Advanced Data Analysis with Python

Cecilia Graiff

November 20, 2025

ALMAnaCH, Inria Paris - École d'Affaires Publiques, SciencesPo
cecilia.graiff@sciencespo.fr

Resources for this class

- McGill lecture on Empirical Risk Minimization
- Google Developers Machine Learning Class Course
- McGill lectures on Feature Engineering

Questions about last week's topic

Machine Learning Workflows

From statistics to machine learning

- We studied the **core statistical methods** of data analysis: regression, classification, times series, ...
- Statistics and machine learning **partially overlap**

From statistics to machine learning

- We studied the **core statistical methods** of data analysis: regression, classification, times series, ...
- Statistics and machine learning **partially overlap**
- ML: more focus on **predictions**, statistics: more focus on **inference and patterns interpretation**

- How do we build a machine learning workflow?

- How do we build a machine learning workflow?
- Where is the shift between statistics and machine learning?

- How do we build a machine learning workflow?
- Where is the shift between statistics and machine learning?
- When do we need machine learning?

What is machine learning

Definition (Machine Learning)

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can **learn from data and generalise to unseen data**, and thus perform tasks without explicit instructions.

Source: Wikipedia

Learning Paradigms

- **Supervised Learning**: an algorithm maps input to output based on example input-output pairs (**labeled data**).
- **Unsupervised Learning**: an algorithm learns patterns from **unlabeled data**.

Supervised Learning

Let D be a dataset of n observations:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

- $x_i \in \mathbb{R}^p$ — feature vector for observation i
- $y_i \in \mathcal{Y}$ — label (continuous for regression, categorical for classification)

Supervised Learning

Let D be a dataset of n observations:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

- $x_i \in \mathbb{R}^p$ — feature vector for observation i
- $y_i \in \mathcal{Y}$ — label (continuous for regression, categorical for classification)

ML formulation: Learn a function

$$f_\theta : \mathbb{R}^p \to \mathcal{Y}$$

that maps features to labels.

Unsupervised Learning

Let D be a dataset of n observations without labels:

$$\mathcal{D} = \{x_1, x_2, \dots, x_n\}, \quad x_i \in \mathbb{R}^p$$

Unsupervised Learning

Let D be a dataset of n observations without labels:

$$\mathcal{D} = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^p$$

ML formulation: Discover structure or patterns in the data, such as clusters or low-dimensional representations.

ML model workflow

1. Data collection

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning
4. Feature engineering

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning
4. Feature engineering
5. Data splitting

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning
4. Feature engineering
5. Data splitting
6. Model training

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning
4. Feature engineering
5. Data splitting
6. Model training
7. Model evaluation

ML model workflow

1. Data collection
2. Exploratory data analysis and model selection
3. Data cleaning
4. Feature engineering
5. Data splitting
6. Model training
7. Model evaluation
8. Model tuning

Feature Engineering

Feature Engineering

Definition

Feature engineering is the process of transforming the raw data into a **targeted type of input**.

Feature Engineering

Definition

Feature engineering is the process of transforming the raw data into a **targeted type of input**.

It includes:

- Target transformations
- **Feature extraction**
- Feature encoding

In this lecture, we will talk about feature extraction. Feature extraction and encoding are more useful for more complex types of models (e.g. neural networks).

Binning

- **Binning** consists of converting a continuous numerical feature into discrete "bins" or intervals, where each bin represents a **range**.

Log Transform

- Takes a numeric feature x and replaces it with $log(x)$
  - Careful: can only work **on positive values!**

Log Transform

- Takes a numeric feature x and replaces it with $log(x)$
    - Careful: can only work **on positive values!**
- Useful to:
    - Reduce skewness
    - Handle large ranges
    - Reduce effect of outliers

Scaling: Normalization

Normalization involves scaling all values in a fixed range between 0 and 1:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

- **Scaling** is useful for comparing variables that are in a very different range, such as **age** and **income**
- Effect of **outliers** increases: handle them before!

Scaling: Standardization

Standardization scales the values while taking into account standard deviation:

$$x' = \frac{x - \mu}{\sigma}$$

- **Reduces** effect of **outliers**

Other feature engineering techniques

- We handled in the past lectures some feature engineering techniques, such as:
  - Outlier detection (with boxplots)
  - Dataframe grouping and splitting
- Several other techniques exist, also more advanced ones (BagOfWords, Principal Component Analysis, clustering...)

Data handling in ML

Data

- Golden rule: **garbage in, garbage out**
- Some data quality standards to always check for:
  - Missing values
  - Missing features (e.g., the demographics info for the patients are not fully present)
  - Duplicates
  - Imbalanced datasets
- One main question: **Do I have the necessary data to answer my research question?**

Data splitting

- Core concept of ML: **data splitting**
- `train` and `test` set; eventually **evaluation** set

Data splitting

- Core concept of ML: **data splitting**
- `train` and `test` set; eventually **evaluation** set
- **Why splitting?**
  - Allows you to **test the model on unseen data**
  - Prevents overfitting
  - Ensures generalization

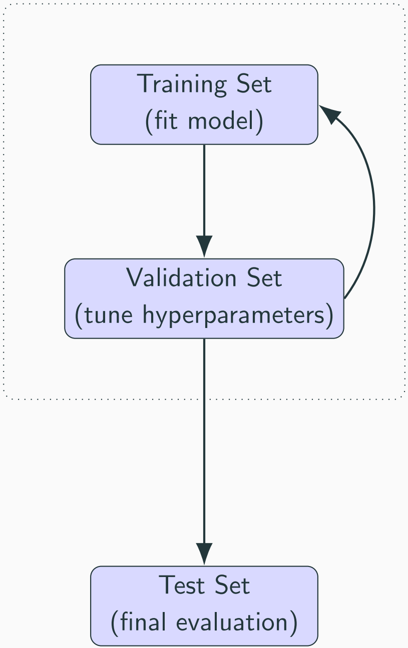
Usual splits

- `train`: the model is **trained** on the training set
- `test`: the model is tested on the **test set**, which consists of unseen data
- `eval`: testing **during training** for hyperparameter finetuning and other adjustements

Usual splits

- `train`: the model is **trained** on the training set
- `test`: the model is tested on the **test set**, which consists of unseen data
- `eval`: testing **during training** for hyperparameter finetuning and other adjustements

Repeatedly testing the model on the test set to adjust parameters can lead to the model **memorizing patterns from the test set** and thus to misleading results, hence the necessity of an **evaluation set**.


Training Set
(fit model)
Validation Set
(tune hyperparameters)
Test Set
(final evaluation)

Common splits

- 70% train, 15% test, 15% eval
- 80% train, 10% test, 10% eval
- The best proportion is very **dataset-dependent**: you will have to **experiment with the data** to find it.

Imbalanced datasets

Example of **classification** problem:

- Dataset is **imbalanced** if the ratio of different classes presents a strong misproportion.

Consequences

- The model does not learn the **features of each class**
- The model does not learn the **class distribution**

Stratified split and k-fold cross-validation

- Stratified split (available in `sklearn`): to respect a specific balance in the data split
- K-fold cross-validation: if the dataset is very small, chances are that the test samples will be imbalanced. You can avoid this by:
  - **dividing the dataset into K equal-sized folds**
  - For each iteration, training on $K-1$ folds and testing on the remaining one

Downsampling and Upweighting

If the dataset is strongly imbalanced, these methods will not be effective. In that case, you can proceed with:

1. Step 1: **downsampling**
2. Step 2: **upweighting**.

Downsampling

- Artificially reduce the majority class
- Consequence: the minority class will be **represented enough**

Downsampling

- Artificially reduce the majority class
- Consequence: the minority class will be **represented enough**

With this method, you introduce a **prediction bias** (careful: this is different from the **bias term** that we talked about in the previous lectures!

Upweighting

- To deal with the **prediction bias**, the **errors** need to have a bigger weight
- **You must "upweight" the majority classes by the factor to which you downsampled**
- Meaning: when the model mistakenly predicts the majority class, treat the **loss** as if it were 25 errors (multiply the regular loss by 25).

Imbalanced datasets: caveat

- While **class imbalance** is a very specific problem of **classification approaches**, the data distribution is essential to the **data quality**
- Therefore: **imbalanced datasets** are always an issue, not only in classification
- Examples:
  - Biases in linear regression: e.g. wage data: your test set contains many **high salaries**, but we know that this is only the case for a few people
  - Therefore: importance of accurate **variable selection** and **data splitting**.

Model validation and tuning

Hyperparameters

Definition

Hyperparameters are the parameters that can be set in order to define any configurable part of a model's learning process.

Hyperparameters

Definition

Hyperparameters are the parameters that can be set in order to define any configurable part of a model's learning process.

- Not all models have hyperparameters; usually for more complex models, e.g. neural networks
- **OLS regression**: normally no hyperparameter; if regulated, it can have hyperparameters

Loss Function

Definition

The **loss function** $L(y, f(x))$ is a **point-wise measure** of errors in predictions for a single x.

$$L : Y \times \mathbb{R}^g \to \mathbb{R}$$

Loss Function

Definition

The **loss function** $L(y, f(x))$ is a **point-wise measure** of errors in predictions for a single x.

$$L : Y \times \mathbb{R}^g \to \mathbb{R}$$

- Quantifies **how wrong the model is**
- Goal: **minimize the loss**
- We discussed **evaluation metrics** for each presented method - they are useful to **computing the loss** (attention: several ways of doing that!)

Risk Minimization

Definition

The **risk** of a model is the **expected loss**.

$$R(f) := \mathbb{E}_{x,y}[L(y, f(x))] = \int L(y, f(x))\, dP_{x,y}$$

Where:

- $L(y, f(x))$ is the loss
- $P_{x,y}$ is the data distribution

Usually, we do not have access to the **real distribution**, but to a **training set and test set** that **represent the data**. (Remember the population vs sample differentiation in statistics!)

- Therefore: one possible methos is **empirical risk minimization**

Overfitting

Definition

We speak of **overfitting** when the models learns the training set so closely that it fails to **generalize** on unseen data.

- Real-world problem! In **applicative settings**, you need your model to be able to predict on new data as accurately as possible.

Underfitting

Definition

An **underfitting** model fails to performs well even on the training data.

Overfitting and underfitting example

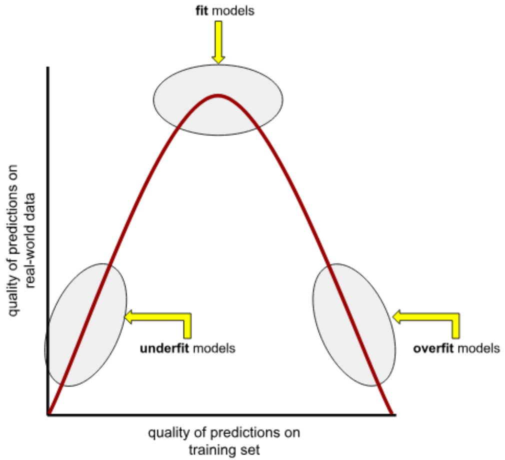


Figure 14. *Underfit, fit, and overfit models.*

Source: Google Developers Crash Course on ML.

How do you detect overfitting?

- Indicator: **loss function**
- Plot the loss function; a plot with several loss function is called the **generalization plot**
- If the model **overfits**, the loss curve is good at first, and then **diverges**

Regularization

- The idea behind **regularization** is to minimize both **loss** and **complexity**, thus fitting the data **as well as possible** but also **as simply as possible**.

Regularization

- The idea behind **regularization** is to minimize both **loss** and **complexity**, thus fitting the data **as well as possible** but also **as simply as possible**.
- Two base methods:
  - **L1 regularization (used for LASSO)**: adds a penalty based on the **absolute value** of coefficients.
  - **L2 regularization (used for ridge regression)**: adds a penalty based on the **square** of the coefficients.

L1 Regularization (LASSO)

Lasso adds an L_1 penalty:

$$\text{Loss} = \text{MSE} + \lambda \sum_{i=1}^{n} |w_i|$$

Characteristics:

- w_i are the **weights** (=coefficients)
- Produces sparse models (many $\beta_j = 0$)
- Performs feature selection
- Unstable when predictors are highly correlated

L2 Regularization (RIDGE)

Ridge adds an L_2 penalty on coefficient magnitude.

$$\text{Loss} = \text{MSE} + \lambda \sum_{i=1}^{n} w_i^2$$

Effects:

- No coefficients become exactly zero (no feature elimination)
- Works well with correlated predictors

Common uses of ridge and lasso

- **Lasso**:
    - Feature selection
    - Some predictors have a way bigger impact than others

Common uses of ridge and lasso

- **Lasso**:
  - Feature selection
  - Some predictors have a way bigger impact than others
- **Ridge**:
  - Address multicollinearity
  - You do not want to **zero out** any predictor

Common uses of ridge and lasso

- **Lasso**:
  - Feature selection
  - Some predictors have a way bigger impact than others
- **Ridge**:
  - Address multicollinearity
  - You do not want to **zero out** any predictor

Caveats:

- Those are not the only existing regularization techniques, but the only ones we will hand here
- This is an **experimental** process, so these regularization techniques might not always be beneficial
- Always prioritize **interpretability** of your model.

The Big Picture of this course

How does everything fit together?

- This course mixed a **statistical approach** to **lab sessions**
- Here, we are introducing **ML concepts** to ease the shift from statistics to ML

- **Exploratory data analysis** has a descriptive intent, and is often the first step to assess **which model to use** and switch to a **predictive setting**
- **Linear regression and classification** are the basis of ML, the simplest models, and essential to understanding **more complex models**
- **Time series** are a more complex statistical topic; they fit under ML if regarded in their predictive aspects, but beware of the added complexity of time measurement
  - Example: if **splitting data**, the different time of the data must be taken into account
- **Causal Inference** expands traditional ML to include **causality** - but we saw its strong relation to **linear regression**

Questions?

Final project guidelines

- The code needs to be **on GitHub** (you can create a private repository and add me as a contributor, meaning **I will receive an email giving me access to your code**)
- Do this at the end, when you finished the modifications
- The paper needs to **contain the reference to the repository**. You will **upload it on Moodle** (only one upload per group!)

Final project guidelines

- **Requirement: predictive step**
- While a focus on EDA and graph-based representations is perfectly fine, the project **must contain at least one of the models we studied**

Deadline on **December 20**!