Machine Learning with PyTorch and Scikit-Learn
From Coursera

Cristo Daniel Alvarado

October 24, 2025

Contents

Code List

Chapter 1

Module 1

The goal of this chapter is to explore the foundational concepts of machine learning, focusing on how algorithms can transform data into knowledge. We delve into the practical applications of supervised and unsupervised learning, equipping you with the skills to implement these techniques using Python tools for effective data analysis and prediction.

Observation 1.0.1 (**Learning Objectives**)
Learning Objectives:

- Analyze data patterns to make future predictions.
- Design systems for supervised and unsupervised learning.
- Implement machine learning algorithms using Python tools.

1.1 Overview

The goal basically is to learn machine learning using Python libraries such as Scikit-Learn and PyTorch.

1.2 Giving computers the ability to learn from data

So, basically in this scenario we will do the following:

- Implement machine learning algorithms using Python tools.
- Desing systems for supervised and unsupervised learning.
- Analyze data patterns to make future predictions.

1.3 Introduction

Definition 1.3.1 (**Machine Learning**)
Machine learning is the *application and science of algorithms that make sense of data.*

The goal of this section is to introduce some of the main concepts and different types of machine learning, together with a basic introduction to the relevant terminology. The goal is to stablish the groundwork for successfully using machine learning techniques for practical prolem solving.

1.3.1 The Three Different Types of Machine Learning

In this age, we have a large amount of structured and non-structured data.

Definition 1.3.2 (**Structured and Non-structured Data**)
Structured data refers *to information that is organized in a predefined format*, typically within a fixed schema, such as rows and columns in a database or spreadsheet

In contrast, **non-structured data** *lacks a predefined format or structure and exists in its native, raw state.* It is typically qualitative and encompasses a wide variety of formats such as text documents, emails, social media posts, images, videos, and audio files.

One of the main goals of machine learning is to extract knowledge from data in order to make predictions.

Observation 1.3.1 (**Use of Machine Learning**)
Machine Learning *offers a more efficient alternative for capturing the knowledge in data to improve the performance of predictive models and make data-driven decisions.*

Idea 1.3.1 (**Note on the use of Machine Learning and some of its Applications**)
Also, notable progress has been made in medical applications; for example, researchers demonstrated that deep learning models can detect skin cancer with near-human accuracy link to article.

Another milestone was recently achieved by researchers at DeepMind, who used deep learning to predict 3D protein structures, outperforming physics-based approaches by a substantial margin link to article.

There are three types of machine learning: **supervised learning**, **unsupervised learning**, and **reinforcement learning**. There are fundamental differences between these three types of machine learning and we will look at each of them in detail with some notes on its possible applications.

Three Types of Machine Learning	
Supervised learning	• Labeled data • Direct feedback • Predict outcome/future
Unsupervised learning	• No labels/targets • No feedback • Find hidden structure in data
Reinforcement learning	• Decision process • Reward system • Learn series of actions

Table 1.1: Three Types of Machine Learning.

1.3.2 Supervised Learning

The main goal of **supervised learning** is to *learn a model from labeled training data that allows us to make predicitons about unseen or future data.*

The term *supervised* refers to a set of training examples (data inputs), where the desired output signals (labels) are already known.

Definition 1.3.3 (**Supervised Learning**)
Supervised learning is the *process of modeling the relationship between data inputs and the labels.*

We can think of supervised learning as *label learning.*

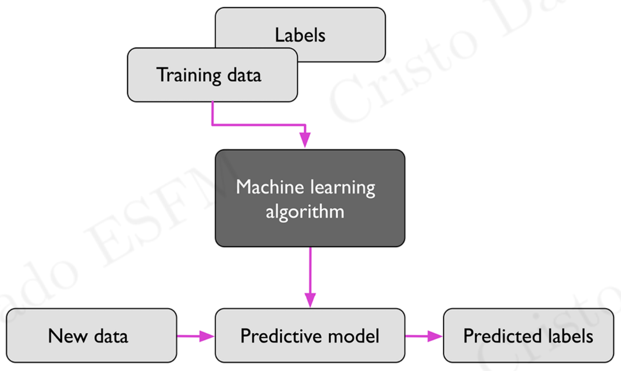


Figure 1.1: General Process of Supervised Learning

Example 1.3.1 (**Example of Supervised Learning**)
Lets consider the example of email spam filtering. We can train a model using a supervised machine learning algorithm on a corpus of labeled emails, which are correctly marked as spam or non-spam.

To predict whether a new email belongs to either of the two categories, a supervised learning task with discrete class labels, such as in the previous email spam filtering example.

This is called a **classification task**. Another subcategory of supervised learning is **regression**, where the outcome signal is a continous value.

1.3.3 Classification for Predicting Class Labels

Definition 1.3.4 (**Classification**)
Classification is *a subcategory of supervised learning, where the goal is to predict the categorical class labels of new instances or data points based on past observations.*

Thos class labels are discrete, unordered values that can be understood as the group memberships of the data points. The previously mentioned example of email spam detection represents a typical example of a binary classification task, where the machine learning algorithm learns a set of rules to distinguish between two possible classes: spam and no-spam emails.

The next ilustration shows the concept of binary classification task given 30 training examples; 15 training are labeled as **class A** and the other 15 labeled as **class B**.

Our dataset is two-dimensional, meaning that each example has two values associated with it: x_1 and x_2. We can use a supervised machine learning algorithm to to learn a rule (decision boundary represented as the dashed line) that can separate two classes and classify new data into each of those two categories given its x_1 and x_2 values.

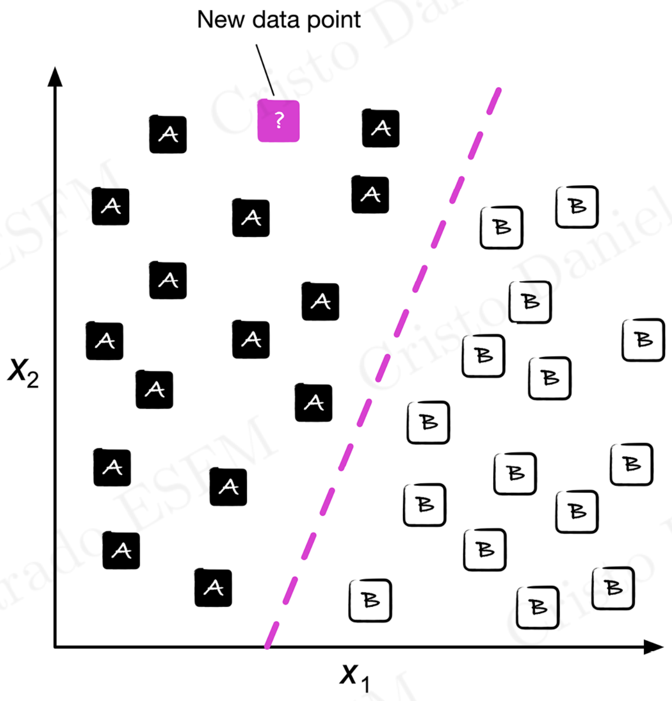


Figure 1.2: Classification in Machine Learning

Observation 1.3.2 (**Nature of classification of class labels**)
When a set of class labels does not have to be of a binar ynature. The predictive model learned by a supervised learning algorithm can assign any class label that was presented in the training dataset to a new, unlabeled data point or instance.

Example 1.3.2
An example of a multiclass classification task is handwritten character recognition.

1.3.4 Regression for Predicting Continous Outcomes

Definition 1.3.5 (**Regression Analysis**)
In a **regression analysis**, *we are given a number of predictor (**explanatory**) variables and a continus response variable (**outcome**) and we try to finde a relationship between those variables that allows us to predict an outcome.*

Observation 1.3.3 (**Feature and Target Variables**)
In the field of machine learning, the predictor variables are commonly called *features*, and the response variables are referred to as *target variables*.

Example 1.3.3
Let's asume we want to predict the mat SAT scores of students. If there is a relationship between time spent studying for the test and the final scores, we could use it as training data to learn a model that uses the study time to predict the test scores of future students.

Given a feature variable x and a target variable y, we fit a straight line to the data points that minimizes the error in predicting y from x.

We can now use the intercept and slope learned from this data to predict the target variable of new data.

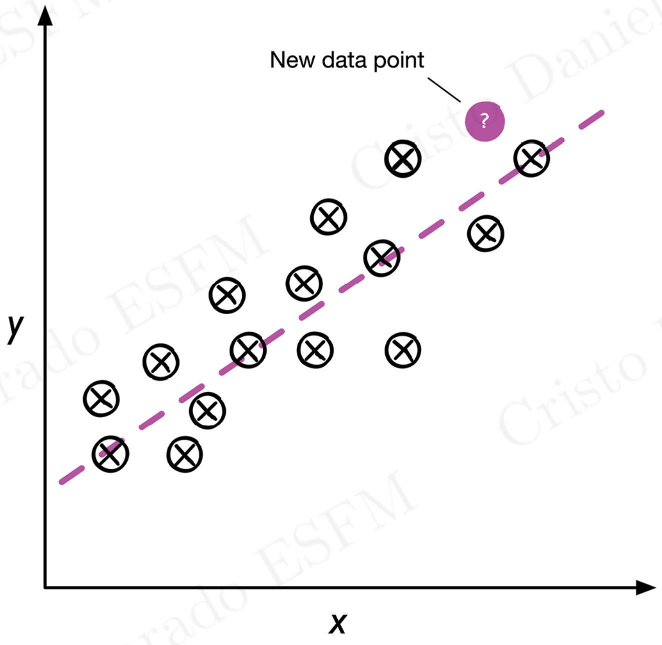


Figure 1.3: Linear Aproximation

1.4 Solving Problems with Reinforcement Learning

Definition 1.4.1 (Reinforcement Learning)
Reinforcement Learning has the *goal to develop a system* (called **agent**) *that improves its performance through interactions with the enviroment.*

Observation 1.4.1
Typically, the information coming from a current state of the enviroment also includes a **reward signal**. With this in mind, reinforcement learning can be vewed as being realted to supervised learning.

This feedback is not the correct ground-truth label or value, *but a measure of how well the action is evaluated by a reward function.*

Using a reward signal, a system can measure this reward via an exploratory trial-and-error approach or deliberative planning.

Example 1.4.1
One of the most popular examples of reinforcement learning is **chess program**. The agent decides upon a series of moves and the reward can be defined as win or lose at the end of the game.

In synthesis, the reinforcement learning can be described with the following diagram.

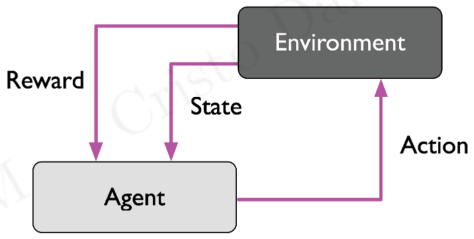


Figure 1.4: Reinforcement Learning Process

There are many different subtypes of reinforcement learning, but in a general scheme the agent tries to maximize the reward through a series of interactions with the enviroment.

Idea 1.4.1
In summary, reinforcement learning is concerned with learning to choose a series of actions that maximizes the total reward which could be earned immediately after taking an action or vía *delayed* feedback.

1.5 Unsupervised Learning

Unsupervised learning tries to find the hidden structures of our data. Using supervised learning or reinforcement learning, we have the right answer or a measure of a reward for particular actions, respectively. Unsupervised learning explores the structure of our data to extract meaningful information without the guidance of a known outcome variable.

1.5.1 Finding Subgroups With Clustering

Definition 1.5.1 (Clustering)
Clustering *is an exploratory data analysis or pattern discovery technique that organizes information into meaningful subgroups* (called **clusters**) *without any prior knowledge of group memberships.*

Each cluster defines a group of objects that share a certain degree of similarity but are more dissimilar to objects in other clusters which is why clustering is called (sometimes) **unsupervised classification**.

The following scatter plot shows how clustering can organizse unlabeled data into three disctinct groups or clusters.

1.5.2 Dimensionality Reduction for Data Compresion

Another subfield of unsupervised learning is dimensionality reduction.

Definition 1.5.2 (Dimensionality Reduction)
Dimensionality reduction is the process to reduce the dimension of the data we are working it, preserving the original and meaningful data needed to interpret it.

Often, we work with data of high dimensionality that can present challenges for limited storage space and the computational performance of machine learning algorithms. The unsupervised deimensionality reduction is commonly used in feature preprocessing to remove noise from data, which can degrade the predictive performance of certain algorithms.

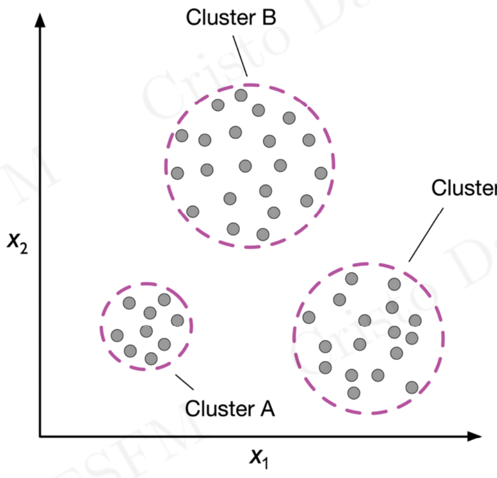


Figure 1.5: Caption

Dimensionality reduction compresses the data into a smaller dinensional subspace while retaining most of the relevant information.

Observation 1.5.1
Sometimes, dimensionality reduction is useful for visualizing data. The following figure is a good and useful example of it.

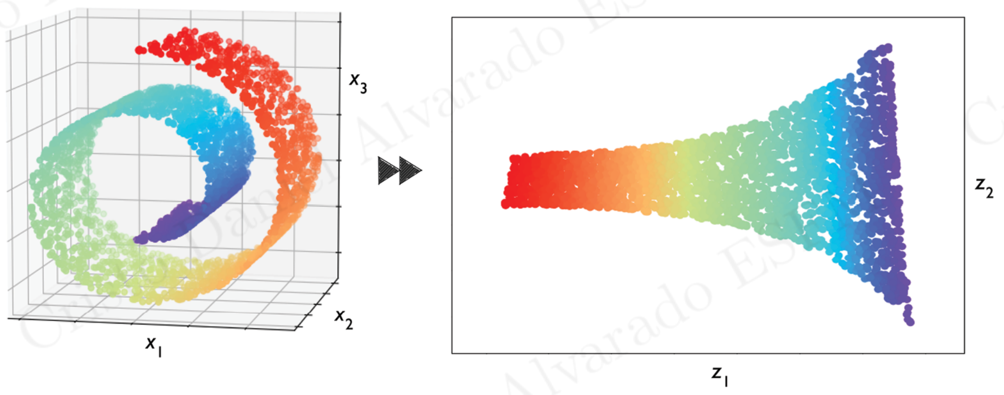
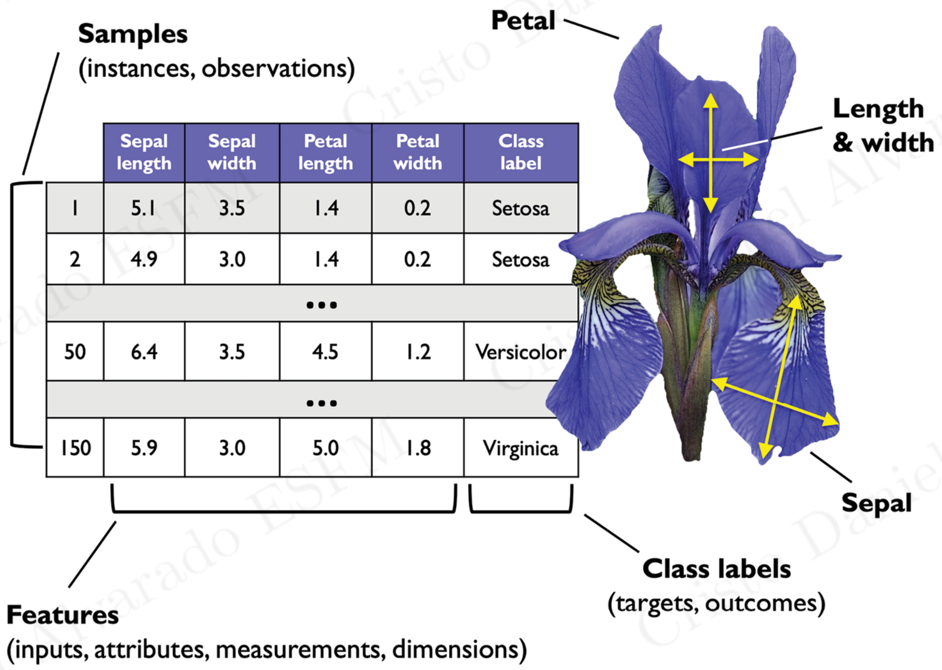


Figure 1.6: Example of Dimensionality Reduction in a Dataset.

1.6 Terminology and Notations

Definition 1.6.1 (**Dataset**)
A **dataset** is a *collection of related sets of information that is composed of separate elements* but can be manipulated as a unit by a computer.

One image that contains most of the information that we'll be using throught this course is the following:


Samples
(instances, observations)
Petal
Length & width
Sepal
Class labels
(targets, outcomes)
Features
(inputs, attributes, measurements, dimensions)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
•••					
50	6.4	3.5	4.5	1.2	Versicolor
•••					
150	5.9	3.0	5.0	1.8	Virginica

Figure 1.7: Samples, Features, and Class labels.

Example 1.6.1
The Iris dataset contains the measurements of 150 iris flowers from three different species (setosa, versicolor and virginica).

Each flower example represents one row in our dataset and the flower measurements in centimeters are stored as columns, called **features of the dataset**.

Definition 1.6.2 (**Feature**)
In a dataset, a **feature** *is an input, attribute, measurement or dimensions of something.*

Idea 1.6.1 (**Notation**)
The iris dataset consists of 150 examples and four features, we will write them as a 150×4 matrix:

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

Vectors will be identified as the column matrix $\vec{x} \in \mathbb{R}^{n\times 1}$. To refer to the elements of the matrix we will use the notation $x_i^{(j)}$.

Observation 1.6.1 (**Use of Notation**)
In the latter idea, $x_1^{(150)}$ refers to the first dimension flower example 150, corresponding to the sepal length.

Each row in the matrix X represents one flower instance and can be written as a four-dimensional row vector:

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

Each feature is a 150-dimensional column vector, denoted by x_j:

$$x_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

We represent the target variables (here, class labels) as 150-dimensional column vector:

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(150)} \end{bmatrix}$$

where $y^{(i)} \in \{\text{Setosa}, \text{Versicolor}, \text{Virginica}\}$.

1.6.1 Machine Learning Teminology

Definition 1.6.3 (Terminology in Machine Learning)
Some of the following terms are largely used in machine learning:

- **Training example**: A row in a table representing the dataset and synonymous with an observation, record, instance, or sample (in most contexts, sample refers to a collection of training examples).
- **Training**: Model fitting, for parametric models similar to parameter estimation.
- **Feature** (x): A column in a data table or data (design) matrix. Synonymous with predictor, variable, input, attribute, or covariate.
- **Target** (y): Synonymous with outcome, output, response variable, dependent variable, (class) label, and ground truth.
- **Loss function**: Often used synonymously with a cost function. Sometimes the loss function is also called an error function. In some literature, the term 'loss' refers to the loss measured for a single data point, and the cost is a measurement that computes the loss (average or summed) over the entire dataset.

1.7 Machine Learning Systems

In this section we'll dsicuss the other important parts of a machine learning system accompaying the learning algorithm.

1.7.1 Preprocessing: Getting Data into Shape

Raw data rarely comes in the form and shape that is necessary for the optimal performance of a learning algorithm. Thus, the preprocessing of the data is one of the most crucial steps in any machine learning application.

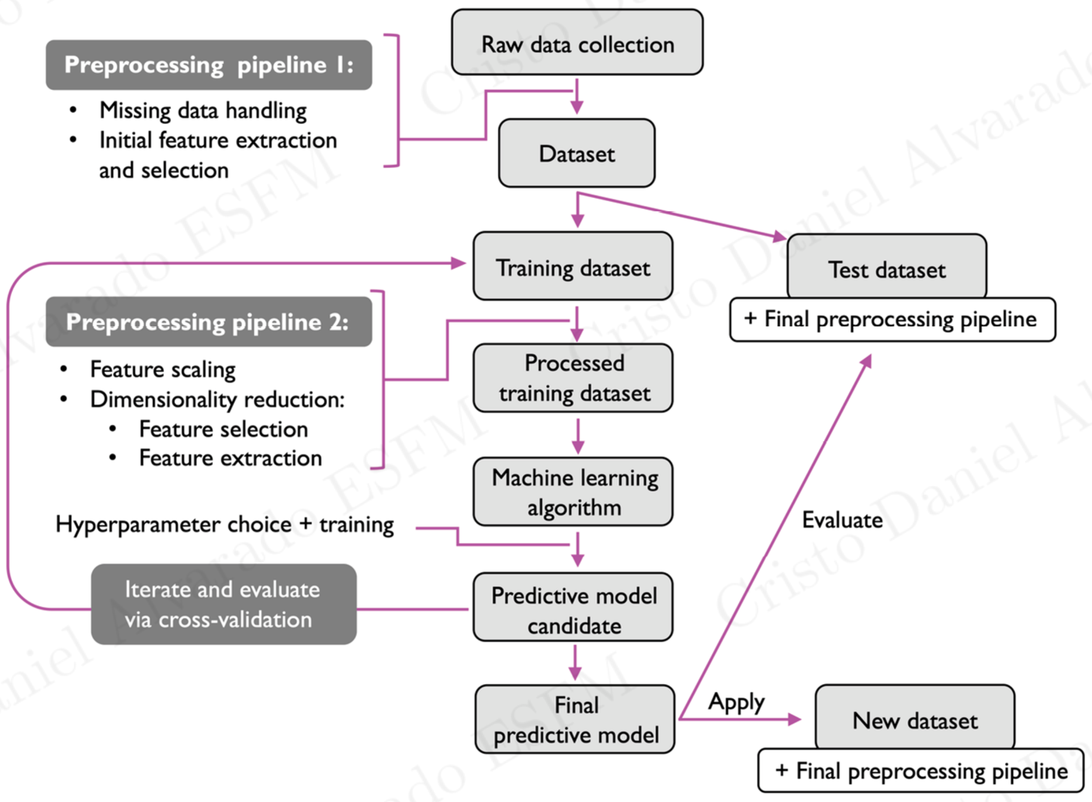

Raw data collection
Preprocessing pipeline 1:
Missing data handling
Initial feature extraction and selection
Dataset
Training dataset
Test dataset
+ Final preprocessing pipeline
Preprocessing pipeline 2:
Feature scaling
Dimensionality reduction:
Feature selection
Feature extraction
Processed training dataset
Machine learning algorithm
Hyperparameter choice + training
Evaluate
Iterate and evaluate via cross-validation
Predictive model candidate
Final predictive model
Apply
New dataset
+ Final preprocessing pipeline

Figure 1.8: Machine Learning System.

Definition 1.7.1 (Preprocessing)
Preprocessing is the *process of converting raw data into dara suitable for a machine learning algorithm.*

Example 1.7.1 (Optimal Features in the Iris Dataset)
Let's consider the iris dataset from the previous example. We can think of raw data as a series of flower images from which we want to extract **meaningful features**. Useful features could be:

- Centered around the color of the flowers.
- Height.
- Length.
- Width.

Observation 1.7.1 (Data Convesion)
Many machine learning models require that the selected features are on the same scale for optimal performance, which is achieved by transforming the features to the range $[0, 1]$.

Other way to achieve this is use standard normal distribution with zero mean and unit variance.

Observation 1.7.2 (**Use of Dimensionaly Reduction**)
When certain features are highly correlated and therefore, redundant, dimensionaly techniques may be useful for compressing the features onto a lower dimensional subspace.

The main advantage of dimensionality reduction is that less storage space is required, and therefore the learning algorithm can run much faster. This can also improve the performance of a model if the dataset contains a large number of irrelevant features (or noise).

Definition 1.7.2 (**Noise**)
In data science and statistics, **noise** refers to the *random, uncontrollable, and unexplained variations in the data. It is the part of the data that does not represent the underlying phenomenon you are trying to study or model.*

Observation 1.7.3 (**How to Deal with Noise?**)
You can rarely eliminate noise completely, but you can manage it.

(1) **Data Cleaning & Preprocessing**: This is the first line of defense.

- **Smoothing**: Applying algorithms to average out random fluctuations in time-series or signal data.
- **Binning**: Grouping numerical values into bins to smooth out small irregularities.
- **Correcting Typos**: Standardizing text entries.

(2) **Collecting More Data**: A larger dataset can sometimes "drown out" the noise, as the true signal becomes stronger with more examples.

(3) **Using Robust Algorithms**: Some machine learning models (like Random Forests) are inherently more resistant to noise than others (like Decision Trees).

(4) **Feature Selection**: Removing irrelevant or redundant features that mostly contain noise.

Idea 1.7.1
To determine whether our machine learning algorithm performs well not only on the training dataset but also generalizes well to new data, *we also want to randomly divide the dataset into separate training and test datasets.* In this way, we can measure whether our machine learning model is performing well.

1.7.2 Training and Selecting a Predictive Model

We cannot get learning for free.

Observation 1.7.4
Meany different machine learning models have been developed to solve different problem tasks.

The goal is that sometimes is easier to use a certain machine learning model for some tasks than use it for other ones.

Example 1.7.2
Each clasification algorithm has its inherent biases, and no single calssification model enjoys superiority if we do not make any assumptions about the task.

In summary, it's essencial to compare at least a handful of different learning algotrithms in order to train and select the best performing model.

Observation 1.7.5
Before deciding which models we shall use, we have to decide upon a metric to measure performance. One commonly used metric is classification accuracy, which is defined as the proportion of correctly classified instances.

Sometimes we may have a machine learning model which was trained using a dataset for model selection, but one question arises: what if we do not use this training dataset? To adress this issue, we can use different techniques such as cross-validation.

Definition 1.7.3 (**Cross-Validation**)
Cross-validation *we devide further a dataset into training and validation subsets in order to estimate the generalization performance of the model.*

Observation 1.7.6 (**Hyperparameters**)
We also cannot expect that the default parameters of the different learning algorithms provided by software libraries are optimal for our specific problem task. Therefore, we will make frequent use of hyperparameter optimization techniques that help us fine-tune the performance of our model in later sections.

We can think of those hyperparameters as parameters that are not learned from the data but represent the knobs of a model that we can turn to improve its performance.

1.7.3 Evaluating Models and Predicting Unseen Data Instances

After selecting a model that has been fitted on the training dataset, we can use the test dataset to estimate how well it performs on unseen data to determine the **generalization error**.

Definition 1.7.4 (**Generalization Error**)
Generalization Error is *formally the expected error of the model on a new, random instance drawn from the same underlying data distribution.*

Observation 1.7.7 (**Dataset Dependancy**)
It is important to note that the parameters for previously mentioned procedures, such as feature scaling and dimensionality reduction, **are solely obtained from the training dataset**, and the **same parameters are later reapplied to transform the test dataset as well as any new data instances—the performance measured on the test data may be overly optimistic otherwise**.

Chapter 2

Training Simple Machine Learning Algorithms for Classification

Appendix A

Using Python for Machine Learning

A.1 Basic Configuration

For machine learning tasks, we will mostly refer to the *Scikit-Learn* library, which is one of the most popular and accessible open-source machine learning libraries for python.

When we focus on a subfield of machine learning called: **deep learning**, we wiell use the latest version of the *PyTorch library*, which specializes in the training of the so called deep **netural network models**.

Definition A.1.1 (Scikit-learn)
Scikit-learn is a *classical machine learning library for Python. It is built on top of NumPy and SciPy and provides a clean, uniform, and simple API for a wide variety of traditional ML algorithms.*

PyTorch is an *open-source deep learning framework developed primarily by Facebook's AI Research lab (FAIR). It provides the foundational building blocks for building and training neural networks, with a strong focus on flexibility and speed.*

We'll use version 3.9 of Python. To check the version of python we use the following code:

```
python --version
```

Code A.1: Check Python Version

Observation A.1.1 (Use of pip)
To install aditional packages used thorught the course, we can install them via the `pip` program.

Definition A.1.2 (Pip)
`pip` is the *standard package installer for Python.* It is a *command-line utility that allows users to install, manage, and uninstall Python packages and libraries that are not part of the Python standard library.*

After installing python, it's possible to execute the following comand in order to install some package:

```
        pip install SomePackage
```

Code A.2: `pip` Package Installation.

A.2 Anaconda Python Distribution and Package Manager

Idea A.2.1
A recommended open-source package management system for installing python for scientific computing contexts is conda by Continuum Analytics.

Definition A.2.1 (Conda)
Conda is a *free and licensed under a permissive open-source licence.* Its goal is to help with the *installation and version management of Python packages for data science, math and engineering across different operating systems.*

Conda comes in different flavours, like a Linux installation. Some of them are **Anaconda**, **Miniconda** and **Minforge**.

- **Anaconda** comes with many scientific computing packages pre-installed. The Anaconda installer can be downloaded here and an Anaconda quick start guide is available here.
- **Miniconda** is a leaner alternative to Anaconda (here). Essentially, it is similar to Anaconda but without any packages pre-installed, which many people (including the authors) prefer.
- **Miniforge** is similar to Miniconda but community-maintained and uses a different package repository (conda-forge) from Miniconda and Anaconda. We found that Miniforge is a great alternative to Miniconda. Download and installation instructions can be found in the GitHub repository here.

After successfully installing conda through either Anaconda, Miniconda, or Miniforge, we can install and update, respectively, new Python packages using the following command:

```
conda install SomePackage
conda update SomePackage
```

Code A.3: Conda Package Installation.

Packages that are not available through the official conda channel might be available via the community-supported conda-forge project (conda-forge), which can be specified via the –channel conda-forge flag. For example:

```
conda install SomePackage --channel conda-forge
```

Code A.4: caption

Packages not available throught the default conda channel or conda-forge can be installed via pip as explainded earlier in Code A.2.

A.3 Packages for Scientific Computing, Data Science, and Machine Learning

We will use mainly NumPy's arrays and sometimes Pandas (highler level data manipulation tools). Also the Matplotlib library will be useful to visualize quantitative data. The version of the packages to install are the following:

- NumPy 1.21.2
- SciPy 1.7.0
- Scikit-learn 1.0
- Matplotlib 3.4.3
- Pandas 1.3.2

After installing these packages, you can double-check the installed version by importing the package in Python and accessing its `__version__` attribute:

```
> Use ‘numpy.__version__‘ to check the installed version.
'1.21.2'
```

Code A.5: Check Version of Packages