Chapter II: Machine Learning

II.1 Introduction

Machine learning is a branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data without being explicitly programmed for every task. In simple words, ML teaches systems to think and understand like humans by learning from the data.

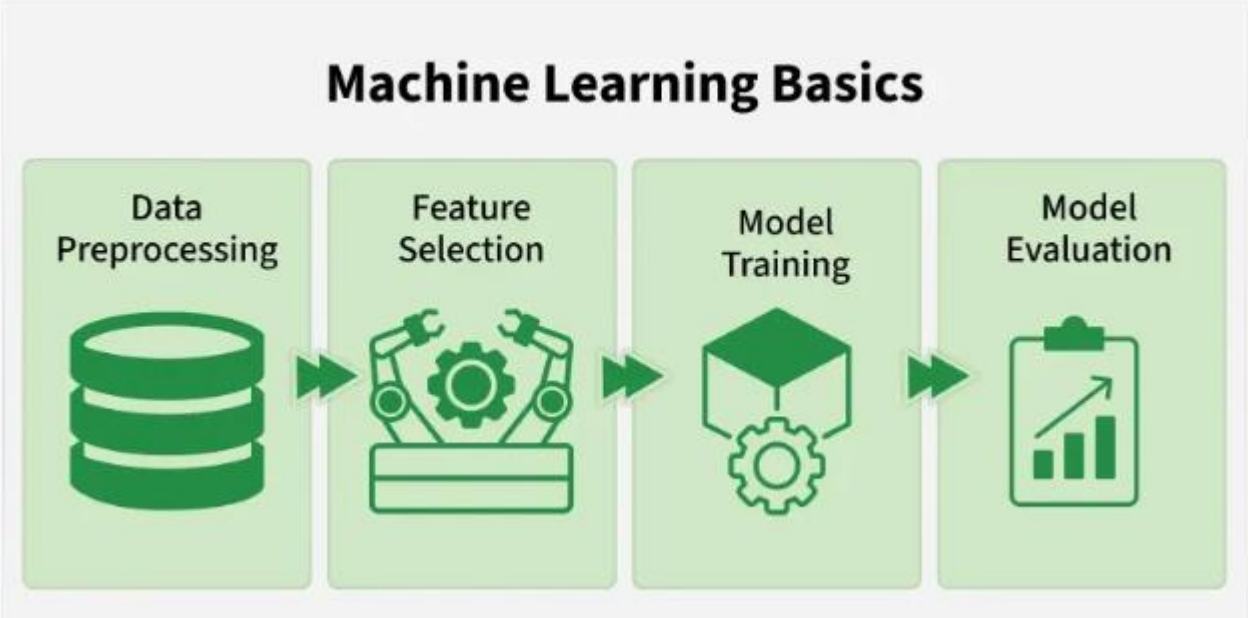


Figure II.1: Machine Learning.

Machine Learning is mainly divided into three core types:

- **Supervised Learning:** Trains models on labeled data to predict or classify new, unseen data.
- **Unsupervised Learning**: Finds patterns or groups in unlabeled data, like clustering or dimensionality reduction.
- **Reinforcement Learning**: Learns through trial and error to maximize rewards, ideal for decision-making tasks.

Note: The following are not part of the original three core types of ML, but they have become increasingly important in real-world applications, especially in deep learning.

Additional Types:

- **Self-Supervised Learning**: It is often considered as a subset of unsupervised learning, but it has grown into its own field due to its success in training large-scale models. It generates its own labels from the data, without any manual labeling.

- **Semi-Supervised Learning:** This approach combines a small amount of labeled data with a large amount of unlabeled data. It's useful when labeling data is expensive or time-consuming.

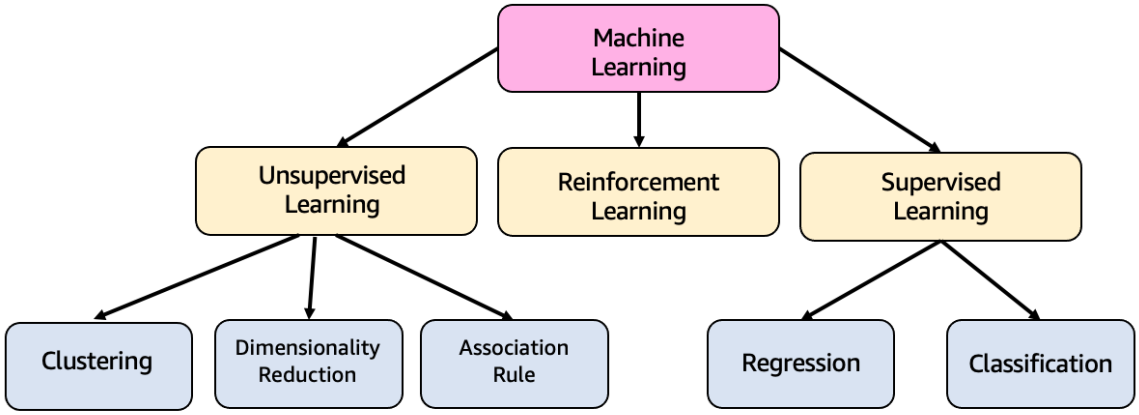


Figure II.2: Types of Machine Learning.

II.2 Supervised Learning

Supervised learning algorithms are generally categorized into **two main types:**

- **Classification**: where the goal is to predict discrete labels or categories
- **Regression**: where the aim is to predict continuous numerical values.

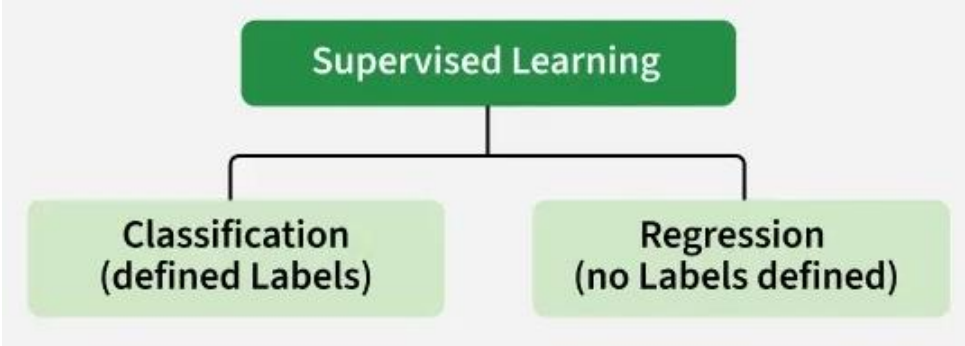


Figure II.3: Types of Supervised Learning.

II.2.1 Classification based Supervised Learning

In supervised classification, the objective is to learn a mapping function from an input space X to a discrete output space y, using a dataset of labeled examples. Each sample consists of a feature vector and its corresponding class label. The learned model aims to generalize well to unseen data.

To properly evaluate generalization performance, the available dataset is divided into disjoint subsets. The most common practice is to split the data into a training set and a test set. Typical splitting ratios include 70% training – 30% testing, 75% training – 25% testing, or 80% training – 20% testing.

The training set contains both input features and their corresponding labels. It is used exclusively to estimate the parameters of the model. During this phase, supervised learning algorithms learn the relationship between inputs and outputs by minimizing a predefined loss function.

The test set, in contrast, is used only for performance evaluation. It is not involved in the learning process. Instead, it provides an unbiased estimate of the model's generalization capability by measuring metrics such as accuracy, error rate, or confusion matrix.

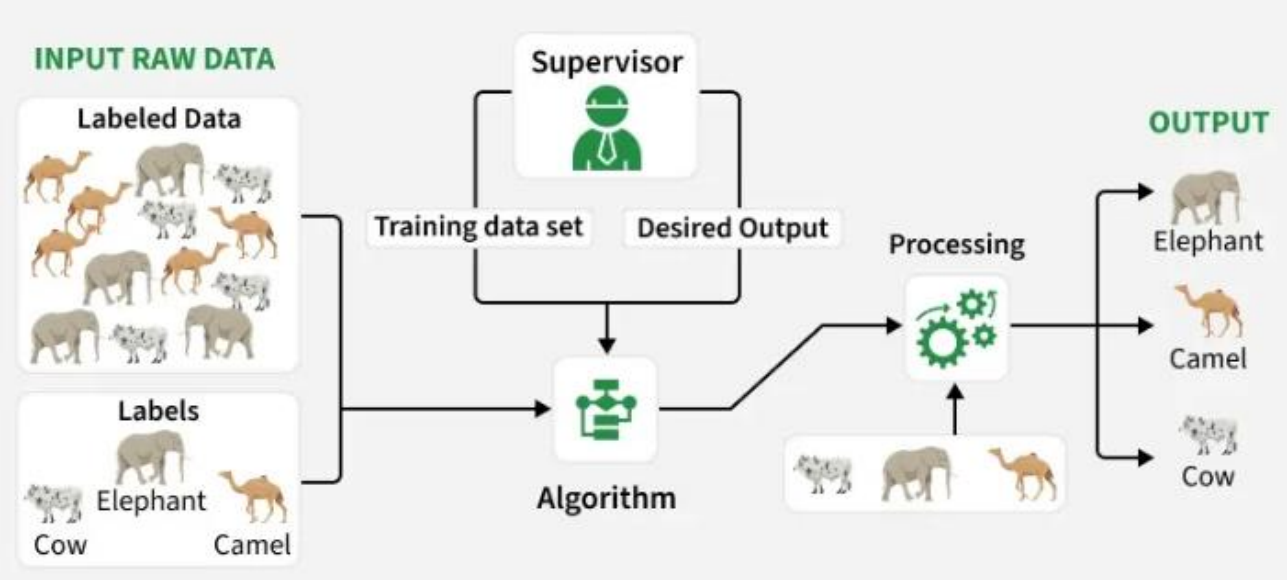


Figure II.4: Classification based supervised learning.

In supervised classification, we consider an input space:

$$\mathcal{X} \subseteq \mathbb{R}^d$$

And a finite output space (set of classes):

$$\mathcal{Y} = \{C_1, C_2, \ldots, C_K\}$$

We are given a labeled dataset:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

Where:

- $x_i \in \mathcal{X}$ is the feature vector,
- $y_i \in \mathcal{Y}$ is the corresponding class label.

The objective is to learn a classifier

$$g : \mathcal{X} \to \mathcal{Y}$$

That approximates the unknown true mapping between inputs and outputs, and minimizes the expected risk:

$$R(g) = \mathbb{E}[L(Y, g(X))],$$

where $L(\cdot)$ is a loss function (commonly the 0–1 loss).

- **Data Splitting**

To evaluate the generalization performance of the classifier, the dataset D is partitioned into disjoint subsets:

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}, \quad \mathcal{D}_{train} \cap \mathcal{D}_{test} = \varnothing.$$

Typically,

$$N = N_{train} + N_{test}.$$

Common splitting ratios are :

$$\frac{N_{train}}{N} \in \{0.7, 0.75, 0.8\}, \quad \frac{N_{test}}{N} = 1 - \frac{N_{train}}{N}$$

- **Training Phase**

The classifier is learned using only the training set:

$$\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1}^{N_{train}}$$

Model parameters θ are estimated by minimizing the empirical risk:

$$\hat{R}_{train}(g) = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} L(y_i, g(x_i))$$

Different supervised learning algorithms (e.g., k-NN, LDA, GMM, SVM, neural networks) correspond to different hypothesis spaces and optimization procedures.

- **Testing Phase**

The test set:

$$\mathcal{D}_{test} = \{(x_i, y_i)\}_{i=1}^{N_{test}}$$

Is used exclusively for performance evaluation.

The empirical test error is:

$$\hat{R}_{test}(g) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} L(y_i, g(x_i))$$

This provides an unbiased estimate of the true generalization error R(g).

During **training**, the classifier is learned by minimizing the **empirical risk** over the training set, which consists of input–output pairs (x_i,C_k). This means the model adjusts its parameters to correctly classify the training samples as much as possible. However, empirical risk is only an approximation of the true risk over the unknown data distribution. During **testing**, the model is evaluated on a separate test set to estimate its generalization performance, i.e., how well it will classify unseen data. To understand the theoretical limit of performance, we refer to the **Bayes classifier**, which uses the true posterior probabilities P(C_k|x) and achieves the minimum possible classification error (Bayes risk**)**. In other words, empirical risk minimization in training is a practical way to approximate the Bayes optimal decision rule, and testing measures how closely our learned classifier approaches this theoretical optimum.

In supervised classification, we have a dataset of labeled samples (x_i, C_k), where $x_i \in \mathbb{R}^d$ is the feature vector and $C_k \in \{C_1, C_2, \ldots, C_K\}$ is the class label. During **training**, the classifier is learned by minimizing the **empirical risk** over the training set:

$$\hat{R}_{train}(g) = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} L(y_i, g(x_i))$$

Where L(·) is the 0–1 loss function. This process adjusts the model parameters to classify the training samples as accurately as possible. The **test set**, consisting of unseen data, is used to evaluate the generalization performance of the classifier by computing the empirical error on the test samples.

To understand the theoretical limit of classification performance, we refer to the **Bayesian framework**. For K classes, the **posterior probability** of each class given a feature vector x is:

$$P(C_k \mid x) = \frac{P(x \mid C_k)P(C_k)}{\sum_{i=1}^{K} P(x \mid C_i)P(C_i)}$$

For each class C_k the **posterior probabilities** are:

$$P(C_k \mid x) = \frac{P(x \mid C_k)P(C_k)}{P(x)}$$

Where

$$P(x) = \sum_{i=1}^{K} P(x \mid C_i) P(C_i)$$

The Objective is Minimizing Probability of Error:

Under **0–1 loss**, the probability of error is:

$$P(error) = P(g(X) \neq Y)$$

To minimize global error, we minimize error **pointwise for each** x **.**

In the case of Conditional Probability of Error, **Suppose we decide class Cj**

An error occurs when the true class is any Ci with i≠j. Thus,

$$P(error \mid x, \text{decide } C_j) = \sum_{i \neq j} P(C_i \mid x)$$

But since

$$\sum_{i=1}^{K} P(C_i \mid x) = 1$$

We obtain

$$P(error \mid x, C_j) = 1 - P(C_j \mid x)$$

We must choose the class Cjthat minimizes Conditional Error:

$$1 - P(C_j \mid x)$$

This is equivalent to maximizing:

$$P(C_j \mid x)$$

The **Bayes classifier** chooses the class with the largest posterior probability:

$$g^*(x) = \arg\max_k P(C_k \mid x)$$

This is the **MAP rule** for K classes.

Which is equivalent to minimizing the **conditional error** at each x :

$$P(error \mid x) = 1 - \max_k P(C_k \mid x)$$

For each x , the minimum achievable error is:

$$P(error \mid x) = \min_j (1 - P(C_j \mid x))$$

Integrating over all possible x gives the **Global Bayes risk**, the minimum achievable probability of misclassification:

$$R^* = \int (1 - \max_k P(C_k \mid x))p(x)dx$$

This is the **minimum possible classification error** for K classes. Thus, **empirical risk minimization during training** can be viewed as a practical attempt to approximate this Bayes optimal decision rule, while **testing** measures how closely the learned classifier approaches the theoretical minimum error.

For any other classifier g(x) choosing class Cj:

$$P(error \mid x) = 1 - P(C_j \mid x)$$

The smallest possible value at each xxx is obtained when Cj corresponds to the largest posterior.

Thus:

$$R(g) \geq R^*$$

Therefore,

The MAP rule derived from Bayes theorem minimizes the probability of error for K classes.

Bayes theorem → Posterior probabilities $P(C_k \mid x)$ → Choose class with largest posterior (MAP)

→ Minimizes conditional error $1 - P(C_j \mid x)$ → Minimizes total probability of error (Bayes risk)

Define the Loss Matrix

Let L(Ci,Cj) be the loss incurred when the **true class** is Ci but the classifier decides Cj:

$$L(C_i, C_j) \geq 0, \quad i, j = 1, \ldots, K$$

- Correct classification: $L(C_i, C_i) = 0$
- Misclassification: $L(C_i, C_j) > 0$ for $i \neq j$
- In general, the matrix allows **different penalties** for different types of errors.

Conditional Risk for a Single Observation

For a feature vector x, the **conditional risk** of deciding class Cj is:

$$R(C_j \mid x) = \sum_{i=1}^{K} L(C_i, C_j)\, P(C_i \mid x)$$

- This is the **expected loss** if we choose C_j
- Weighted by the **posterior probabilities** $P(C_i \mid x)$

Bayes Decision Rule with General Loss

To minimize the conditional risk:

$$g^*(x) = \arg\min_{C_j} R(C_j \mid x) = \arg\min_{C_j} \sum_{i=1}^{K} L(C_i, C_j)\, P(C_i \mid x)$$

This is the **generalized MAP rule** for an arbitrary loss matrix.

Bayes Risk (Expected Risk)

The **overall risk** (Bayes risk) is obtained by averaging over all possible x:

$$R^* = \int \underbrace{\min_{C_j} \sum_{i=1}^{K} L(C_i, C_j)\, P(C_i \mid x)}_{\text{minimum conditional risk at } x} p(x)\, dx$$

- R^* is the **minimum achievable expected loss.**
- This reduces to the usual Bayes error if $L(C_i, C_j)$ is 0–1 loss.

Special Case: 0–1 Loss

If

$$L(C_i, C_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}$$

Then

$$R(C_j \mid x) = 1 - P(C_j \mid x), \quad g^*(x) = \arg\max_k P(C_k \mid x)$$

Recovering the **classical MAP/Bayes classifier**.

- **Loss matrix** $L(C_i, C_j)$ allows weighting misclassifications differently.
- **Conditional risk** at x: $R(C_j \mid x) = \sum_i L(C_i, C_j) P(C_i \mid x)$
- **Bayes decision rule**: choose the class that **minimizes conditional risk**.
- **Bayes risk**: average of minimum conditional risk over all x:

$$R^* = \int \min_j \sum_i L(C_i, C_j) P(C_i \mid x) p(x) dx$$

This is the most general form of the **Bayesian classifier and minimum risk principle**.

❖ **performance evaluation in classification-based supervised learning**

In classification, the goal is to predict a categorical label (e.g., "Spam" or "Not Spam"). Since the model outputs a prediction, we need specific tools to measure *how right or wrong* it is.

1. Confusion Matrix

The confusion matrix is one of the best ways to evaluate the performance of a classification model in the Machine Learning. It allows to evaluate model's performance, identify where it went wrong, and provide guidance on how can be fixed. This concept "confuse" the most people, particularly beginners who are just beginning with artificial intelligence or machine learning.

A **confusion matrix** represents the accuracy of a classifier in a **Machine Learning** (ML). The confusion matrix displays the number true positives and true negatives. This matrix helps in analyzing the model performance, identifying incorrect classifications, and improving prediction accuracy.

A **confusion matrix** is a N x N matrix where N is total number of target categories. It compares actual target values to those predicted by the **ML model**. This allows us to get a comprehensive view of our classification model's performance and the types of errors that it makes.

Following figure shown, a binary classification problem would require a **matrix of 2 x 2,**, which has 4 values.

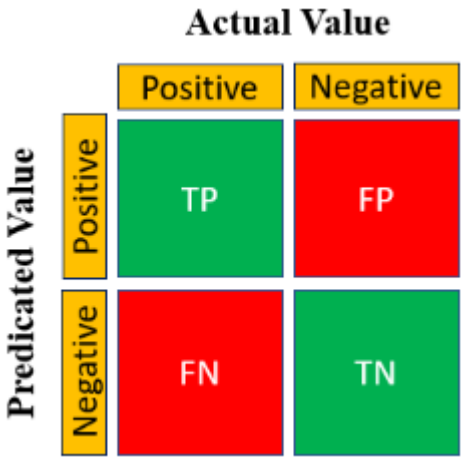


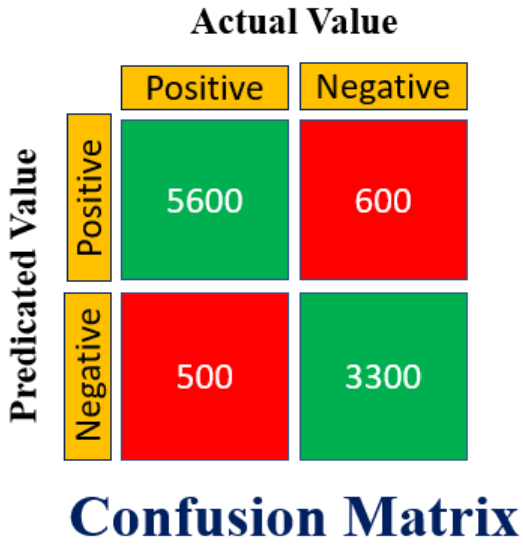
Figure II.5: Confusion matrix.

Terms used in Confusion Matrix

- **True Positive:** It indicates that the predicted value or class is the same as the actual value. The model predicted that the value would be positive.
- **True Negative:** It indicates that the predicted value or class is the same as the actual class. The model predicted that the value would be negative.
- **False Positive (FP) – Type I Error:** The value predicted was incorrect. The model predicted that the value would be positive, even though it was negative. It is also called the Type I error.
- **False Negative (FN) – Type II Error:** The value predicted was incorrect. The model predicted that the value would be negative, even though it was positive. It is also called the Type II error.

◆ **Example :Confusion Matrix**

Let understand with an example. We have developed a Machine Learning Model for **5G System** which can predict that when there can be **crash or reboot** of an gNodeB. Considering 10000 data points in a classification dataset. Following confusion matrix result show classifiers with a **logistic regression algorithm** or a **decision trees**.



Confusion Matrix

- True Positive (TP) = 5600 means the model correctly classified 5600 positive class data points. In other words, model predicted 5600 time correctly there will be a crash
- True Negative (TN) = 3300 means the model correctly predicted or classified there is no crash
- False Positive (FP) = 600 means the model incorrectly predicted there is a crash, where there was actually no crash
- False Negative (FN) = 500 meaning the model incorrectly predicted there is no crash, where there was actually a crash

This can be considered a pretty decent Machine Learning Model, showing relatively larger no. of true positive and true negative values.

❖ Why we need Confusion Matrix

Confusion matrix information helps to calculate ML Model performance in terms of Accuracy, Precision, Recall and F-1 score.

- **ML Model Accuracy:** Considering all the classes i.e. positive and negative, how many of them we have predicted correctly. Accuracy should be high as possible for a Machine Learning Model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

- **ML Model Precision:** Considering all the classes, ML model have predicted as positive vs how many are actually positive. Precision of a ML Model should be high as possible.

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}}$$

- **ML Model Recall:** Considering all the positive classes, how many we predicted correctly. Recall of a ML model should be high as possible.

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

- **ML Model F1-score:** It helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of arithmetic Mean by punishing the extreme values more.

$$\textbf{F1-Score} = \frac{\textbf{2* Recall * Precision}}{\textbf{Recall + Precision}}$$

II.2.3 Regression based Supervised Learning

Supervised Learning is a branch of Machine Learning, characterized by the making of an algorithm which learns to map an input to a particular output, using a labeled training dataset. In the previous article, we looked at the basics of supervised machine learning and one of it's branches: regression. Below, we will dive into the main algorithms used in supervised regression learning and a few applications.

Linear Regression

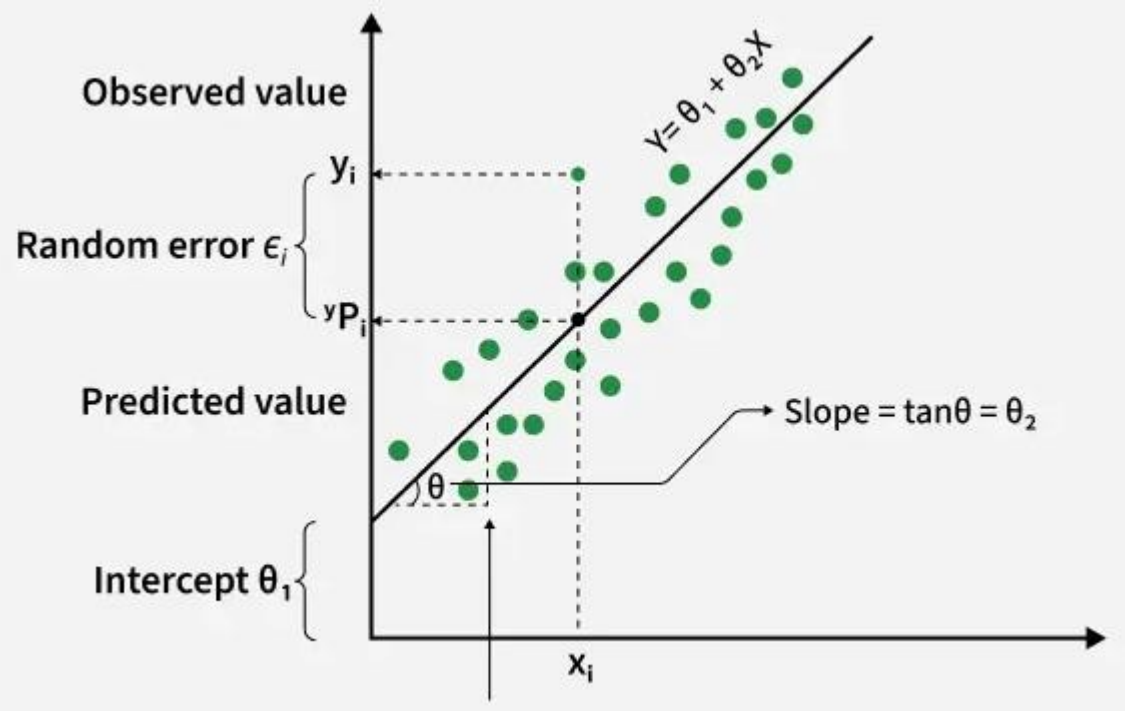
Linear regression allows to predict a dependent variable value (y) based on one or more given independent variables (x_1, x_2, ,…). Hence, this regression technique establishes a linear relationship between x (the input) and y (the output).

In this case, the regression model will establish a linear relationship between a single independent variable (x) and a continuous dependent variable (y). Thus, the model needs to start off with a hypothesis function of the following form

$$y = \theta_0 + \theta_1 x$$

In this case, corresponds to the intercept, while corresponds to the coefficient of x or, in other words, the slope. It is expected, in the case of supervised learning, that the values for x and y will be given, while training the model. While doing so, the model will fit the best line to predict the value of y for a given input value x.

To do so, it needs to find the best and values, by iterating through the training dataset. To determine the optimal and values and establish how well our model is fitting the values, we can establish some evaluation metrics.



The most popular evaluation metrics for determining performance of regression models is the Root Mean Squared Error, or RMSE, which corresponds to the root of the mean of the squared errors. The formula is given below.

$$\mathbf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)}$$

In this formula:

- n is the total number of data points
- is the actual output value
- is the predicted output value

Once the optimal values have been identified, the model should have established the linear relationship with the least error or, in other words, with a minimized RMSE.

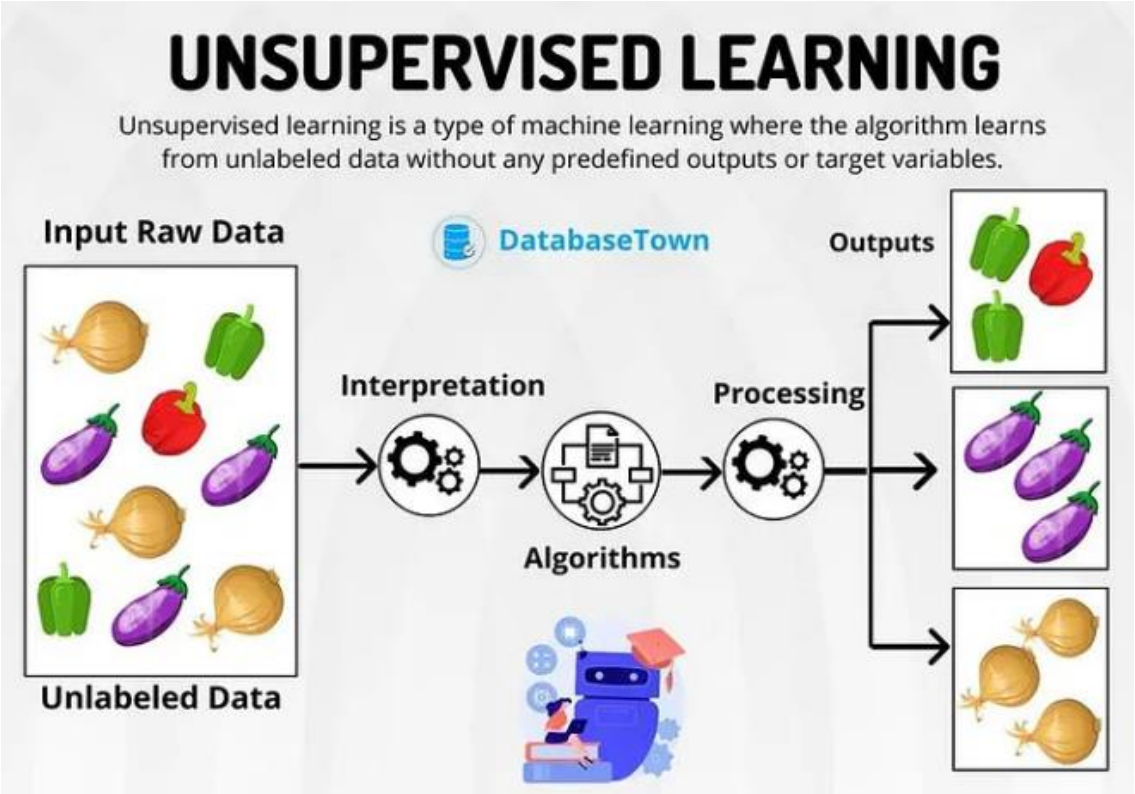
Other evaluation metrics like the Mean Squared Error (MSE), the Mean Absolute Error (MAE) or the coefficient of determination ( ) can also used.

Simple linear regression is the most common model used in finance. Indeed, it can be used for portfolio management, asset valuation or optimization. For instance, it can be used to forecast returns and the

operational performances of a business. However, there are some cases in which the target or dependent variable is not impacted only by a single predictor. Therefore, we would need a model to fit these more elaborate cases: we would use multiple linear regression.

II.3 Unsupervised learning

Unsupervised learning is a type of machine learning where the algorithm learns patterns from unlabelled data. Unlike supervised learning, where the model is trained on a dataset with input-output pairs, unsupervised learning works with data that has no labels. The goal is to identify hidden structures and patterns within the data.



Unsupervised learning are again divided into **four main categories** based on their purpose:

- Clustering
- Association Rule Mining
- Dimensionality Reduction
- Anomaly Detection

1. Clustering

- Clustering is the process of grouping similar data points together based on their features. The idea is to ensure that points within the same group (cluster) are more similar to each other than to those in other groups. Clustering is one of the most widely used techniques in Unsupervised Learning.

2. Dimensionality Reduction

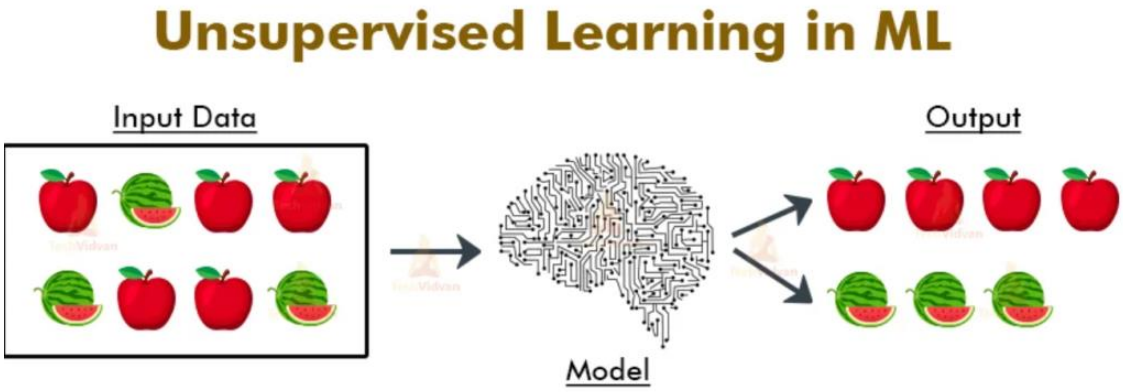
- Dimensionality Reduction involves reducing the number of features (dimensions) in a dataset while preserving as much information as possible. This technique is crucial for visualizing high-dimensional data and reducing computational complexity.

3. Anomaly Detection

- Anomaly Detection is the process of identifying rare items, events, or observations that do not conform to the general pattern of the data. These anomalies are often of interest because they may represent critical incidents, such as fraud or malfunction.

4. Association Rules

- Association Rules are used to discover relationships between variables in large datasets. This technique is often used in market basket analysis to identify items that frequently co-occur in transactions.



Common Algorithms in Unsupervised Learning

Several algorithms can be employed in Unsupervised Learning, each tailored to different types of tasks and data structures. Below are some of the most commonly used algorithms:

1. k-Means Clustering

- **Type:** Clustering
- **Description:** k-Means is one of the most popular clustering algorithms. It partitions the data into k clusters by minimizing the sum of squared distances between data points and the centroid of their assigned cluster.
- **Use Case:** Customer segmentation, where customers are grouped based on their purchasing behavior.

2. Hierarchical Clustering

- **Type:** Clustering
- **Description:** Hierarchical Clustering builds nested clusters by either merging smaller clusters into larger ones (agglomerative) or splitting larger clusters into smaller ones (divisive). The results are often represented in a dendrogram, a tree-like diagram.
- **Use Case:** Gene expression data analysis, where genes with similar expression patterns are grouped together.

3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- **Type:** Clustering
- **Description:** DBSCAN groups together points that are closely packed based on a distance metric and identifies points in low-density regions as outliers. Unlike k-Means, DBSCAN does not require specifying the number of clusters beforehand.
- **Use Case:** Identifying geographical clusters of disease outbreaks.

4. Principal Component Analysis (PCA)

- **Type:** Dimensionality Reduction
- **Description:** PCA reduces the dimensionality of a dataset by transforming it into a set of linearly uncorrelated variables called principal components. These components capture the most variance in the data.
- **Use Case:** Reducing the number of features in image compression while retaining essential information.

5. t-Distributed Stochastic Neighbor Embedding (t-SNE)

- **Type:** Dimensionality Reduction
- **Description:** t-SNE is a non-linear dimensionality reduction technique that is particularly useful for visualizing high-dimensional datasets in 2D or 3D. It preserves the local structure of the data, making it ideal for visualizing clusters.
- **Use Case:** Visualizing clusters in high-dimensional datasets, such as word embeddings in natural language processing.

6. Autoencoders

- **Type:** Dimensionality Reduction

- **Description:** Autoencoders are neural networks that learn to compress data into a lower-dimensional representation and then reconstruct the original data from this compressed form. They are used for tasks like noise reduction and anomaly detection.
- **Use Case:** Image denoising, where the goal is to remove noise from images while preserving essential details.

7. Apriori Algorithm

- **Type:** Association Rules
- **Description:** The Apriori Algorithm is used to find frequent itemsets in transactional data and to generate association rules. It is based on the principle that any subset of a frequent itemset must also be frequent.
- **Use Case:** Market basket analysis, where the goal is to identify products frequently purchased together.
-

II.3.1 Implementing Unsupervised Learning Step-by-Step

Implementing Unsupervised Learning involves several steps, from data preparation to model evaluation. Here's a basic outline:

1. Data Collection

- Gather data relevant to the problem you want to solve. The data should be comprehensive enough to reveal patterns and structures.

2. Data Preprocessing

- **Cleaning:** Handle missing values, remove duplicates, and address inconsistencies in the data.
- **Normalization/Standardization:** Scale features so that they contribute equally to the model's predictions, especially in clustering tasks.
- **Dimensionality Reduction:** If dealing with high-dimensional data, consider applying PCA or another dimensionality reduction technique to simplify the data.

3. Choosing a Model

- Select an appropriate algorithm based on the problem type (clustering, dimensionality reduction, or association) and the characteristics of your data.

4. Training the Model

- Apply the chosen algorithm to the dataset. In Unsupervised Learning, “training” often involves identifying clusters, reducing dimensions, or discovering associations rather than optimizing a predictive model.

5. Evaluating the Model

- **Silhouette Score:** Used to evaluate the quality of clusters by measuring how similar a point is to its own cluster compared to other clusters.
- **Explained Variance:** In PCA, measures the proportion of the dataset’s variance captured by the principal components.
- **Visual Inspection:** Techniques like t-SNE often require visual inspection to assess the quality of the clusters or the data projection.

6. Interpreting Results

- Analyze the patterns, clusters, or associations discovered by the model. These insights can guide further analysis or decision-making.

7. Deploying the Model

- Once satisfied with the model’s results, deploy it to identify patterns, clusters, or anomalies in new data.

II.3.1 Applications of Unsupervised Learning

Unsupervised Learning is used in a variety of industries and domains:

- **Marketing:** Customer segmentation, targeted marketing, and identifying customer personas.
- **Finance:** Fraud detection, risk assessment, and market segmentation.
- **Healthcare:** Disease outbreak detection, patient clustering for personalized treatment, and gene expression analysis.
- **Retail:** Market basket analysis, inventory management, and product recommendation systems.
- **Social Media:** Topic modeling, sentiment analysis, and user clustering.

Challenges in Unsupervised Learning

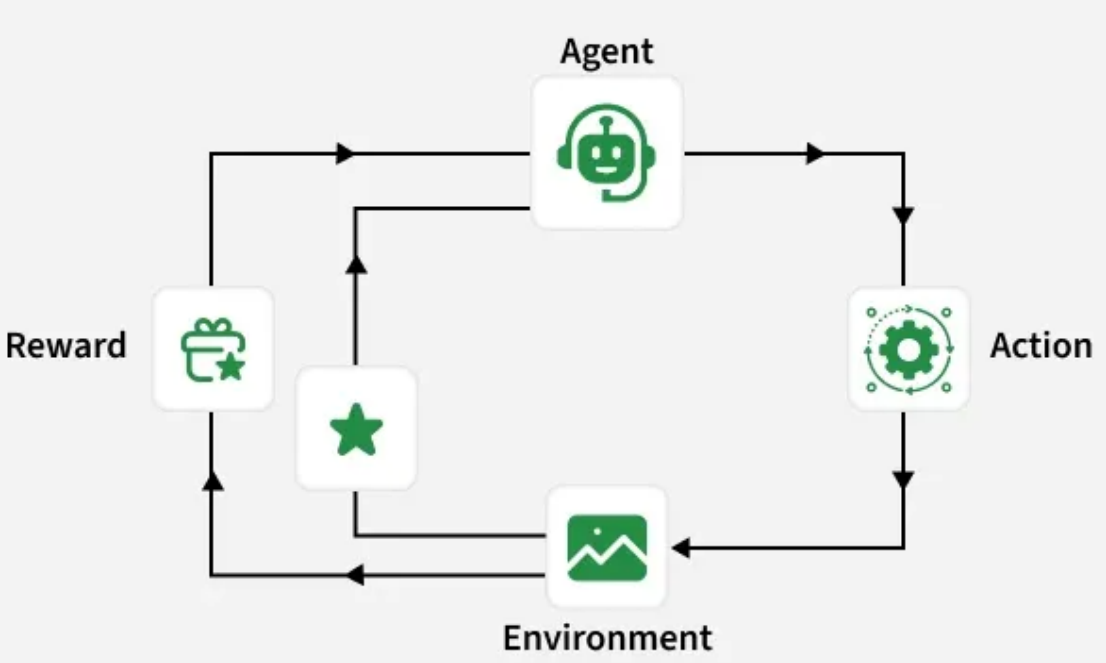
Despite its potential, Unsupervised Learning comes with its own set of challenges:

- **Interpretability:** The results of Unsupervised Learning can be difficult to interpret, especially in high-dimensional spaces.

- **Evaluation:** Unlike Supervised Learning, there is no straightforward way to evaluate the performance of an Unsupervised Learning model, as there are no labels to compare against.
- **Scalability:** Some algorithms, like hierarchical clustering, struggle to scale to very large datasets.
- **Choosing the Right Number of Clusters:** In clustering, determining the optimal number of clusters can be challenging and often requires domain knowledge or heuristic methods.

II.3 Reinforcement learning

Reinforcement learning interacts with environment and learn from them based on rewards.



Reinforcement Learning

1. Model-Based Methods

These methods use a model of the environment to predict outcomes and help the agent plan actions by simulating potential results.

- Markov decision processes (MDPs)
- Bellman equation
- Value iteration algorithm
- Monte Carlo Tree Search

2. Model-Free Methods

The agent learns directly from experience by interacting with the environment and adjusting its actions based on feedback.

- Q-Learning

- SARSA
- Monte Carlo Methods
- Reinforce Algorithm
- Actor-Critic Algorithm
- Asynchronous Advantage Actor-Critic (A3C)