

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

Week 2

Recap of last week

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

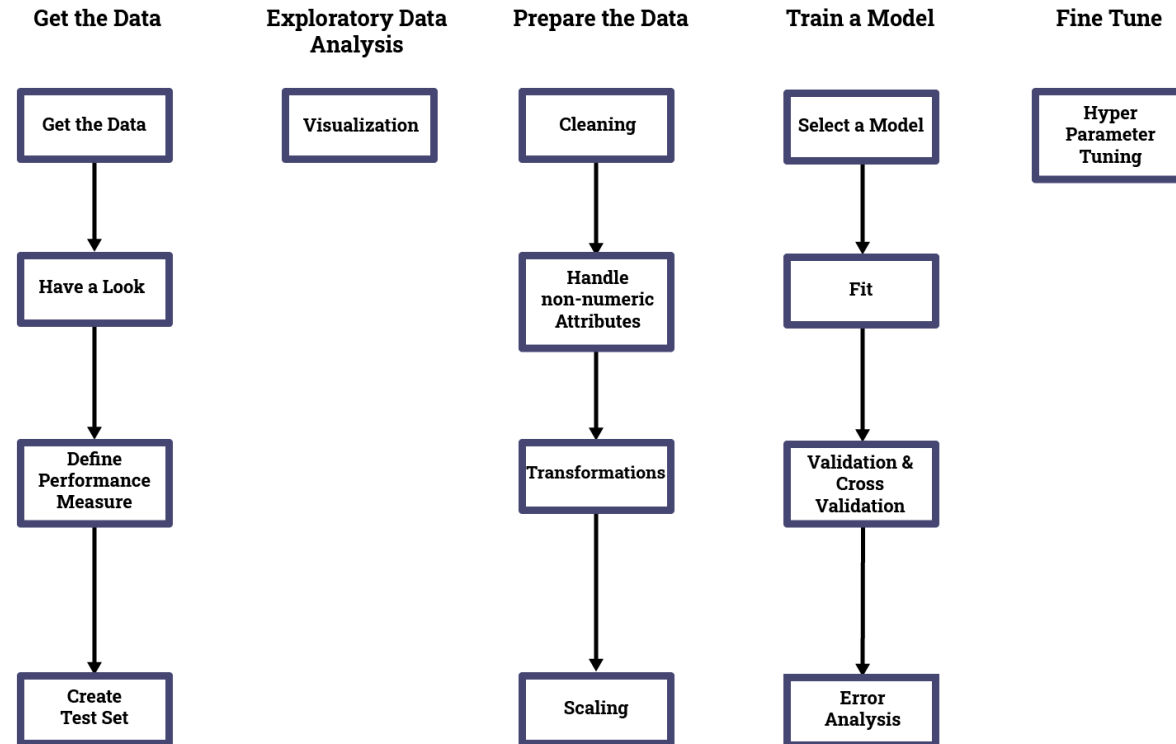
Plan

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, 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

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 \(Fine_tuning.ipynb\)](#)

Using an AI Assistant

[Learning about Linear Regression using an Assistant as private tutor \(https://www.perplexity.ai/search/using-python-sklearn-and-matplotlibTYv7oGdRO6upR5L5OSirg\)](https://www.perplexity.ai/search/using-python-sklearn-and-matplotlibTYv7oGdRO6upR5L5OSirg)

Week 3

Recipe "Prepare the Data" step: Transformations

We covered this at end of Week 2

We recap 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

We skipped over this topic in Week 2; we cover it now

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 \(Prepare data Overview.ipynb#Using-pipelines-to-avoid-cheating-in-cross-validation\)](#)

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 \(Classification Overview.ipynb\)](#)
- [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 \(Classification Notebook Overview.ipynb#Categorical-variables\)](#)

Week 4

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

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
- You know the *mechanical process* to implement transformations: Pipelines

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

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 \(Imbalanced Data.ipynb\)](#)

Transformations: the "why"

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

In an earlier week, we presented the [mechanics \(Prepare data Overview.ipynb\)](#) (how to use `sklearn` to implement transformation Pipelines) of Transformations. This week, we focus on the necessity (the "why"): transforming raw data into something that tells a story

- [Becoming a successful Data Scientist \(Becoming a successful Data Scientist.ipynb\)](#)
- [Transformations: overview \(Transformations Overview.ipynb\)](#)
 - linked notebooks:

Week 5

Transformations: the "why" (continued)

- [Becoming a successful Data Scientist](#)
[\(Becoming a successful Data Scientist.ipynb\)](#)
- [Transformations: overview](#) ([Transformations Overview.ipynb#Scaling](#))
 - linked notebooks:
 - [Transformations: scaling](#) ([Transformations Scaling.ipynb](#))
 - [Transformations: normalization](#)
[\(Transformations Normalization.ipynb\)](#)
 - [Other Transformations](#) ([Transformations Other.ipynb](#))

Loss functions: mathematical basis

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

More models for classification

Plan

- More models: Decision Trees, Naive Bayes
 - 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 \(Naive Bayes.ipynb\)](#)

Week 6

More models for classification

Plan

Continue with more models for classification.

We continue with the *ensemble* technique that *combines* the prediction of multiple models.

Combining multiple models: Ensembles (continued)

- [Trees, Forests, Ensembles \(Ensembles.ipynb#Boosting\)](#)

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 \(Classification_coda.ipynb\)](#)

Loss functions: mathematical basis (deferred from previous week)

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

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 \(PCA_Yield_Curve_Intro.ipynb\)](#)

Week 7

We continue with the Unsupervised Learning topic.

Unsupervised Learning

Unsupervised Learning: PCA (continued)

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

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 \(SVM_Derivation.ipynb\)](#)

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 \(Gradient_Descent.ipynb\)](#)

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

Deeper Dive

Deep Learning

DL Week 1 Introduction to Neural Networks and Deep Learning

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: Tensorflow, Keras
 - [Intro to Keras \(Tensorflow Keras.ipynb\)](#)
 - **Note**
 - If you have problems using the `plot_model` function in Keras on local machine: see [here \(Setup ML Environment NYU.ipynb#Too visualization-of-graphs-\(optional\)\)](#) for a fix.
 - Linked notebooks
 - [DNN Tensorflow example Notebook local \(DNN TensorFlow example.ipynb\)](#) (**local machine**)
 - [DNN Tensorflow example Notebook from github \(https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/DNN_TensorFlow_example\)](https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/DNN_TensorFlow_example) (**Google Colab**)
- Practical Colab
 - **Colab:** [Practical Colab Notebook from github \(https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/Colab_practical.ipynb\)](https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/Colab_practical.ipynb)

Practical advice

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

Here is a quick review of Neural Networks

- [Neural Network summary](#) ([Neural Network summary.ipynb](#)).

Practical advice (continued)

- Karpathy: [Recipe for training Neural Nets](#) ([Karpathy Recipe for training NN.ipynb](#)).

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 \(Training Neural Network Operation Forward and Backward Pass.ipynb\)](#)

Deeper Dives

- [Keras, from past to present \(Tensorflow Keras Archaeology.ipynb\)](#)
- [History/Computation Graphs: Tensorflow version 1 \(DNN TensorFlow Using TF version 1.ipynb\)](#)
- [Raw TensorFlow example Notebook from github \(https://colab.research.google.com/github/kenperry-public/ML_Spring_2024/blob/master/Raw_TensorFlow.ipynb\) \(Colab\)](#)

In [1]: ****Convolutional Neural Networks (CNN)****

****Plan****

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_2024/blob/master/CNN_demo.ipynb) (****Colab****)
 - [CNN example **from github**](CNN_demo.ipynb) (****local machine****)
- Code
 - [CNN example **from github**](https://colab.research.google.com/github/kenperry-public/ML_Fall_2024/blob/master/CNN_demo.ipynb) (****Colab****)
 - [CNN example **from github**](CNN_demo.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)
```

Cell In[1], line 1

DL Week 3 Training Neural Networks: details

SyntaxError: invalid syntax

Plan

- Why training a Neural Network can be difficult: fine-details of training
- Introduce a new layer type: Recurrent layers
 - Part of our "sprint": final layer type
 - Will revisit more theoretical issues in subsequent lectures

Training Neural Networks: the fine details

- [The dynamics of training \(Training Neural Networks Overview.ipynb\)](#)
 - Effects of changing: activation functions; weight initialization
 - initialization and scaling
 - dropout
 - learning rate schedules
 - vanishing/exploding gradients

Recurrent Neural Networks (RNN)

- [Introduction to Recurrent Neural Network \(RNN\) \(Intro to RNN.ipynb\)](#)

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

Classification

- Assignment notebook: [Ships in satellite images](#)
([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

