognizing the issue, and diagnosing it. That is the role of Error Analysis.

The second step is attempting to improve Performance. Quite often we will need to perform *Feature Engineering*.

We explain

- why it is often necessary to create *synthetic* features to augment or replace *raw* feature
- the mechanical process in `sklearn` that makes the application of transformations easy and consistent

Error Analysis: worked example (deferred from prior week)

- Error Analysis (Error_Analysis_Overview.ipynb)

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 (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**)

Week 5: Other Classification models

Imbalanced data

- Imbalanced data (Imbalanced_Data.ipynb)

More models for classification

Plan

- More models: Decision Trees, Naive Bayes, Support Vector Classifier
    - Different flavor: more procedural, less mathematical
    - Decision Trees: a model with *non-linear* boundaries
- Ensembles
    - Bagging and Boosting
    - Random Forests

Decision Trees, Ensembles

- Decision Trees: Overview (Decision_Trees_Overview.ipynb)
- Decision Trees (Decision_Trees_Notebook_Overview.ipynb)
    - linked notebook (Decision_Trees.ipynb)
- Trees, Forests, Ensembles (Ensembles.ipynb)

Naive Bayes

- Naive Bayes (Naive_Bayes.ipynb)

Support Vector Classifiers

- Support Vector Machines: Overview (SVM_Overview.ipynb)
- SVC Loss function (SVM_Hinge_Loss.ipynb)
- SVC: Large Margin Classification (SVM_Large_Margin.ipynb)
- SVM: Kernel Transformations (SVM_Kernel_Functions.ipynb)
- SVM Wrapup (SVM_Coda.ipynb)

Classification: final thoughts

- Classification: coda -- review again (Classification_coda.ipynb)

  SVC conversation (https://www.perplexity.ai/search/what-is-the-relationship-betwe-Pq8r22pISH.gUGmM1gkgbg#4)

Week 6: Unsupervised Learning

More models for classification (continued)

SVC: review

- SVC: Key points (SVM_Large_Margin.ipynb#SVC:-Key-points)
- SVC conversation (https://www.perplexity.ai/search/what-is-the-relationship-betwe-Pq8r22pISH.gUGmM1gkgbg#4)

SVM (deferred from last week)

- SVM: Kernel Transformations (SVM_Kernel_Functions.ipynb)
- SVM Wrapup (SVM_Coda.ipynb)

Naive Bayes (deferred from last week)

- Naive Bayes (Naive_Bayes.ipynb)

Classification coda

- Classification: probability distribution over classes (Classification_coda.ipynb#Output:-probabilities-or-just-classes-?)

Unsupervised Learning

Unsupervised Learning: PCA

- Unsupervised Learning: Overview (Unsupervised_Overview.ipynb)
- PCA Notebook Overview (Unsupervised_Notebook_Overview.ipynb)
    - linked notebook (Unsupervised.ipynb)
- PCA in Finance (PCA_Yield_Curve_Intro.ipynb)

Unsupervised Learning: PCA (continued)

- Importance of number of components: visualization (Unsupervised.ipynb#Visualizing-the-fidelity-of-the-reduced-dimension-representation)
- Interpreting the components (Unsupervised.ipynb#Can-we-interpret-the-components-?)

Week 7: DL Week 1 Introduction to Neural Networks and Deep Learning

Bridge between Classical ML and Deep Learning

Gradient Descent

Machine Learning is based on minimization of a Loss Function. Gradient Descent is one algorithm to achieve that.

- [Gradient Descent (Gradient_Descent.ipynb)](Gradient_Descent.ipynb)

Recommender Systems (Pseudo SVD)

How does Amazon/Netflix/etc. recommend products/films to us ? We describe a method similar to SVD but that is solved using Gradient Descent.

This theme of creating a custom Loss Functions and minimizing it via Gradient Descent is a recurring theme in the upcoming Deep Learning second half of the course.

- [Recommender Systems (Recommender_Systems.ipynb)](Recommender_Systems.ipynb)
- [Preview: Some cool Loss functions (Loss_functions.ipynb#Loss-functions-for-Deep-Learning:-Preview)](Loss_functions.ipynb#Loss-functions-for-Deep-Learning:-Preview)

Deeper Dives

- Other matrix factorization methods (Unsupervised_Other_Factorizations.ipynb)

Classical ML: deeper dives

Loss functions: mathematical basis (deferred)

Where do the Loss functions of Classical Machine Learning come from ? We take a brief mathematical detour into Loss functions.

- Entropy, Cross Entropy, and KL Divergence (Entropy_Cross_Entropy_KL_Divergence.ipynb)
- Loss functions: the math (Loss_functions.ipynb)
    - Maximum likelihood
    - Preview: custom loss functions and Deep Learning

Deeper Dives

- Linear Regression in more depth (Linear_Regression_fitting.ipynb)
- Interpretation: Linear Models (Linear_Model_Interpretation.ipynb)
- Missing data: clever ways to impute values (Missing_Data.ipynb)
- Feature importance (Feature_Importance.ipynb)
- SVC Loss function derivation (SVM_Derivation.ipynb)

Deep Learning: Introduction

Plan

Deep Learning/Neural networks

- Set up your Tensorflow environment (Tensorflow_setup.ipynb)
- Neural Networks Overview (Neural_Networks_Overview.ipynb)

Neural network: practical

- Coding Neural Networks: Keras
    - Intro to Keras (Keras_intro.ipynb)
    - **Note**
        - If you have problems using the `plot_model` function in Keras on your local machine: see here (Setup_ML_Environment_NYU.ipynb#Tools-for-visualization-of-graphs-(optional)) for a fix.
    - Linked notebooks
        - DNN Keras example (DNN_Keras_example.ipynb) **local machine**

DL Week 2 Intro to NN (continued); Convolutional Neural Networks

Introduction (continued)

Here is a quick review of Neural Networks

- Neural Network summary (Neural_Network_summary.ipynb)

Overview continued:

- Overview (continued) (Neural_Networks_Overview.ipynb#What-is-$W_\llp$-?-Where-did-Θ-go-?)

Coding a Neural Network

- Coding Neural Networks: Keras
    - Intro to Keras (Keras_intro.ipynb) **covered last lecture**
    - **Note**
        - If you have problems using the `plot_model` function in Keras on your local machine: see here (Setup_ML_Environment_NYU.ipynb#Tools-for-visualization-of-graphs-(optional)) for a fix.
    - Linked notebooks
        - DNN Keras example (DNN_Keras_example.ipynb) **local machine**
        - DNN Keras example Notebook from github (https://colab.research.google.com/github/kenperry-public/ML_Fall_2025/blob/master/DNN_Keras_example.ipynb) (**Google Colab**)
- Practical Colab
    - **Colab**: Practical Colab Notebook from github (https://colab.research.google.com/github/kenperry-public/ML_Fall_2025/blob/master/Colab_practical.ipynb)

Practical advice (continued)

- Karpathy: Recipe for training Neural Nets (Karpathy_Recipe_for_training_NN.ipynb)

NN: in depth

Plan

The topics introduced in the Neural Networks Overview are now covered more in-depth.

- Where do Neural Networks get their power from ?
- How exactly do we compute the gradients ?
- How does a special language/library facilitate automatic computation of the gradients ?

Neural network theory

- A neural network is a Universal Function Approximator (Universal_Function_Approximator.ipynb)

Training Neural Networks (introduction)

- Intro to Training (Neural_Networks_Intro_to_Training.ipynb)
- Training Neural Networks - Back propagation (Training_Neural_Network_Backprop.ipynb)

How to compute gradients automatically

- Why TensorFlow ?: Gradients made easy (Training_Neural_Network_Operation_Forward_and_Backward_Pass.ipynb)

Convolutional Neural Network (CNN) Layer

Plan

Deeper Dives

We introduce a new layer type.

This is motivated by inputs with dimensions in addition to the feature dimension.

The Convolutional Neural Network layer type

- Introduction to CNN (Intro_to_CNN.ipynb)
    - CNN pictorial (CNN_pictorial.ipynb)
- Notational standards, definitions (CNN_Notation.ipynb)
- CNN: Space and Time (CNN_Space_and_Time.ipynb)
    - CNN example from github (https://colab.research.google.com/github/kenperry-public/ML_Fall_2025/blob/master/CNN_Keras.ipynb) (**Colab**)
    - CNN example from github (CNN_Keras.ipynb) (**local machine**)

The following notebooks are an older attempt at a *visual* explanation of the CNN.

Hopefully, the "Introduction" notebook is more intuitive and may supercede these visual notebooks.

- CNN: explained in pictures (CNN_Overview.ipynb)

Deeper dives

- Convolution as Matrix Multiplication (CNN_Convolution_as_Matrix_Multiplication.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 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)

- 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/)
    - Stable diffusion (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`

Classification

- Assignment notebook: [Ships in satellite images (assignments/Classification%20task/Ships_in_satellite_images.ipynb#)](assignments/Classification%20task/Ships_in_satellite_images.ipynb#)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Classification 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

Midterm Project: Bankruptcy One Year Ahead

- Assignment notebook Bankruptcy One Year Ahead (assignments/bankruptcy_one_yr/Bankruptcy_oya.ipynb)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Bankruptcy One Year Ahead
        - 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`

Keras practice

- Assignment notebook Ships in satellite images: Neural Network (assignments/keras_intro/Ships_in_satellite_images_P1.ipynb)
- Data (same as for the Classification assignment)

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

Done
```