

# 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

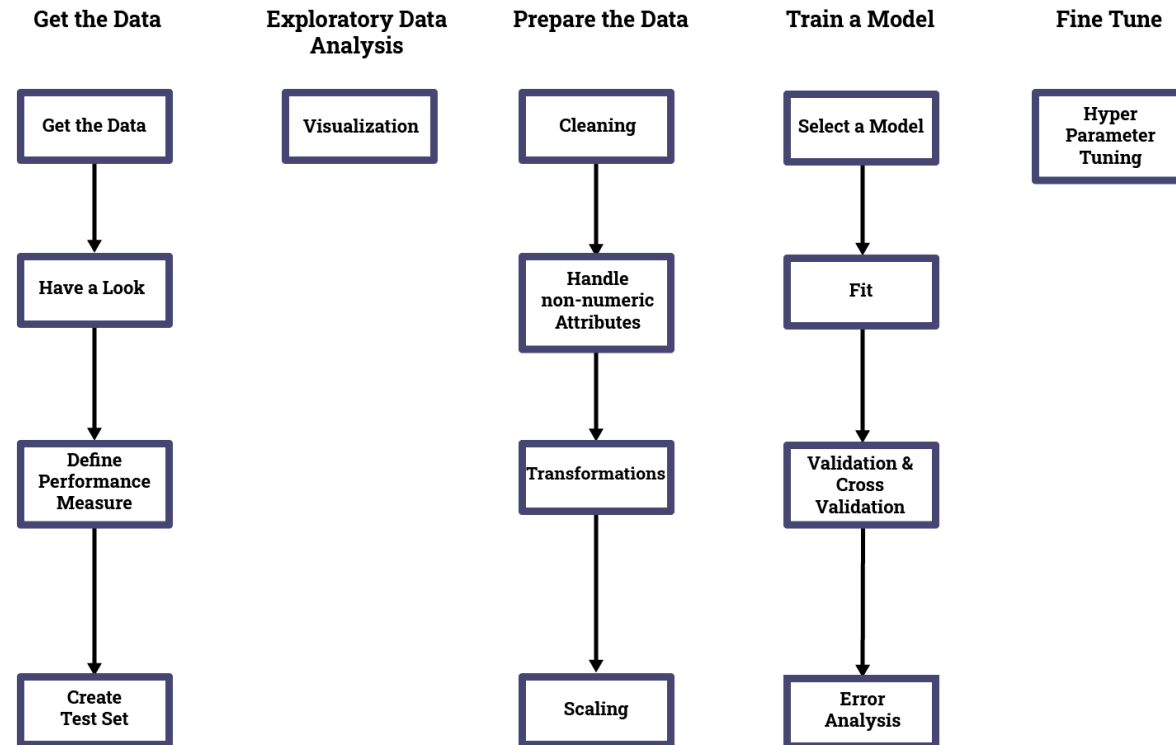
## Plan

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, 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 \(Error Analysis Overview.invnh\)](#)

# 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 \(SVM Coda.ipynb\)](#)

# Week 6: 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 \(Unsupervised.ipynb#Can-we-interpret-the-components-?\)](#)

# 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 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](#))

# 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

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

Done

