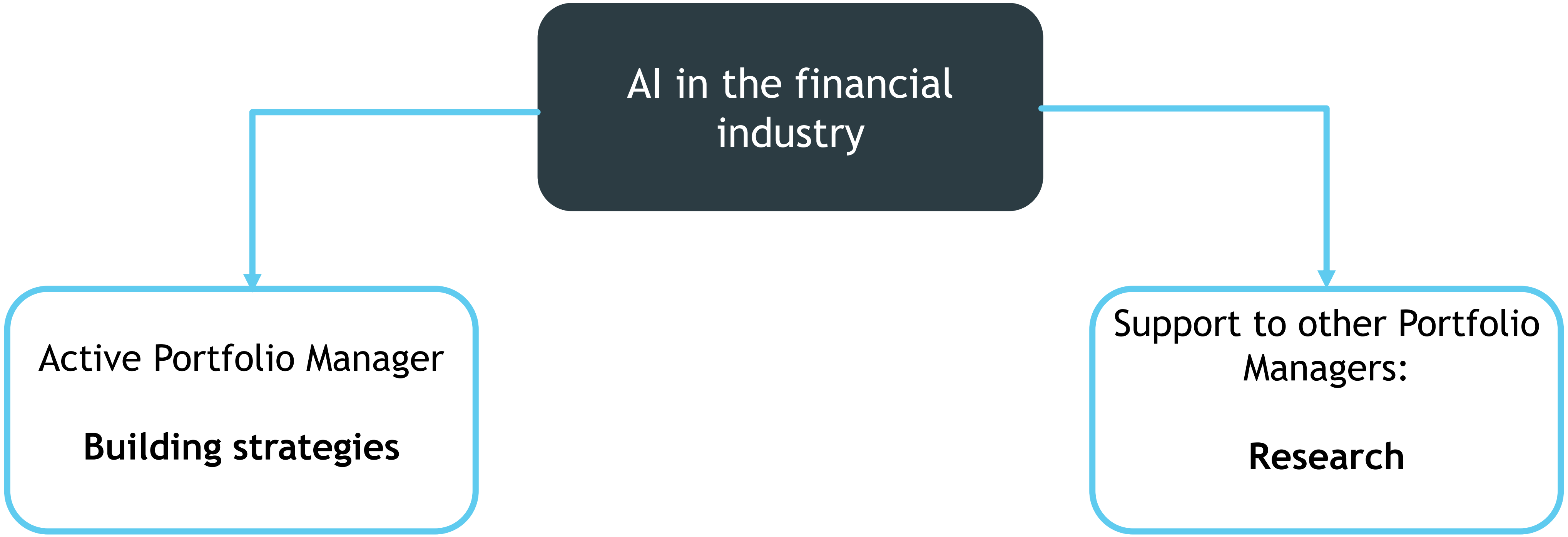
Computational Finance - Lesson 9

29/11/2024

ginevra.angelini@polimi.it

Asset Management & Machine Learning


AI in the financial industry
Active Portfolio Manager
Building strategies
Support to other Portfolio Managers:
Research

Building Strategies

Asset Management & Machine Learning

- In asset management there exist different ways and approaches to build portfolio strategies. They can be divided in two main categories:
    - **Discretional Approach**: the investor elaborates *information* (news, price variables, macroeconomic variables…) in a discretionary way in order to take investment decisions.
    - **Quantitative Approach:** the investor elaborates the information flow through a mathematical model.
        - **Artificial Intelligence:** information are processed by Machine Learning & Artificial Intelligence algorithms.

Why Machine Learning

- In financial markets the dynamic under the evolution of prices is really complex. In general we can state that:

  - **It depends on a high number of factors**
  - **Relations are not linear and sometimes counter-intuitive**
  - **It is really difficult to "theorize" in advance some rules that explain the market**

- Machine Learning, by its capability to model complex and non-linear relationships and of managing a large amount of data, if it is used correctly, it can be able to extract (generalize) some "rules" that can explain the evolution of the market.

Machine Learning

- Machine Learning is the science of programming computers so they can learn from data.

- A computer program is said to learn from experience ***E*** with respect to some task ***T*** and some performance measure ***P***, if its performance on ***T***, as measured by ***P***, improves with experience ***E***. (Tom Mitchell 1997)

- In practice, Machine learning can also be defined as the process of solving a practical problem by

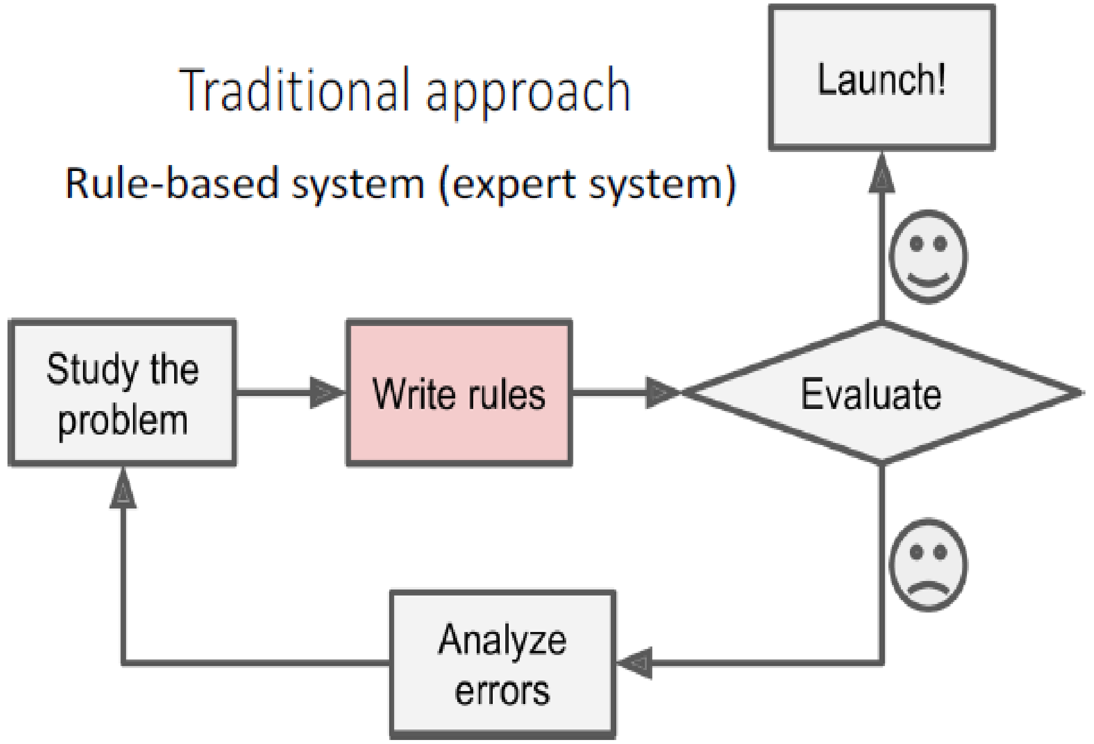
  1) gathering a dataset,
  2) algorithmically building a statistical model based on that dataset.

- In its application across business problems, machine learning is also referred to as predictive analytics.

Machine Learning

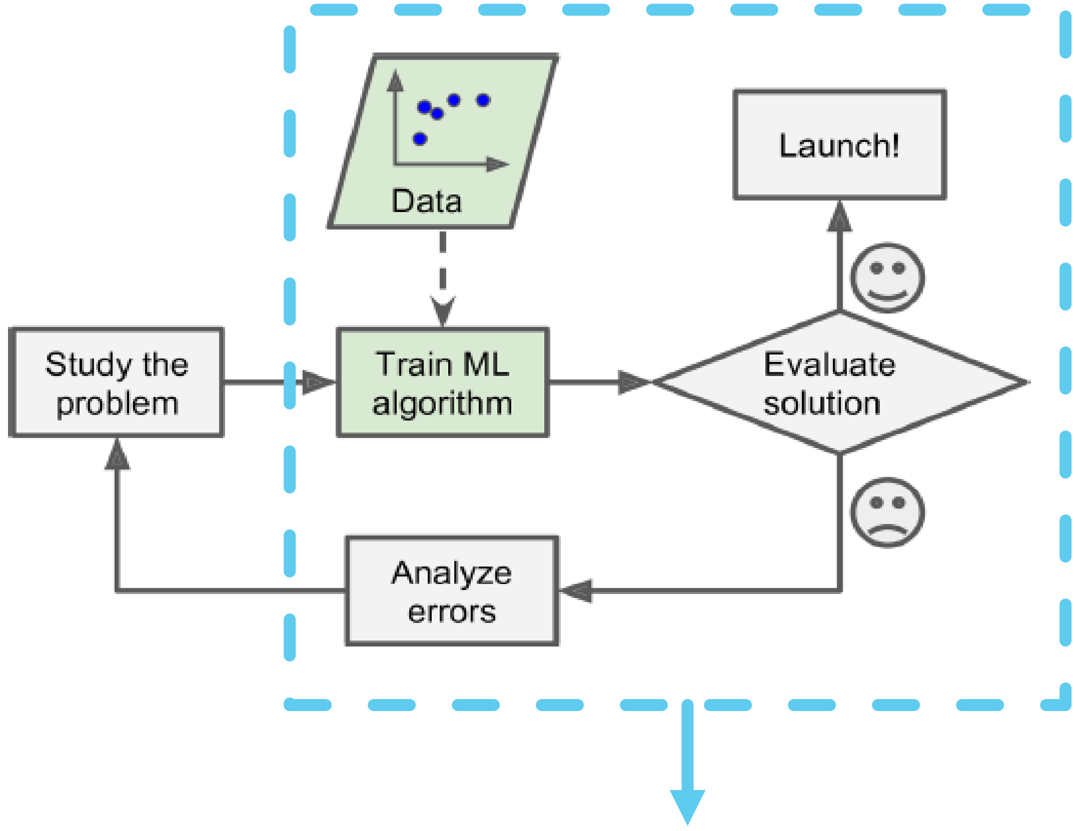
Traditional approach

Rule-based system (expert system)



Machine learning approach

Automatically detect patterns



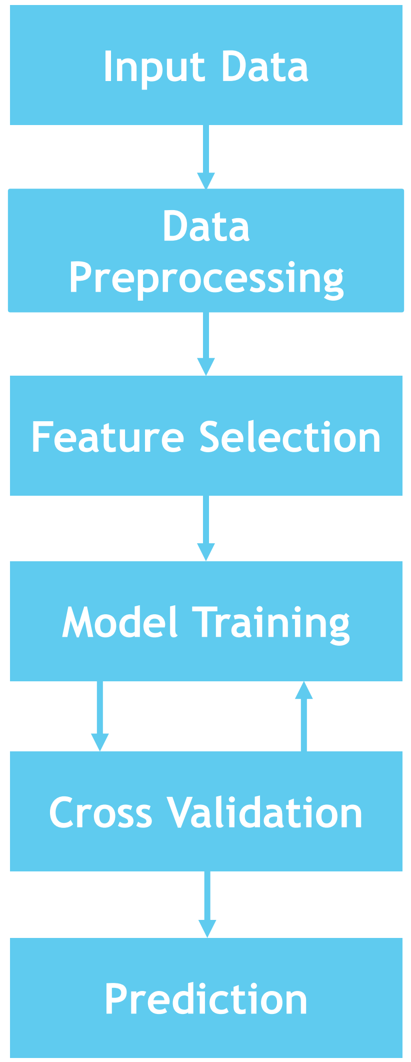
Process that can be updated and automated, adapting to changes in data

Building-Blocks

- The fundamental elements we need in order to apply a machine learning approach are the following:
  - **The problem** → the event of which we want to extrapolate the “rules”
  - **Target variable** →the variable we want to predict (necessary only for a certain class of algorithms)
  - **Input Data** → observations that potentially can “explain” the target variable
  - **Machine Learning model**→ the algorithm used to predict the target variable from the input data

Machine Learning Pipeline

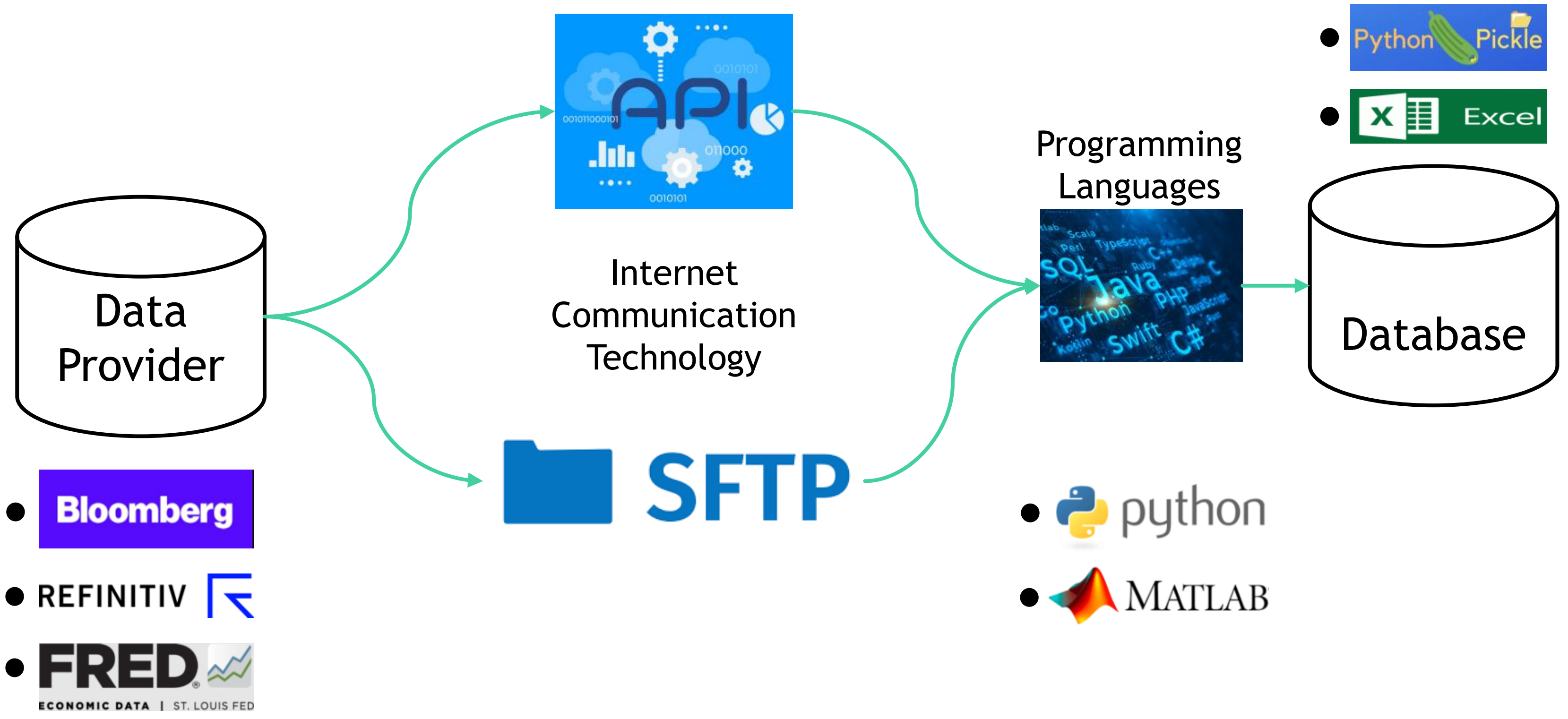
- **Pipeline:** Sequence of operations that must be executed in order to go from raw data to the final output of the model
  - **Input Data - The Data Supply Chain →** provide the supply of data
  - **Data preprocessing →** computing features, dataset splitting , stationarization and normalization
  - **Feature selection →** selection of the input variables that most affect the target variable
  - **Model Training →** estimation of the hyper-parameters of the model
  - **Tuning of hyper-parameters (cross-validation) →** model selection
  - **Test Prediction →** test of the effective accuracy of the model
- **Financial Domain → from prediction to the strategy**

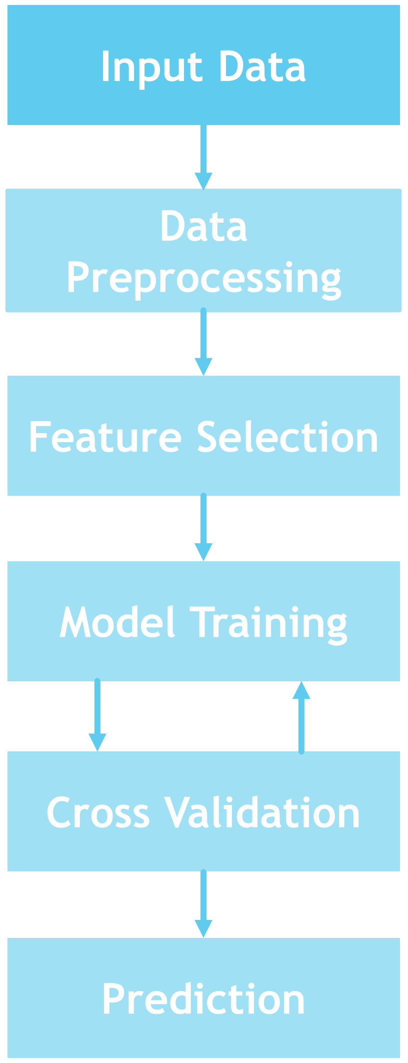


Input Data

- It is possible to use different kind of data: **(i) Macro economic data**; **(ii) Market data** (prices, volatility, …); **(iii) Fundamental data** (Debt, Earning, …); **(iv) Alternative Data** (news, social media data, …)

The Data Supply Chain





Data Preprocessing - Stationarity

- Let X_t be a stochastic process and $F_t^k = (X_t, .., X_{t+k-1})$ be the cumulative distribution function of the unconditional joint distribution of X_t, then the condition such that X_t is ***strictly stationary*** is:

$$F_t^k = F_\tau^k \quad \forall t \neq \tau$$

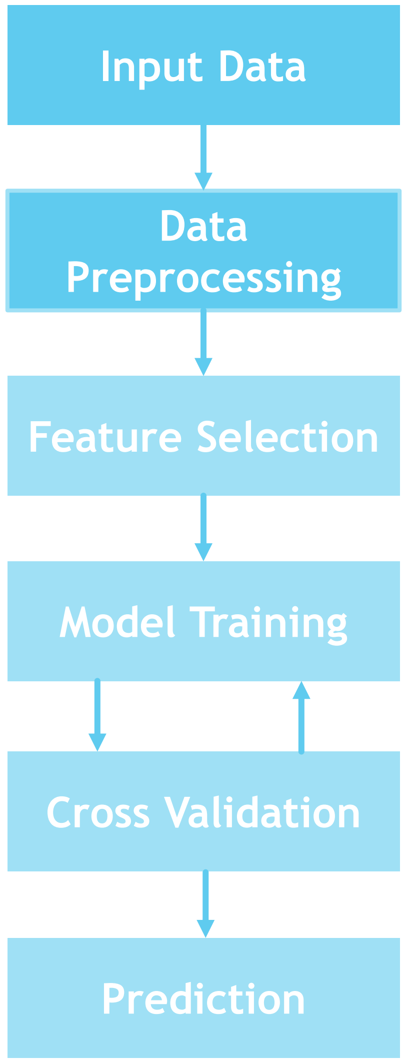
- The stochastic process X_t is said to be **weakly stationary** if satisfies the following conditions:

  i. $E(X_t) = \mu \quad \forall t$
  ii. $Var(X_t) < \infty \quad \forall t$
  iii. $Cov(X_t, X_s) = Cov(X_{t+h}, X_{s+h}) \quad \forall t, s, h$

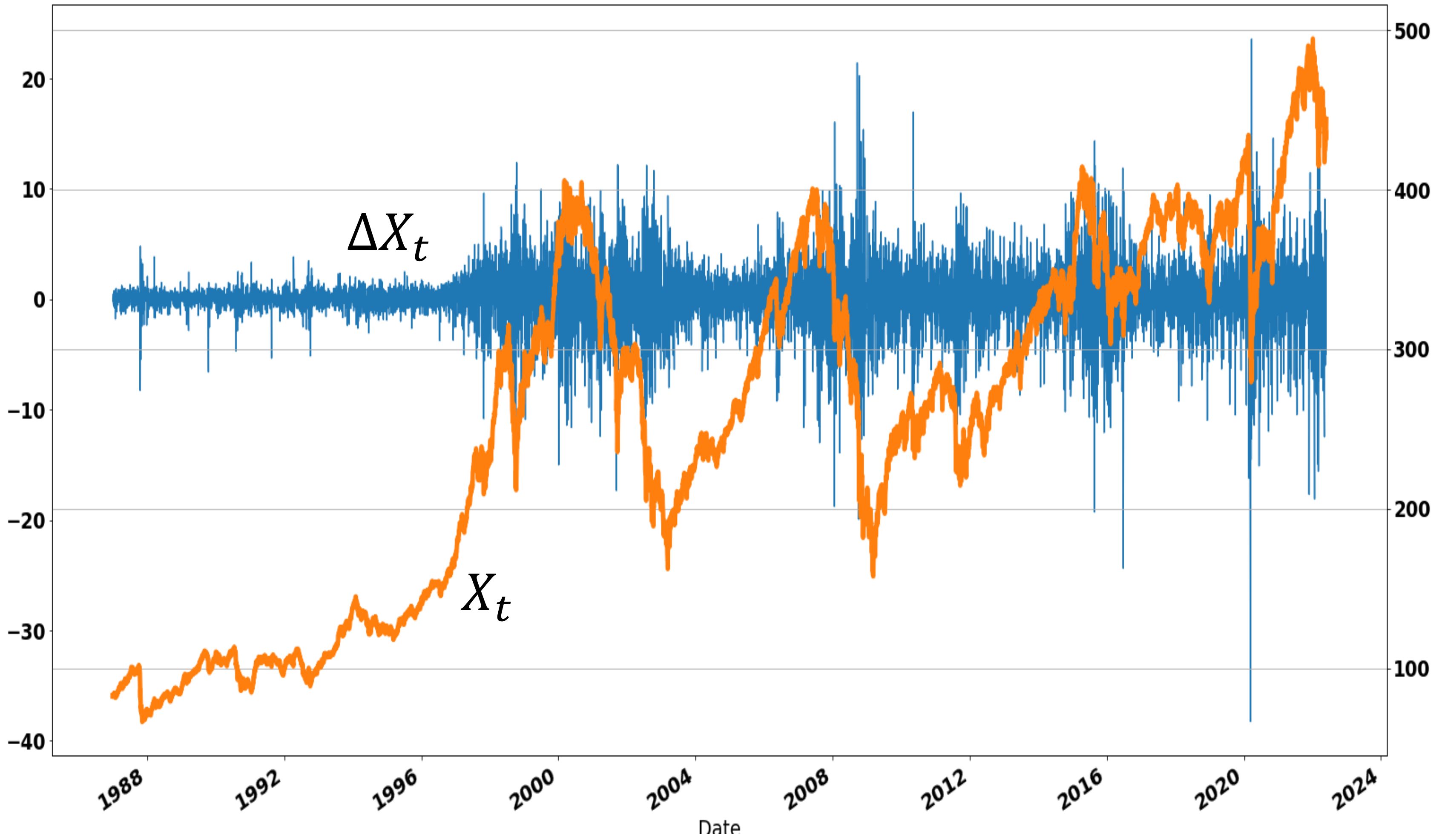
- **Stationarity Test:** Dickey-Fuller Test (ADF), a unit root test

$$\Delta y_t = (\rho - 1) y_{t-1} + \varepsilon_t$$

- H0: $\rho = 1$, not stationary process
- H1: $|\rho| < 1$, stationary process

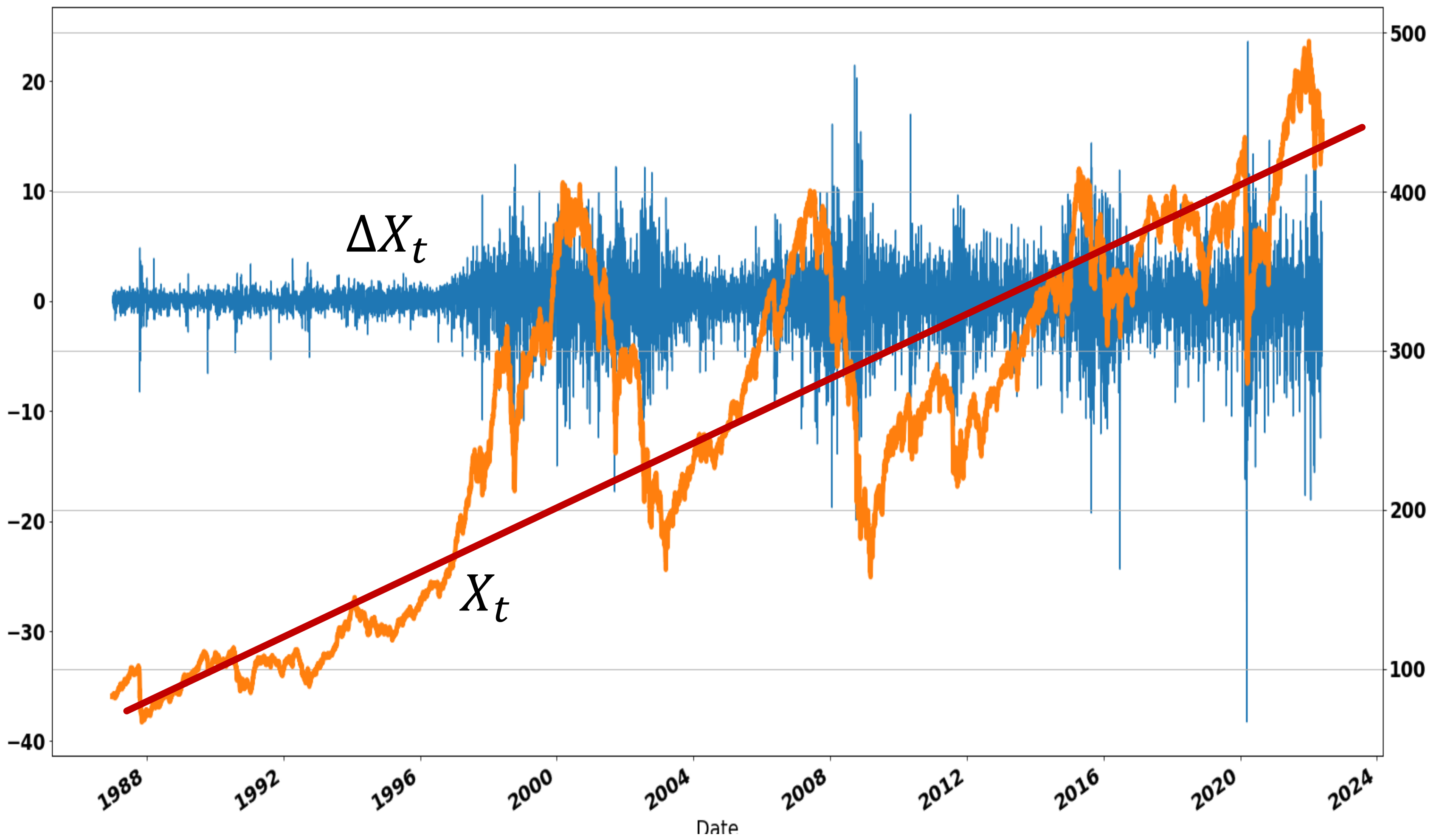


Data Preprocessing - Stationarity



- ❑ X_t : evolution of price of the index STOXX 600
- ❑ ΔX_t: first difference of X_t

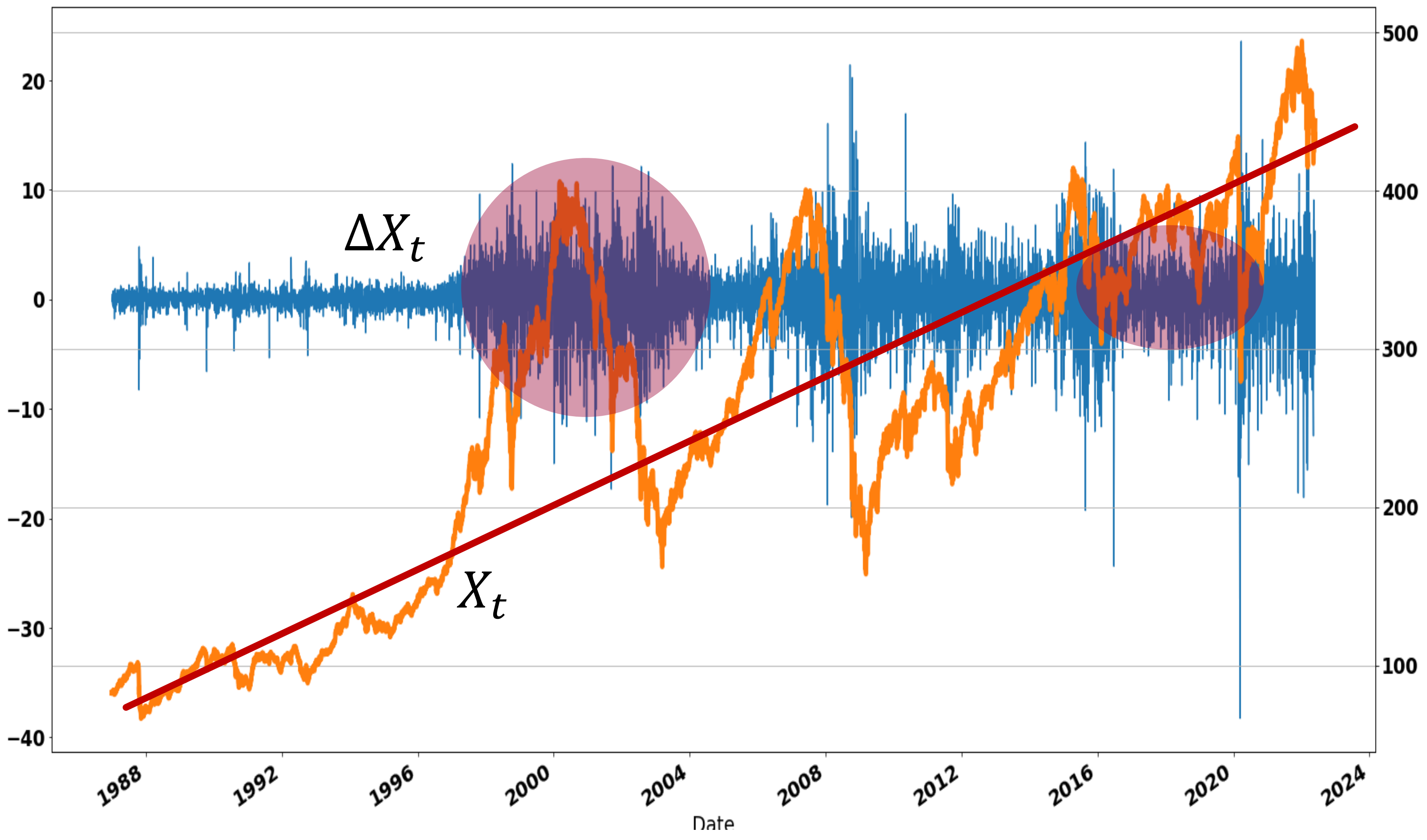
Data Preprocessing - Stationarity



- ❑ X_t : evolution of price of the index STOXX 600
- ❑ ΔX_t: first difference of X_t

- ➢ The price series is not stationary in the mean → Stationarization by differantiation

Data Preprocessing - Stationarity



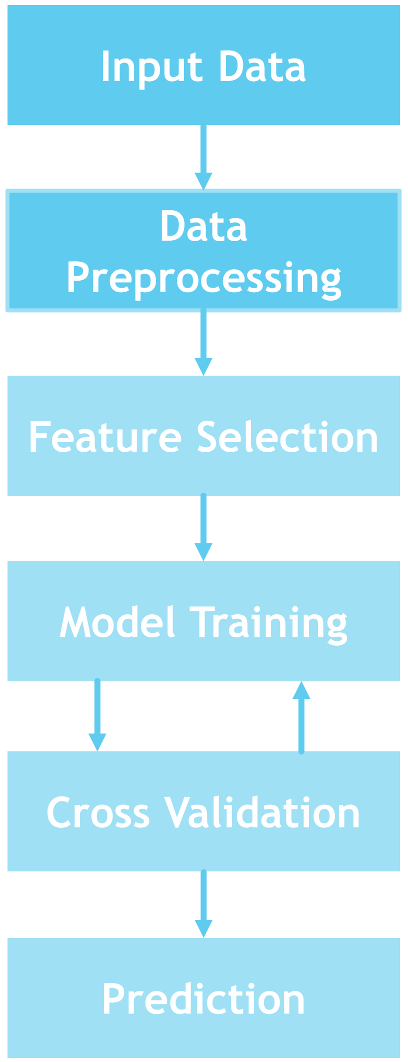
- ❑ X_t : evolution of price of the index STOXX 600
- ❑ ΔX_t: first difference of X_t

- ➢ The price series is not stationary in the mean → Stationarization by differentiation

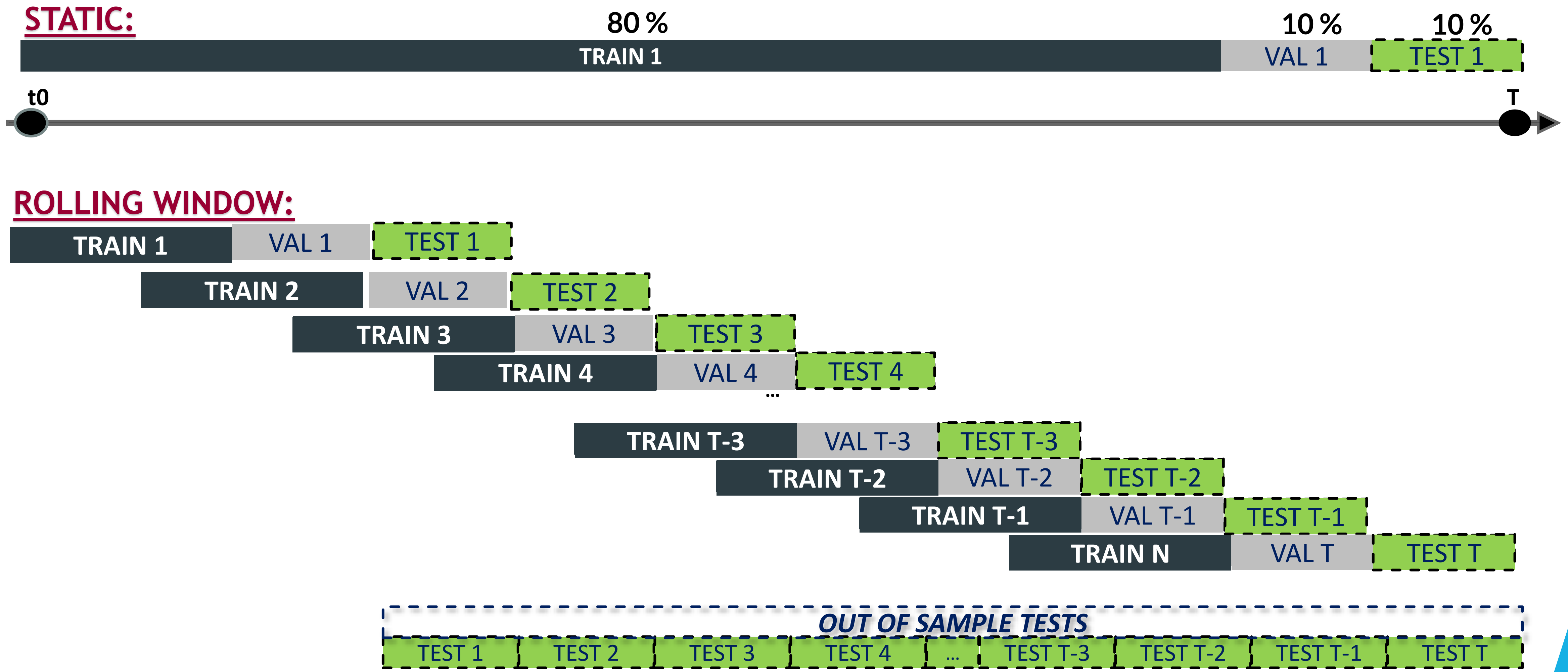
- ➢ **The differentiated series is not stationary in variance (volatility clusters)**

Data Preprocessing – Splitting Dataset

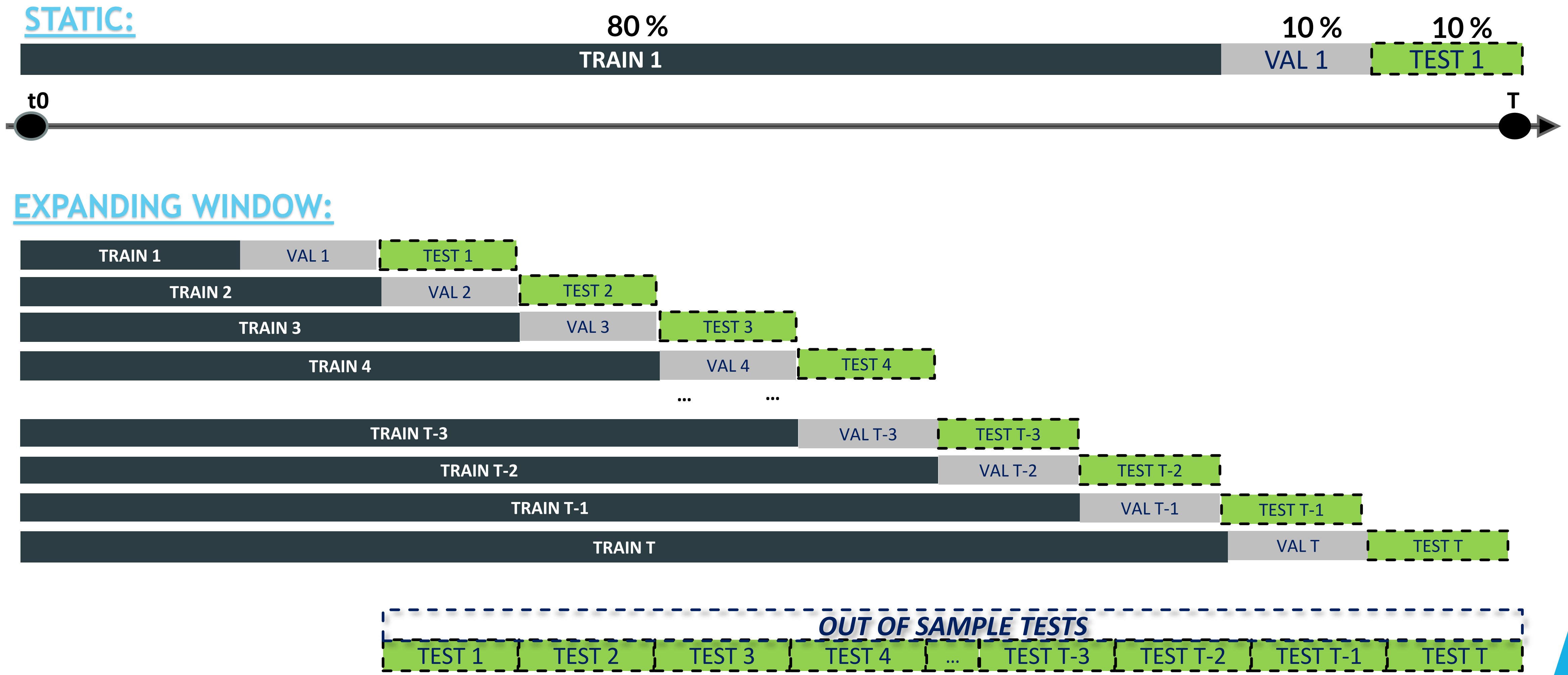
- The best practice for the estimation and testing of a ML model requires the splitting of the dataset in the following subsets:
  - **Training set**, subset of data used for the estimation of the hyperparameters of the model
  - **Validation set**, subset of data used for the selection of the model
  - **Test set**, subset of data used for testing the efficiency of the obtained model
- **Data Leakage**: the intersection of the three subsets must be null



Splitting Dataset: Static vs Rolling Window



Splitting Dataset: Static vs Expanding Window



Feature Selection

- **Feature selection** is the process of *choosing a subset of relevant features* from the original dataset. The goal is to improve the performance of a machine learning model by reducing dimensionality, mitigating the risk of overfitting.

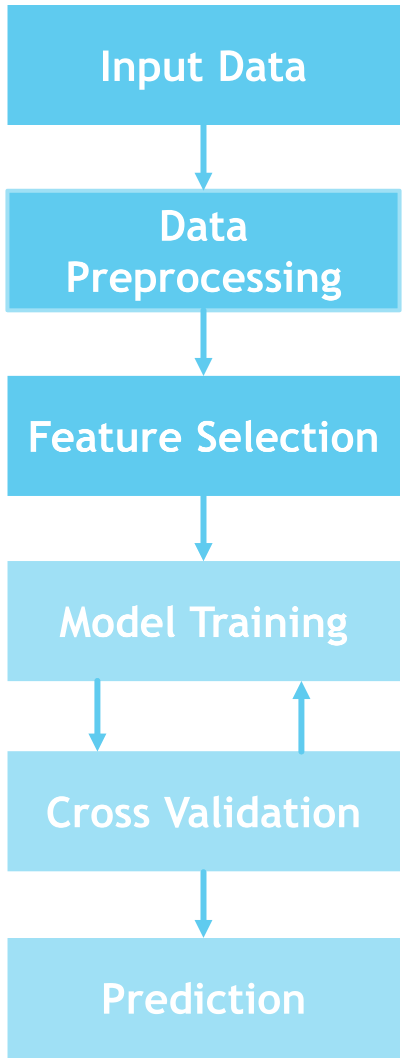
- In feature selection, the aim is to retain the most informative and discriminative features while discarding irrelevant or redundant ones.

- Feature Selection methods can be divided in *two principal categories*:

  - ***Univariate Methods (Filters)***, they evaluate the **relevance of individual features** based on certain criteria, independently of the chosen machine learning algorithm. These methods assess the characteristics of each feature with respect to the target variable and rank or select features accordingly.

    Common criteria include correlation with the target variable, statistical tests, or information-theoretic measures such as **mutual information**.

    - **Advantage:** low computational cost
    - **Drawback:** the interaction between features is not taken into account



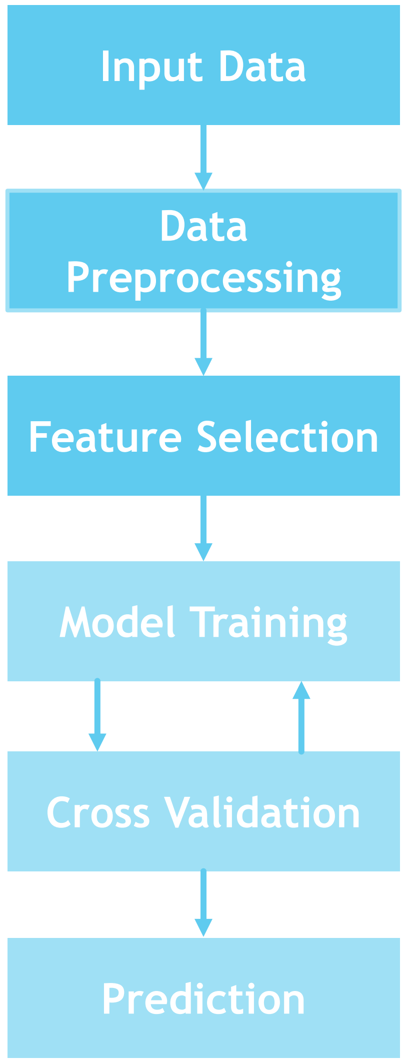
Feature Selection

- ***Multivariate Methods***, they consider the interactions and relationships between multiple features simultaneously, rather than evaluating individual features in isolation.

  These methods take into account the joint effect of features and aim to identify subsets of features that collectively contribute valuable information to the model. Between the most important we can find:

  - **Wrapper Methods :** they generate different subsets of features. This can involve creating combinations of features, adding or removing features iteratively, or using more sophisticated search strategies. (GA; forward selection; backward elimination)

    - **Advantage:** the interaction between features is taken into account
    - **Limitation:** high computational cost



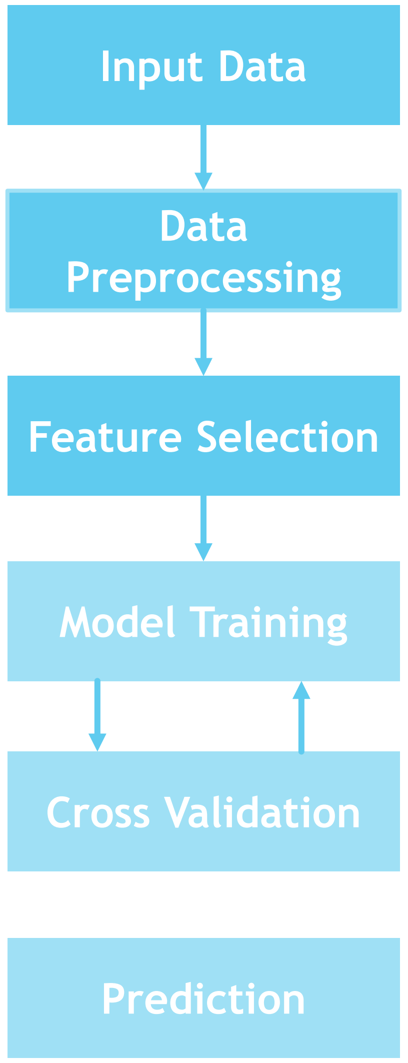
Feature Selection

- **Embedded methods**, they integrate the feature selection process into the model training process. Unlike wrapper methods, which use a separate model, embedded methods perform feature selection as an integral part of the model training itself.

  During the training of a machine learning model, embedded methods evaluate the importance or relevance of features. The model is trained to learn the relationships between features and the target variable while simultaneously assigning importance scores to each feature

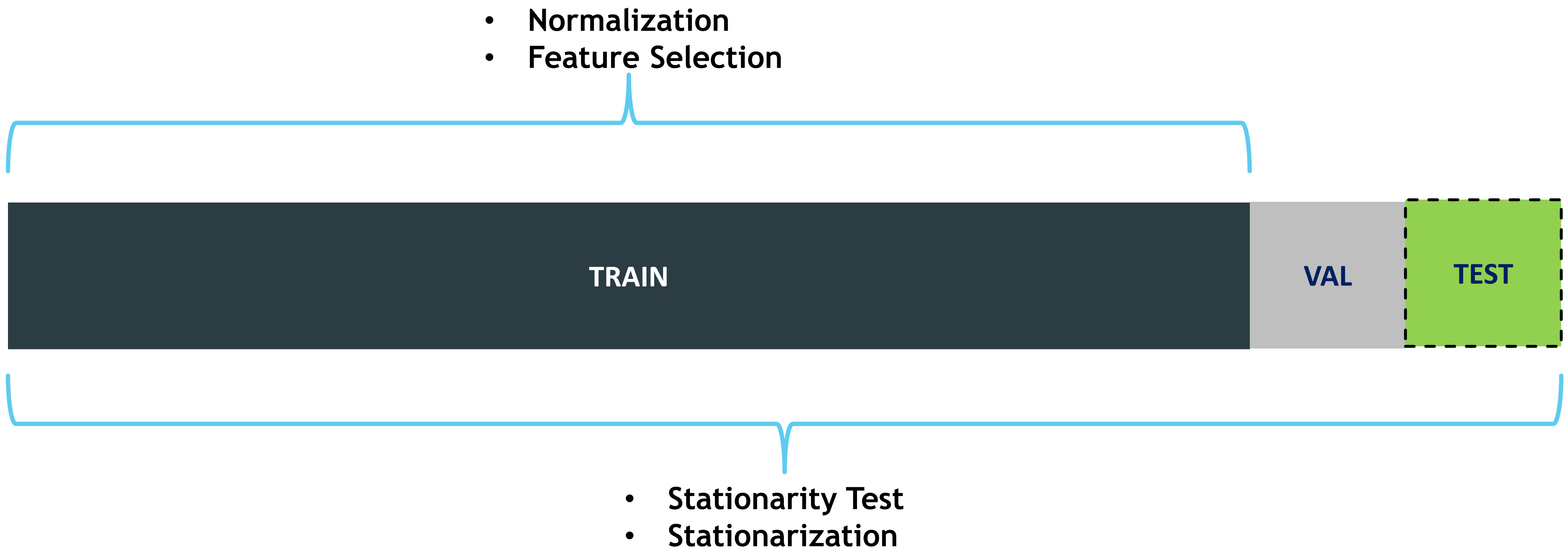
  These methods are specific to certain machine learning algorithms, like:
    - LASSO and Ridge Regression (linear regression techniques)
    - Decision Trees and Ensemble Methods (Random Forest, Gradient Boosting)

  - **Advantages:** Model specific and efficient (feature selection is embedded in the train process)
  - **Drawbacks:** The selected features may vary depending on the specific hyperparameters and settings of the algorithm



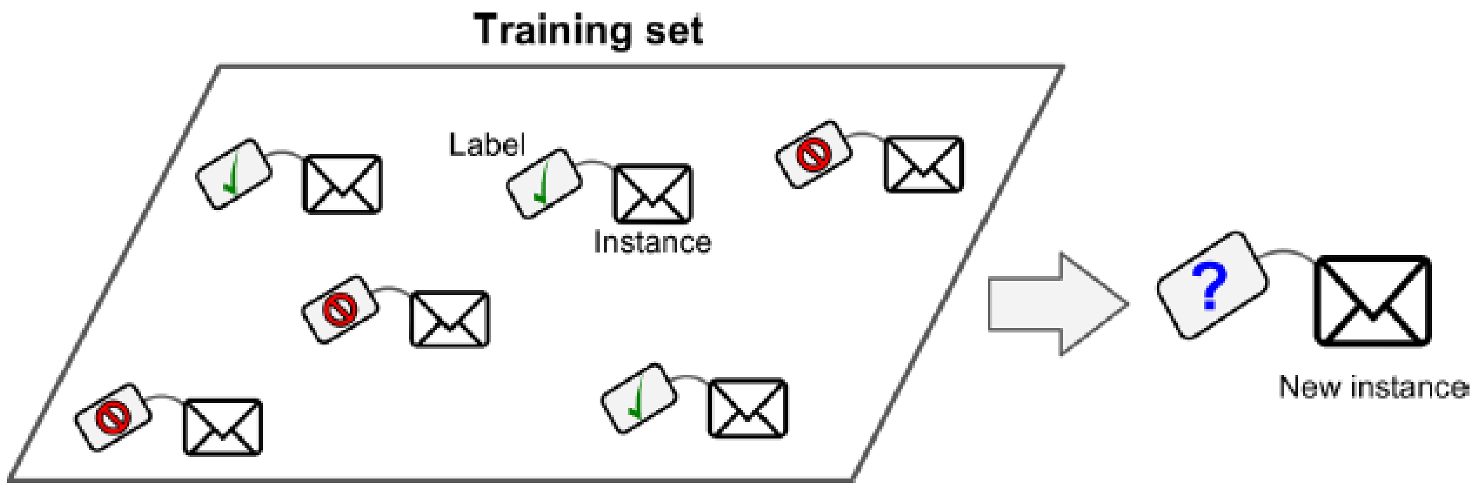
Data Preprocessing

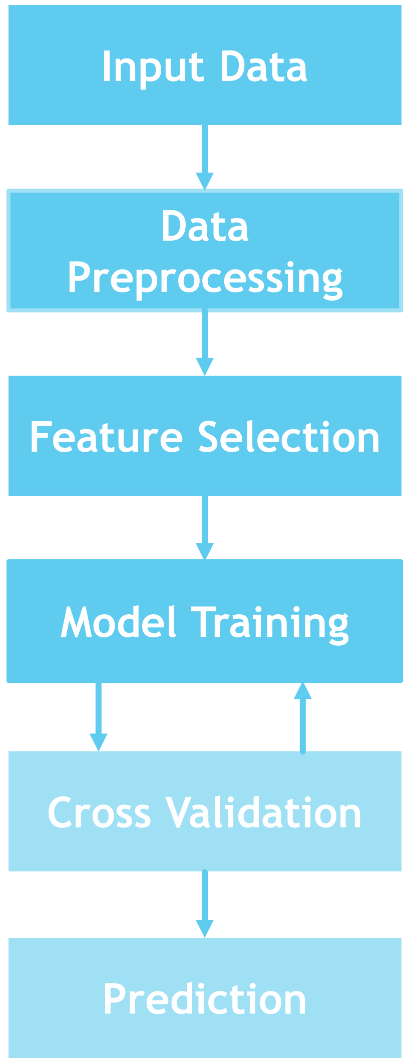
- The **feature selection** and **normalization** operations have to be done necessarily on the **train set**
- The stationarity test (and consequently the stationarization) can be done on all dataset



Model Training

- Machine Learning models may be divided into three (or more) categories:
  - **Supervised learning models:** they define a mapping between the input variables and the outputs. The target variable is known.





Model Training – Supervised Learning

- Let X be the input variables, y the target variable and Θ the set of model parameters. A **supervised ML** model f can be defined as:

$$y = f(X, \Theta)$$

- When dealing with ML models we always have a **Loss function $\mathcal{L}$** or a **Score function $\mathcal{S}$**. Both are functions of the ML model and the set of parameters Θ.

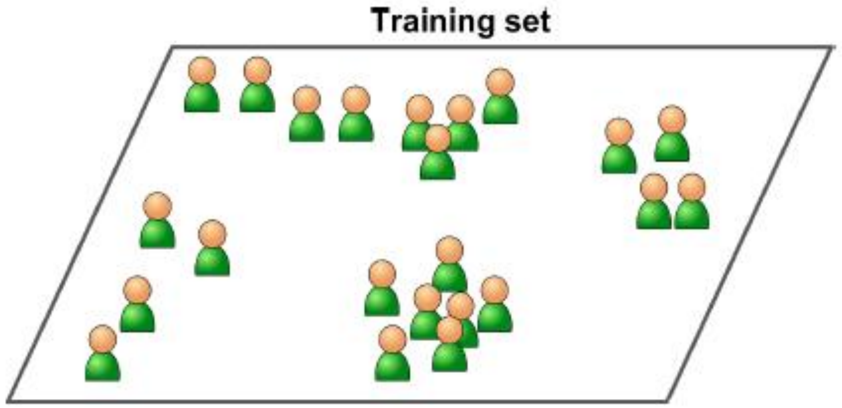
$$\mathcal{L}\big(y, f(X, \Theta)\big) \text{ or } \mathcal{S}\big(y, f(X, \Theta)\big)$$

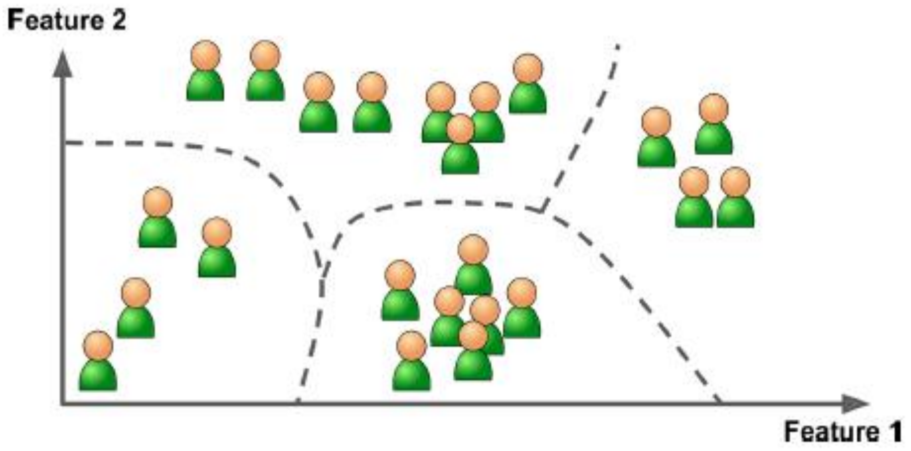
- The estimation of a ML model is then finding the optimal set of parameters Θ^* such that:

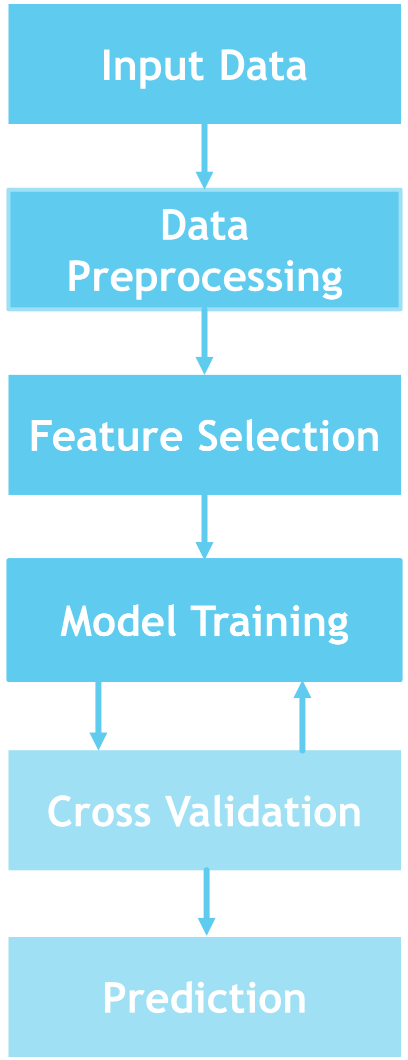
$$\Theta^* = \min \mathcal{L}\big(y, f(X, \Theta)\big) \quad \text{or} \quad \Theta^* = \max \mathcal{S}\big(y, f(X, \Theta)\big)$$

Model Training

- **Unsupervised learning models:** The goal of unsupervised learning is to *find hidden patterns in unlabeled data*, i.e. the target variable is not known.
  - The most common use of unsupervised machine learning is to cluster data into groups of similar examples (i.e. ***clustering***). Other examples are dimensionality reduction and anomaly detection models.

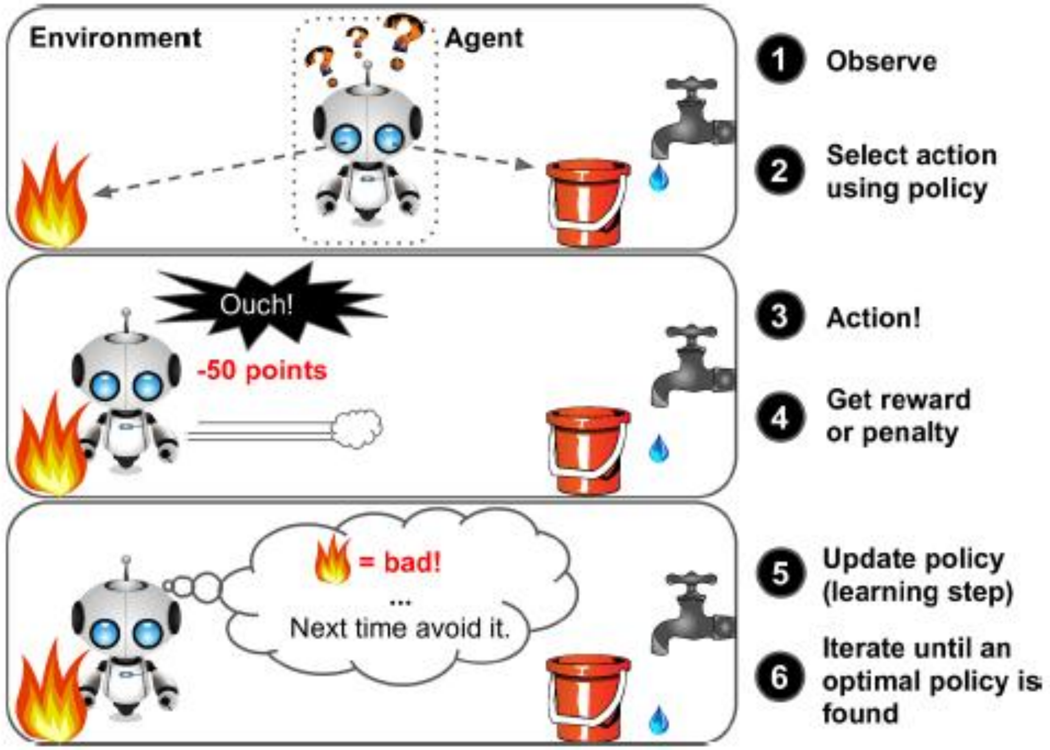


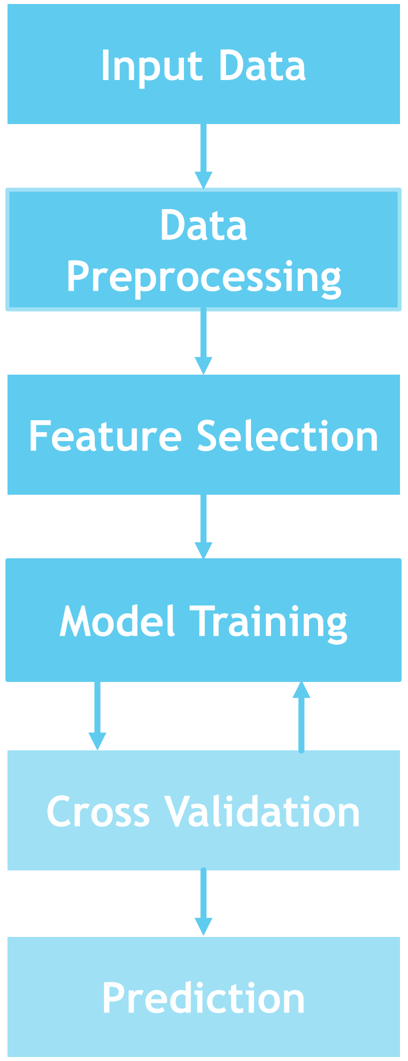




Model Training

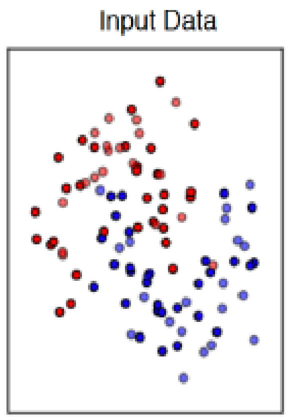
- **Reinforcement learning models:** an agent learns to make decisions by interacting with an environment. The agent takes actions, and the environment provides feedback in the form of rewards or punishments.
  - The goal of reinforcement learning is to maximize an ultimate reward through feedback after a sequence of actions.


Environment
Agent
Ouch!
-50 points
= bad!
...
Next time avoid it.
1 Observe
2 Select action using policy
3 Action!
4 Get reward or penalty
5 Update policy (learning step)
6 Iterate until an optimal policy is found



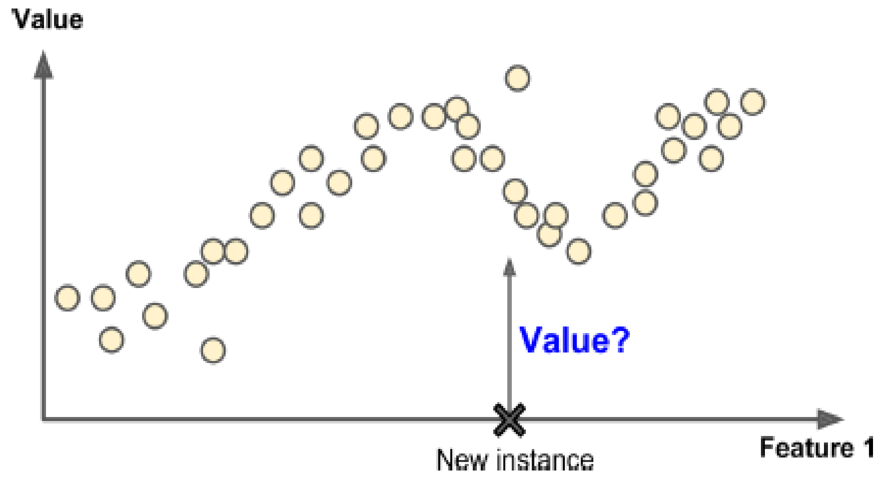
Model Training

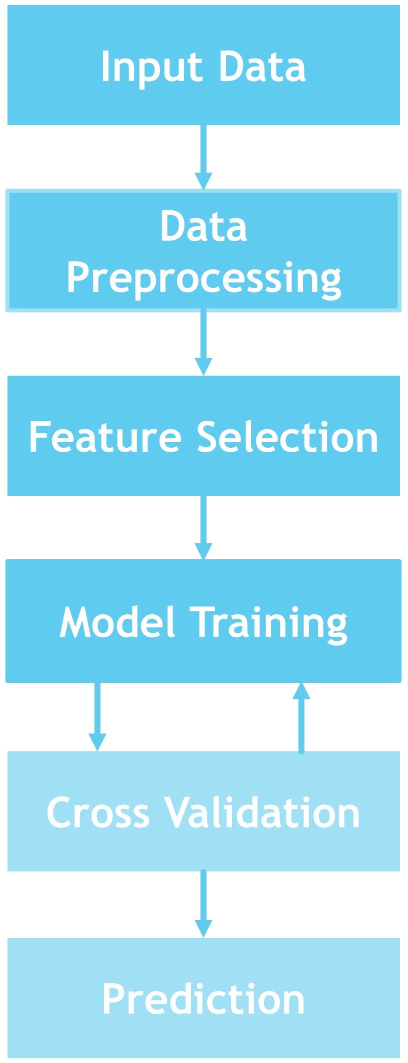
- Another way to categorize ML models is the following:
  - **Classification model:** predict the **category of instances** among two or more discrete classes (e.g. Yes or No)
  - **Regression model:** predict the **continuous** (typically, floating-point) **values** (probabilities)



How to classify this dataset into 2 categories ?

red and blue

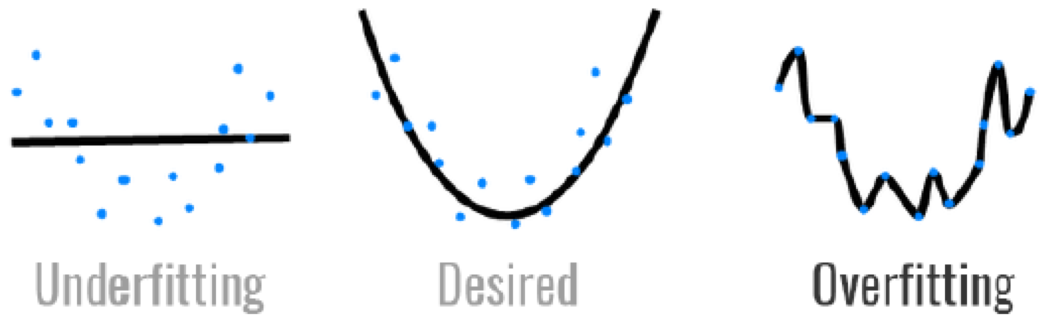




Model Training – Fitting

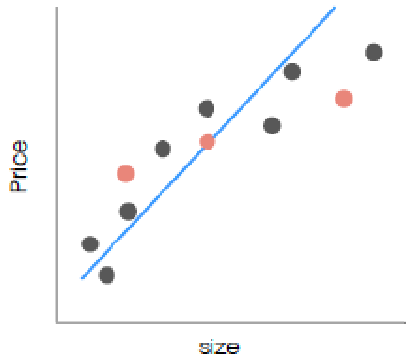
- **Overfitting** the training data: creating a complex model that matches the training data so closely that the model fails to make correct predictions on new data (not generalized well).
    - Possible solutions:
        - Simplify the model by selecting one with fewer parameters
        - Gather more training data
        - Reduce the noise in the training data
        - Constraining a model to make it simpler and reduce the risk of overfitting is called **regularization**

- **Underfitting** the training data: model is too simple to learn the underlying structure of the data.
    - Possible solutions:
        - Selecting a more powerful model, with more parameters
        - Feeding better features
        - Reducing the constraints of the model

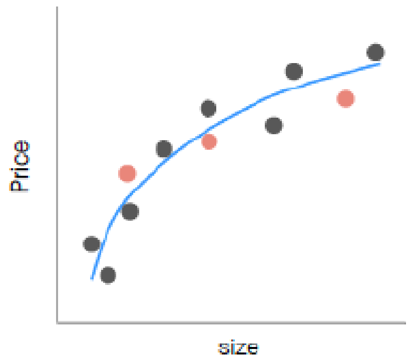


Model Training – Fitting

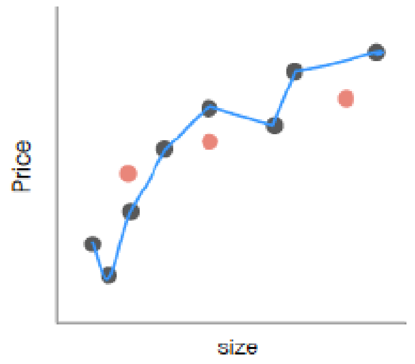
Regression example



Under fitting model
(high training error, high test error)



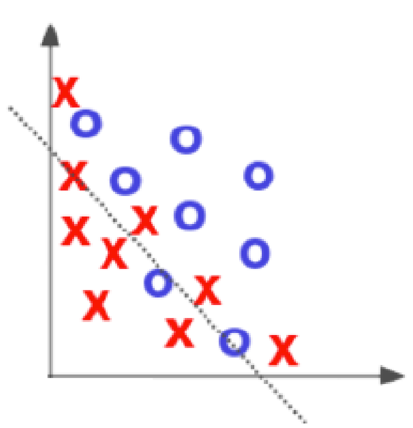
Good Fitting
(low training error, low test error)



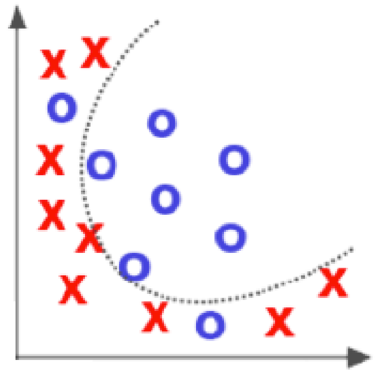
Overfitting Model
(no training error, high test error)

~ Model built
● Known data
● New data

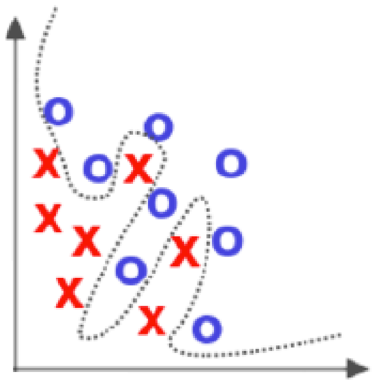
Classification example



Under fitting model
(high training error, high test error)



Good Fitting
(low training error, low test error)



Overfitting Model
(no training error, high test error)

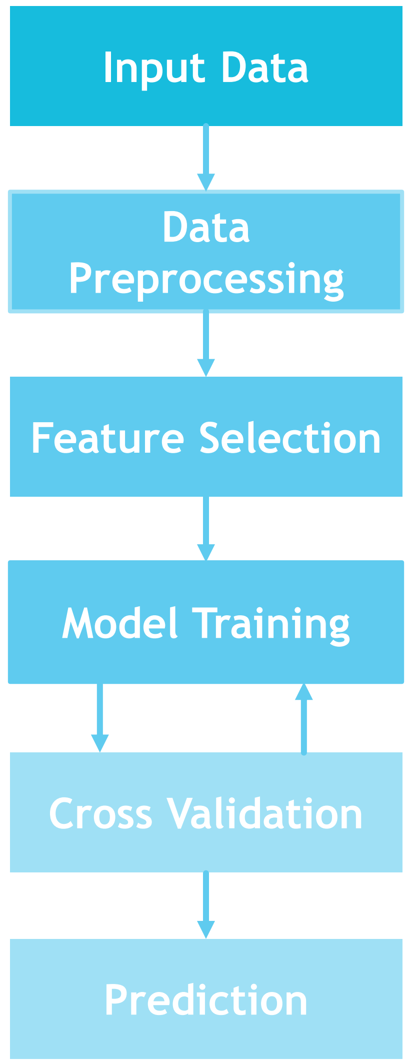
Cross-Validation

- In order to select the model with the best out-of-sample behavior, i.e. the best set of parameters, it is useful to test it on data that have not been seen by the model, i.e. *validation set*.

- The **calibration** of the parameters takes the following steps:

  1. Estimation of parameters Θ^* on the training data
  2. Performance computation of the model on the validation set
  3. Comparison of the performance in validation and in training
  4. Selection of the best model (best in validation and most similar to the training)

- To avoid "wasting" too much training data in validation sets, a common technique is to use ***cross-validation***:
  - The training set is split into k complementary subsets, and each model is repeatedly trained against hold-out validation for k times. Then the average error across all k-folds is used for model selection.
  - A final model is trained using these hyper-parameters on the full training set, and the generalized error is measured on the test set.



Prediction

- Let’s define $f^*(X, \Theta^*)$ as the optimal model resulting from the cross-validation procedure, then we can compute the ***estimation for the target variable*** as:

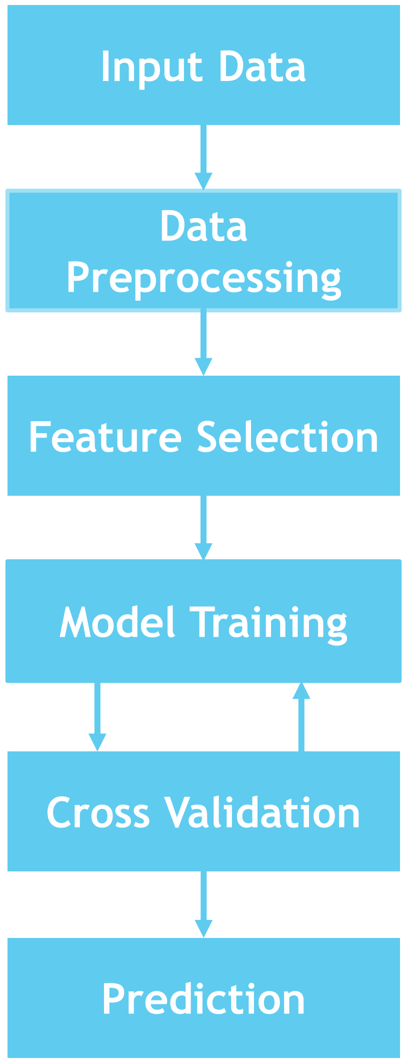
$$\hat{y} = f^*(X, \Theta^*)$$

- In finance, once we have the predictions, i.e. probabilities of belonging to a certain class, we have to transform the outputs into **signals**, thus **building a strategy**.

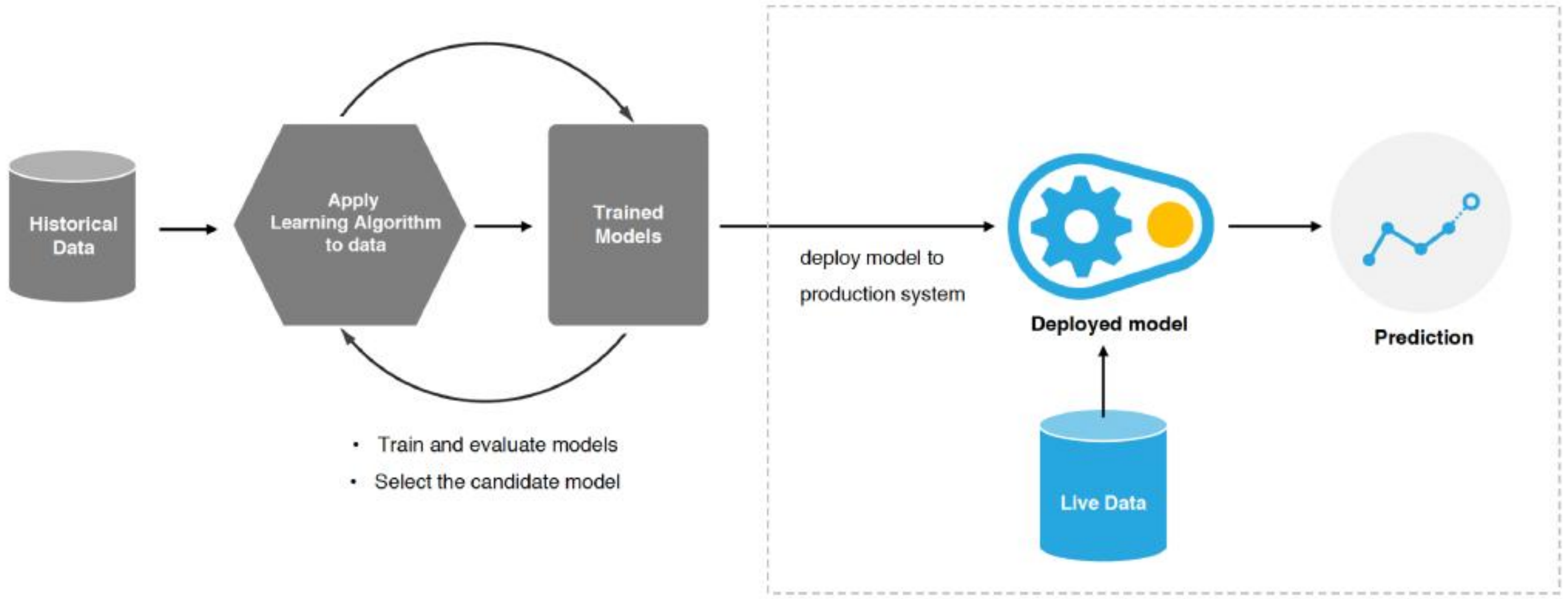
- Ex:
  - **Input Data:** price SP500
  - **Features:** Price indicators
  - **Target variable:** for a binary classification model,

$$\begin{cases} sign\big(ret(21\ days)\big) > 0 & \text{class1} = 1 \\ sign\big(ret(21\ days)\big) < 0 & \text{class2} = 0 \end{cases}$$

  - **ML Model**, i.e. Random Forest, Neural Network, …
  - **Output:** Probability of being in class1 (and in class2). From this probability we have to build a strategy, e.g. we use the prob. of being in class1 as a beta exposure



Live Deploying


Historical Data
Apply Learning Algorithm to data
Trained Models
deploy model to production system
Deployed model
Prediction
Live Data
Train and evaluate models
Select the candidate model

Unsupervised Learning - Clustering

Unsupervised Learning

- ❑ Unsupervised learning is a type of machine learning where the algorithm learns patterns and structures from data without labeled outcomes or explicit instructions about what to predict

  - ❑ **Key Concepts:**

    i. **Input Data**: The data used in unsupervised learning has no labels or target values. For example, you might have a dataset with only features like customer demographics or product attributes

    ii. **Goal**: The goal is to discover hidden patterns, structures, or groupings in the data

    iii. **Algorithms**:
       i. These algorithms work by exploring the similarities, differences, and structures in the data
       ii. Two common approaches are **clustering** and **dimensionality reduction**

Common Techniques in Unsupervised Learning

- **Clustering:** The objective is to group similar data points into clusters based on their features
  - *Examples:* Grouping customers with similar purchasing behavior (market segmentation); identifying groups of stocks with similar price movements.
  - **Popular algorithms:**
    1. *K-Means*, it divides data into k clusters by minimizing the variance within clusters
    2. *Hierarchical Clustering*, it builds a tree-like structure of nested clusters
    3. *DBSCAN*, it groups data based on density, allowing for arbitrary-shaped clusters

- **Dimensionality Reduction:** Reduces the number of variables (dimensions) in the dataset while preserving important information. Helps visualize high-dimensional data or remove noise
  - *Examples:* Visualizing financial data with thousands of features in a 2D or 3D space. Compressing features for faster computation
  - **Popular algorithms:**
    1. *PCA (Principal Component Analysis)*: Projects data onto new axes that maximize variance
    2. *t-SNE (t-Distributed Stochastic Neighbor Embedding)*: Visualizes high-dimensional data by preserving local structures
    3. *Autoencoders*: Neural networks that compress and reconstruct data

Clustering: *K-means*

- **Objective:** Partition *n* data points $x_1, x_2, .., x_n \in \mathbb{R}^d$ into *k* clusters $C_1, C_2, .., C_k$ where $\boldsymbol{x}_i$ is assigned to the cluster with the closest centroid.

- **Mathematical Formulation:** The algorithm minimizes the sum of squared distances between data points and their assigned cluster centroids:

$$\underset{\{\mu_j\},\{C_j\}}{\text{minimize}} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mu_j\|^2$$

  - $\boldsymbol{x}_i$: A data point in $\mathbb{R}^d$
  - μ_j: The centroid of cluster C_j, computed as the mean of all points in C_j

$$\mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$$

  - $\|\boldsymbol{x}_i - \mu_j\|^2$: The squared Euclidean distance

Clustering: *K-means*

- **Steps:**

1. Initialize ***k*** centroids $\mu_1, \mu_2, .., \mu_k$ randomly
2. Assign each data point $\boldsymbol{x_i}$ to the nearest cluster $\boldsymbol{j}$:

$$j = \arg\min_j ||\mathbf{x}_i - \mu_j||^2$$

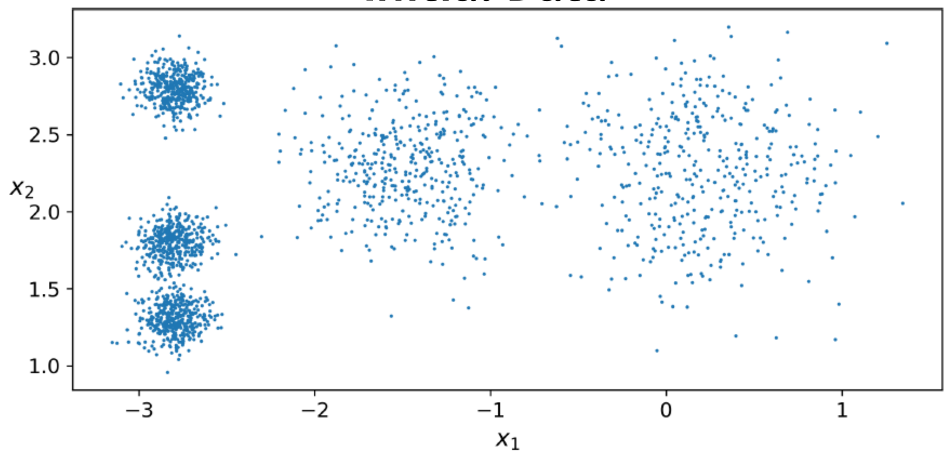
3. Update centroids μ_j as the mean of the assigned points:

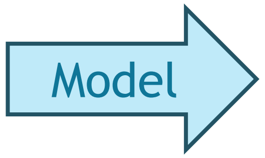
$$\mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$$

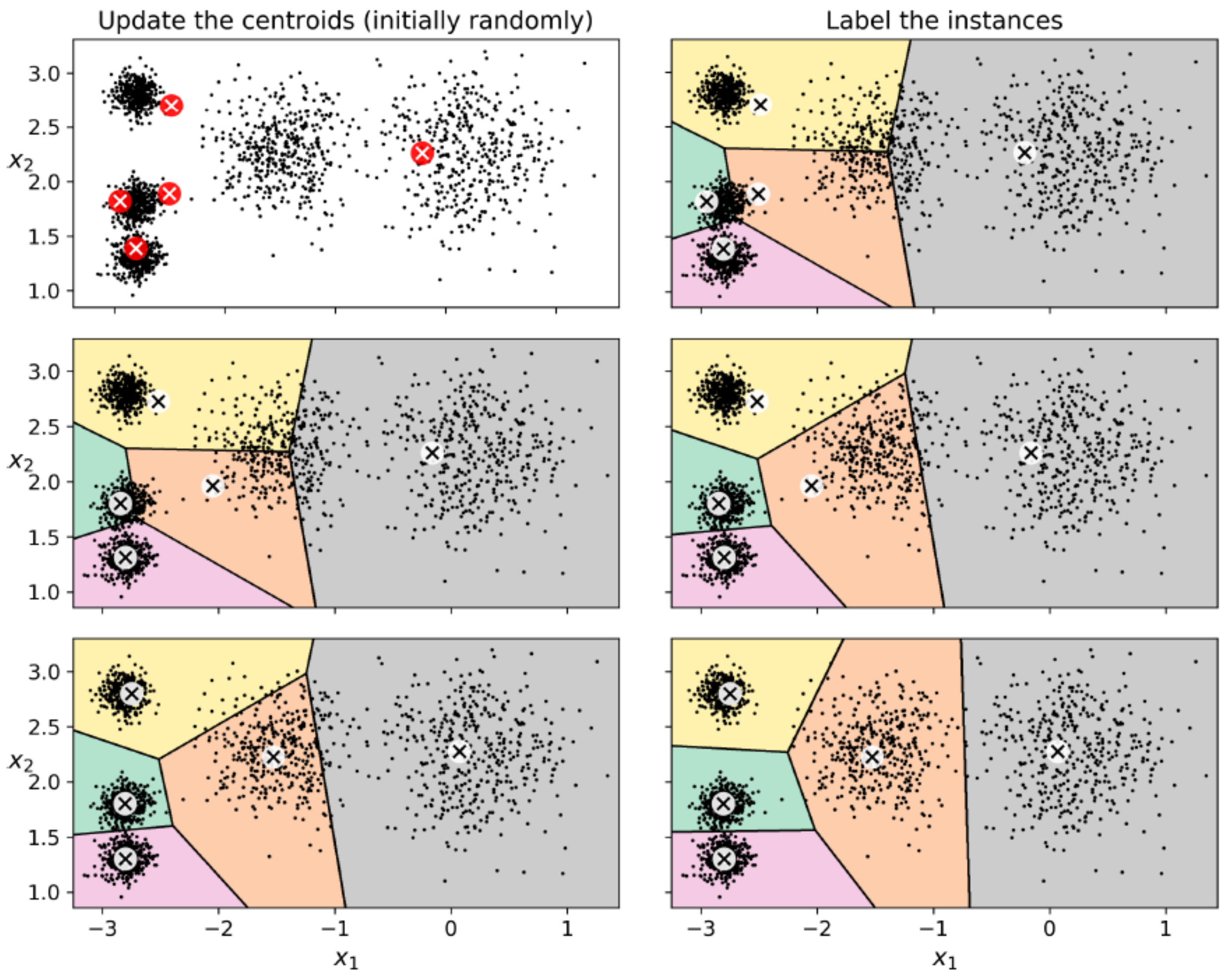
4. Repeat steps 2 and 3 until convergence (e.g., centroids no longer change)

Clustering: *K-means*

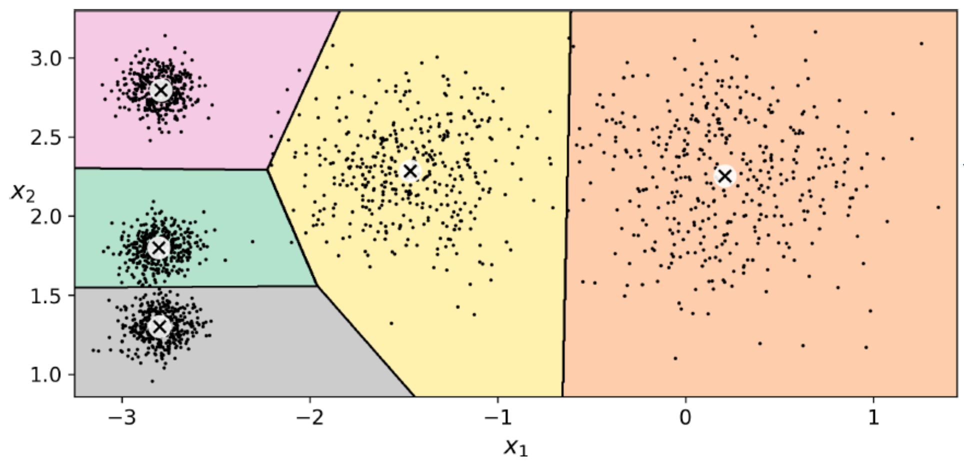
Initial Data

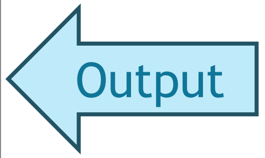






Final Data



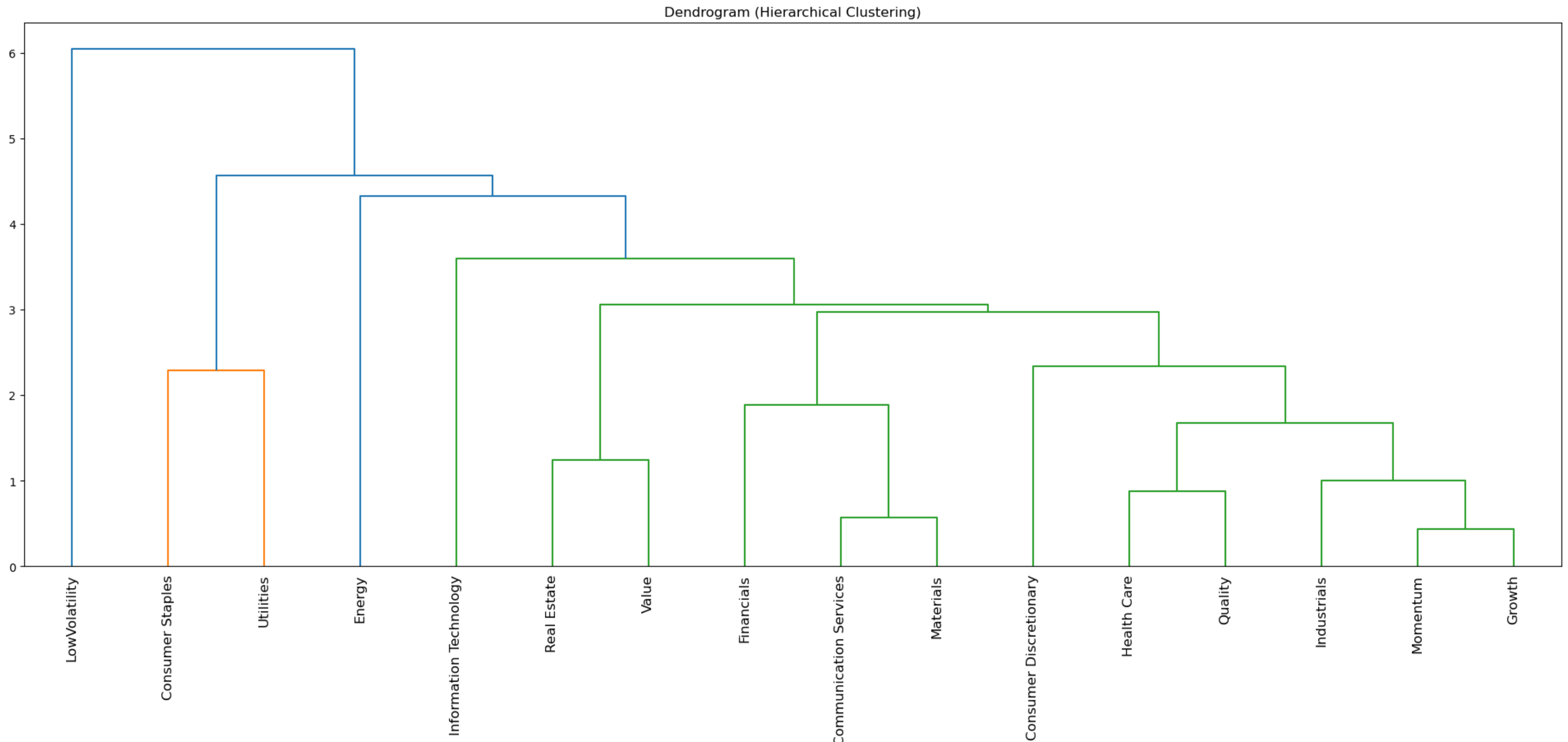


Clustering: *Hierarchical Clustering*

- **Hierarchical clustering:** is a clustering method that builds a hierarchy of clusters by recursively partitioning the data. This can be done in two ways:

  - *Agglomerative (Bottom-Up)*: Starts with each data point as its own cluster. Iteratively merges the closest clusters until all data points are in a single cluster.
  - *Divisive (Top-Down)*: Starts with all data points in one cluster. Iteratively splits clusters into smaller clusters until each point is its own cluster

- It creates a ***dendrogram***, a tree-like diagram that shows the relationship between clusters and data points, allowing users to visualize and decide on the appropriate number of clusters



Clustering: *Hierarchical Clustering*

❑ **Steps**

i. *Compute the Distance Matrix*: Calculate pairwise distances between all data points using a distance metric (e.g., Euclidean, Manhattan, or cosine distance)

ii. *Initialize Clusters*: Treat each data point as a single cluster

iii. *Merge Clusters*: Find the two clusters that are closest based on a linkage criterion (see below). Merge them into a single cluster

iv. *Update the Distance Matrix*: Recalculate distances between the newly formed cluster and all other clusters

v. *Repeat Until Termination*: Continue merging clusters until all data points form one large cluster, or a stopping criterion (e.g., desired number of clusters) is met

Clustering: *Hierarchical Clustering*

➢ The linkage criterion determines how the distance between two clusters is calculated:

1. **Single Linkage**:
   - Distance between the closest pair of points in two clusters:
     $$d_{\text{single}}(C_i, C_j) = \min_{\mathbf{x}_a \in C_i, \mathbf{x}_b \in C_j} \|\mathbf{x}_a - \mathbf{x}_b\|$$
   - Tends to create long, "chain-like" clusters.
2. **Complete Linkage**:
   - Distance between the farthest pair of points in two clusters:
     $$d_{\text{complete}}(C_i, C_j) = \max_{\mathbf{x}_a \in C_i, \mathbf{x}_b \in C_j} \|\mathbf{x}_a - \mathbf{x}_b\|$$
   - Tends to create compact clusters.
3. **Average Linkage**:
   - Average distance between all pairs of points in two clusters:
     $$d_{\text{average}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_j} \|\mathbf{x}_a - \mathbf{x}_b\|$$
   - Balances single and complete linkage.
4. **Centroid Linkage**:
   - Distance between the centroids of two clusters:
     $$d_{\text{centroid}}(C_i, C_j) = \|\mu_i - \mu_j\|$$
   - Sensitive to cluster size and outliers.

Clustering algorithms

Algorithm	Objective	Characteristics	Advantages	Limitations
K-Means	Partition data into k clusters by minimizing within-cluster variance	- Hard clustering - Iterative algorithm - Requires k as input - Uses Euclidean distance	- Simple and fast - Works well for large datasets - Easy to interpret	- Sensitive to initialization - Requires k - Assumes spherical clusters - Struggles with noise and outliers
Hierarchical Clustering	Build a tree-like structure (dendrogram) of nested clusters	- Two types: Agglomerative (bottom-up) and Divisive (top-down) - Does not require pre-defining k	- Produces a hierarchy of clusters - No need to specify k - Works well for small datasets	- Expensive for large datasets - Sensitive to the choice of distance metric and linkage method
DBSCAN	Group points in high-density regions and identify low-density points as outliers	- Density-based - Does not assume clusters are spherical	- Detects clusters of arbitrary shape - Handles noise and outliers well - No need to specify k	- Sensitive to parameters - Struggles with varying cluster densities and high dimensions
Gaussian Mixture Models (GMMs)	Model data as a mixture of Gaussian distributions	- Soft clustering - Probabilistic approach - Uses Expectation-Maximization (EM) for optimization	- Captures complex cluster shapes - Provides probabilities for cluster assignments - Flexible	- Computationally intensive - Requires pre-specifying k - Sensitive to initialization and assumptions about the data