Applied Statistic

Stefano Cecchettini

June 2025

Contents

1 Introduction

The purpose of this article is to resume all arguments seen during the applied statistics course held by Professor Mario Beraha during the academic year 24/25.

2 Basics statistic

Definition (Random Variable): A random variable is a variable that takes on numerical values based on the outcomes of a random event or experiment. Formally, if Ω is the sample space, then

$$X : \Omega \to \mathbb{R} \tag{1}$$

There are two different types of random variable: Discrete RVs and Continuous RVs.

2.1 Discrete case

For a discrete RV X, we can use the **probability mass function** (pmf) that is:

$$p_x(x) = \mathbb{P}(X = x), \sum_{x \in X} p_x(x) = 1 \tag{2}$$

Where $p_x(x)$ is the probability that X equals x. It can be:

1. **Bernoulli:**

$$p(x) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

2. **Binomial:**

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x \in \{0, 1, \ldots, n\} \tag{4}$$

Definition (Conditional expectation): is the expected (average) value of a random variable given that some condition is known to hold, such as the value of another variable.
It tells you: "What is the expected value of X, given that Y=y?"
In formulas:

$$\mathbb{E}[X \mid Y = y] \tag{5}$$

For discrete case is equal to:

$$\mathbb{E}[X \mid Y = y] = \sum_x x \cdot \mathbb{P}(X = x \mid Y = y) \tag{6}$$

or also written as:

$$p_{X|Y}(x \mid y) = \frac{p_{X,Y}(x, y)}{p_Y(y)} \tag{7}$$

2.2 Continuous case

Definition (Probability density function): A **PDF** describes the likelihood of a continuous random variable taking on a specific value. Unlike a probability mass function (used for discrete variables), the PDF does not give probabilities directly. Instead, the probability that a continuous random variable X falls within an interval $[a, b]$ is given by the area under the curve:

$$P(a \leq X \leq b) = \int_a^b f(x)\, dx \tag{8}$$

where $f(x)$ is the probability density function.
The function $f(x)$ must satisfy:

- $f(x) \geq 0$ for all $x \in \mathbb{R}$
- $\int_{-\infty}^{\infty} f(x)\, dx = 1$

For the continuous case, the Conditional p.d.f is:

$$f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x, y)}{f_Y(y)} \tag{9}$$

And the conditional expectation is:

$$\mathbb{E}[X \mid Y = y] = \int_{-\infty}^{\infty} x f_{X|Y}(x \mid y)\, dx \tag{10}$$

Example: Univariate Gaussian (Normal) Distribution

One of the most important and widely used PDFs is the **Normal distribution**, also known as the *Gaussian distribution*. Its probability density function is the following.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{11}$$

with $x \in \mathbb{X}$ and where:

- μ is the mean (expected value),
- σ^2 is the variance

The Normal distribution is fundamental because:

- Many natural phenomena (e.g., heights, test scores, measurement errors) are approximately normally distributed.
- It arises naturally via the **Central Limit Theorem**: the sum (or average) of many independent random variables tends toward a normal distribution, even if the original variables are not normally distributed.

2.3 Basic descriptive Statistics

Definition (Mean): the aritmetic average is:

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{12}$$

Definition (Variance): A measure of the spread of the data:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2 \tag{13}$$

Definition (Covariance): Measures how two variables vary together:

$$s_{XY} = \text{cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{14}$$

It can also be viewed as a matrix. However, if the features are on different scales, it is better to use the correlation matrix.

Definition (Correlation): Standardized covariance, which lies between [-1, 1]:

$$r_{XY} = \text{Cor}(X, Y) = \frac{\text{cov}(X, Y)}{s_X s_Y} \tag{15}$$

It can be Positive, None or Negative if data are represented like:

// TODO: inserisci figura correlazione da L2

For multivariate cases involving more than two variables, we often refer to the **correlation matrix**, which summarizes the pairwise linear relationships between all variables in a vector. The correlation matrix $\mathbf{R}$ is defined as:

$$\mathbf{R} = \begin{pmatrix} 1 & r_{12} & r_{13} & \cdots & r_{1d} \\ r_{21} & 1 & r_{23} & \cdots & r_{2d} \\ r_{31} & r_{32} & 1 & \cdots & r_{3d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{d1} & r_{d2} & r_{d3} & \cdots & 1 \end{pmatrix} \tag{16}$$

In general, r measure the strength of the linear association, i.e.:

1. $r = 0$: implies linear dependency

2. $r < 0$: implies the tendency of one value in the pair to be larger than its average when the other is smaller than its average.
3. $r > 0$: implies the tendency of one value in the pair to be large when the other values is large and also for both values to be small together.
4. $|r| = 1$: implies perfect correlation, negative or positively.

We can represent it with some graphics such as 3D plots, start plots or Chernoff faces.

Definition (Statistic test): A **statistical test** is a formal procedure used to decide whether to accept or reject a specific hypothesis about a population based on sample data. It involves:

- A **null hypothesis** H_0: a statement assumed to be true unless evidence suggests otherwise.
- An **alternative hypothesis** H_1: the statement we consider if there is sufficient evidence against H_0.
- A **test statistic**: a function of the sample data used to make a decision.
- A **rejection region** or **p-value**: criteria to determine whether to reject H_0.

The goal of a statistical test is to assess whether the observed data are consistent with the null hypothesis or provide enough evidence to favor the alternative.

Definition (Shapiro-Wilk Test):
The **Shapiro-Wilk test** is a statistical test used to assess whether a sample comes from a normally distributed population. It is especially suitable for small to moderate sample sizes.

- **Null hypothesis** H_0: The data are drawn from a normal distribution.
- **Alternative hypothesis** H_1: The data are not drawn from a normal distribution.

The test statistic is defined as:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where:

- $x_{(i)}$ are the ordered sample values,
- $\bar{x}$ is the sample mean,
- a_i are constants derived from the expected values of order statistics of a standard normal distribution.

A small value of W indicates a departure from normality. The decision is made by comparing the p-value to a significance level (e.g., $\alpha = 0.05$):

- If $p < \alpha$, reject H_0: the data are not normally distributed.
- If $p \geq \alpha$, do not reject H_0: no evidence against normality.

Definition (Bayes' Theorem):
Bayes' Theorem is a fundamental result in probability theory that describes how to update the probability of a hypothesis based on new evidence. Given two events A and B, with $\mathbb{P}(B) > 0$, Bayes' Theorem states:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A) \cdot \mathbb{P}(A)}{\mathbb{P}(B)}$$

where:

- $\mathbb{P}(A \mid B)$ is the **posterior** probability: the probability of event A given that B has occurred.
- $\mathbb{P}(B \mid A)$ is the **likelihood**: the probability of observing B if A is true.
- $\mathbb{P}(A)$ is the **prior** probability of A, before observing B.
- $\mathbb{P}(B)$ is the **marginal** probability of B, which acts as a normalizing constant.

Definition maximum likelihood estimation (MLE): The *maximum likelihood estimator (MLE)* of θ is the value $\hat{\theta}$ that maximizes the likelihood function:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta \mid x) \tag{17}$$

It is often more convenient to maximize the *log-likelihood*.

Definition (Likelihood function): the likelihood measures how well a statistical model explains observed data. It's a function of the model parameters, showing how likely those parameters are, given the data you observed.

Taken i.i.d. obervations $X_1, X_2, ..., X_n$ with density/mass function $f(x|\theta)$, we can write:

$$L(\theta \mid x) = \prod_{i=1}^{n} f(x_i \mid \theta) \tag{18}$$

You should use the Log-Likelihood $l(\theta)$ that is the natural logarithm of $L(\theta \mid x)$.

We can take two example using the bernoulli distribution and the gaussian one:

1. **Bernoulli:** Using $X_i \in \{0, 1\}$ with its $f(x_i \mid p) = p^{x_i}(1-p)^{1-x_i}$, the likelihood will be:

$$L(p \mid x) = p^{\sum x_i}(1-p)^{n-\sum x_i} \tag{19}$$

   And the Log-Likelihood will be:

$$\ell(p) = k \ln p + (n-k)\ln(1-p), \quad \text{where } k = \sum x_i \tag{20}$$

   The MLE, that is simply $\frac{d\leqslant}{dp} = 0$, gives the best value for p $\hat{p} = \frac{k}{n}$.

2. **Gaussian:** Using $X_i \in \{0, 1\}$ with its $f(x_i \mid \mu, \sigma^2)$. It gives a Log-Likelihood equal to:

$$\ell(\mu, \sigma^2) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{21}$$

   And the MLEs will be:

   (a) $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$

   (b) $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$

Definition (Multivariate Gaussian Distribution): The multivariate Gaussian distribution (also called the multivariate normal distribution) is a generalization of the one-dimensional (univariate) normal distribution to multiple variables.

It describes a vector of random variables that are jointly normally distributed, meaning any linear combination of them is normally distributed. A d-dimensional random vector $X = (X_1, ..., X_d)$ is defined multivariate normal if:

$$f(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{22}$$

Where μ is the mean vector and Σ is the covariance matrix.

Example for a 2D Gaussian Case for Multivariance Gaussian Distribution: Consider a 2-dim random vector:

$$\mu = \begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix} \tag{23}$$

Where ρ σ_X σ_Y σ_Y^2 is the covariance between X and Y with p correlation coefficient. Σ is of course symmetric. It has got different properties like:

1. Marginal distributions remain gaussian (i.e. the single X_i)
2. Linear Transformations: If $Y = AX + b$, then Y is gaussian
3. Conditional distributions: Conditioning on a subset of variables yields a Gaussian.

The application of Multivariate Gaussian are the linear discriminant analysis, covariance estimation, signal processing, gaussian mixture models (GMMs) and multivariate regression.

Definition (Statistical learning): Statistical learning is a field that combines statistics and machine learning to develop models that learn from data in order to make predictions or inferences. It also Builds models that learn from data, quantifies uncertainty in predictions and also balances model complexity and generalization.

3 Unsupervised Learning

Definition (Unsupervised Learning): Unsupervised learning involves analyzing data *without* a target variable. The goal is to uncover hidden patterns or structure within the data. Common techniques include:

- **Clustering** – grouping similar data points.
- **Dimensionality Reduction**(generalization) – reducing the number of variables (e.g., using Principal Component Analysis, PCA using euclidean spaces).
- **Association** - associate similat items that can goes together.

The goal of dimensionality reduction is to transform data from a high-dimensional, potentially complex space into a lower-dimensional Euclidean space, while preserving as much relevant information as possible. Specifically, we aim to summarize data described by p variables using a smaller set of k derived variables, where $k < p$. The challenge lies in finding the right balance between simplification and information retention—too much reduction can lead to a loss of important structure or variability in the data.

Example: Red Blood Cells Suppose we observe a set of red blood cells, and for each cell we measure two features: thickness and diameter. These data points can be visualized as a cloud of points in a two-dimensional space.

Using only the diameter to describe the cells would ignore thickness-related variability, and vice versa. Therefore, we seek a method that retains the **maximum variability** in the data while discarding redundant information. This means we want to:

- **Maximize variance:** capture the direction in which the data vary the most.
- **Minimize residual:** reduce the distance of the data points from the new axis of projection.

If a clear direction of data spread is identified, we can define this as the new horizontal axis (principal component). The second axis is defined as perpendicular to the first. We then project all data onto this new coordinate system.

3.0.1 PCA (Principal Component Analysis)

Definition: Principal Component Analysis (PCA) is a dimensionality reduction technique that seeks to explain the variance in a dataset by transforming the original variables into a new set of uncorrelated variables called *principal components*. Each principal component is a linear combination of the original variables and captures as much variance as possible.

Formally, the objectives of PCA are:

1. **Maximize variance**: find the direction in the data space along which the observations vary the most.
2. **Minimize residual**: reduce the loss of information by keeping directions that retain most of the variability.

PCA is particularly useful when the original variables in X are correlated. It decomposes the variance-covariance structure of X into orthogonal components (principal axes), and projects the data onto the subspace spanned by the top k components. The choice of k is a trade-off: it should be the smallest number of components that captures a sufficient amount of the total variance in the dataset.

Why is dimensionality reduction useful? When the number of variables p is very large, it becomes difficult to manage, visualize, and interpret the data. High-dimensional datasets often contain redundant or irrelevant information. PCA helps by reducing dimensionality while preserving the most important structure.

Definition (Loadings): The loadings are the coefficients (weights) of the linear combinations used to compute the principal components from the original variables. They define the new coordinate system in terms of the original variables.

Definition (Scores): The scores are the coordinates of the data points in the new system of principal axes. In other words, they represent the projection of the original data onto the principal components.

In matrix notation:

$$X = UV^\top$$

where:

- $X \in \mathbb{R}^{n \times p}$: original data matrix,
- $U \in \mathbb{R}^{n \times k}$: score matrix (projected data),
- $V \in \mathbb{R}^{p \times k}$: loadings matrix (principal axes).

Important properties of PCA:

- PCA is **rotation invariant**, meaning it is unaffected by rotation of the original coordinate system.
- The principal components are **uncorrelated** (orthogonal to each other).
- PCA can effectively handle **collinearity** in high-dimensional data.

In general we obtain:

$$PC_j = \phi_{1j}x_1 + \phi_{2j}x_2 + ... + \phi_{pj}x_p \tag{24}$$

And the score (the representation of data in the new dimension space, or projections along the principal components) i is equal to:

$$z_{ij} = \phi_{1j}x_{i1} + \phi_{2j}x_{i2} + ... + \phi_{pj}x_{ip} \tag{25}$$

The loadings are all ϕ_{ij}. In fact, the original variables are linearly combined and weighted according to the loadings to define the new coordinate system.

To obtain the loadings is sufficient to take the variance matrix, compute the eigenvalues and sort them. The first will be the first PC, the second one the second PC, etc.

Advice: When performing PCA, it is important to standardize the data, especially when features are measured on different scales. A common normalization condition is:

$$\sum_{i=1}^{p} \phi_{j1}^2 = 1$$

where ϕ_{j1} represents the loadings.

Assumption: We assume that the data have zero mean. This is achieved by subtracting the mean from each variable before applying PCA.

PCA can also be interpreted as an optimization problem.

Definition (Proportion of Variance Explained – PVE): The PVE is defined as:

$$\text{PVE}_k = \frac{\text{Var}_k}{\text{Var}_{\text{tot}}}$$

It represents the proportion of total variance explained by the k-th principal component.

Steps to Apply PCA:

1. Obtain an $n \times p$ data matrix.
2. Subtract the mean (or standardize the data).
3. Compute the principal components (PCs). The final number of PCs is equal to initial dimension or lower.
4. Choose the number of components k to retain (e.g., using the elbow criterion).
5. Project the data onto the k-dimensional subspace and proceed with the analysis.

Note: It may not always be beneficial to keep adding more components k, as the increase in explained variance diminishes. It is helpful to inspect the graph of the proportion of variance explained versus the number of principal components (the scree plot where you can see how each PCs are relevant to explain datas using their variation).

Definition (Biplot): Like a scatterplot, but with the representation of the original variables.

Probabilistic PCA: Assume that $x = Wz + \epsilon$ with W matrix $p * d$, $\epsilon \sim N(0, \sigma^2 I)$ and $z \sim N(0, l)$. It can be proved that the MLE for W is equivalent to the d-dimensional loadings matrix of PCA. It is usefull for generalization.

PCA via SVD of W: To perform PCA in a better way you can use SVD method on the centered data matrix rather than forming and diagonalizing the covariance matrix using $X = U\Sigma V^T$.

3.1 Clustering

In general, the term refers to a grouping of objects such that the objects in the same group (cluster) are similan to each other and at the same time different from objects belonging to other groups. The goal is to obtain an intra-cluster distance very low and inter-cluster distance maximized.

There are two types of clustering:

1. **Hierarchical clustering**: A set of nested clusters organized as a hierarchiral tree. It can be agglomerative or divisice
2. **Partitional clustering**: A division data objects into subsets such that each data object is in exactly one subset.

Definition Distance: define what makes an object "similar to" another. There are a lot of definitions such as:

1. Eucledian distance: Using pitagora theorem: $d_e(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 ....}$. It can be also standardized, i.e. weighed with the inverse of the variability on that dimension.
2. Manhattan distance: That is $d_M(x, y) = |x_1 - y_1| + |x_2 - y_2| ...$
3. Correlation distance: That is $d_R(x, y) = 1 - r_{xy}$ where r is the correlation between x and y.
4. Mahalanobis distance:
5. Minkowski distance:

The "right" distance metric is something that depends on the application. To verify the goodness of a clustering, it is possible to use numerical indices, which are typically grouped into 3 families:

1. External Metrics (EM): when an external label is provided. (ex precision, recall)
2. Internal Metrics (IM): no external infos (ex Sum of Squared Error)
3. Relative Metrics (RM): used to compare different clusterings with each other (ex SSE)

Using a specific notation like:

n = number of points

m_i = points in cluster i

c_j = points in class j

m_{ij} = points in cluster i coming from class j

$p_{ij} = \frac{m_{ij}}{m_i}$ frequency of element from class j in cluster i

Definition (Entropy)(EM): Let C_j be a cluster and let the dataset be associated with K ground truth classes. The entropy of cluster C_j is defined as:

$$\begin{aligned} \text{Entropy}(C_j) &= -\sum_{i=1}^{K} p_{ij} \log_2(p_{ij}) \\ \text{Entropy}_{\text{total}} &= \sum_{j=1}^{M} \frac{|m_j|}{n} \cdot \text{Entropy}(C_j) \end{aligned} \tag{26}$$

The second equation refers to the total entropy for the clustering solution that is the weighted average of entropies over all clusters where: M is the total number of clusters.

Definition (Purity)(EM): Purity measures how well the clusters contain only data points from a single class. There are:

$$\begin{aligned} \text{For a cluster i: } (p_i) &= \max_j p_{ij} \\ \text{For a clustering: } &= \sum_{i=1}^{M} \frac{m_i}{n} \cdot p_i \end{aligned} \tag{27}$$

Definition (Precision)(EM): for a clustering, is the max of p_{ij} for the cluster i respect to class j.

Definition (Recall)(EM): for a clustering, is the max of $Rec(i,j) = \frac{m_{ij}}{c_j}$. It tells us if we're creating redundant clusters.

Definition (F-measure)(EM): Harmonic mean of precision and recall. It is equal to: $F(i,j) = \frac{2*Prec(i,j)*Rec(i,j)}{Prec(i,j)+Rec(i,j)}$.

Definition (Sum of Squared Errors)(SSE)(IM): Sum of the squared distances between all observations in a cluster and the corresponding method. You can use it to choose the best number of clusters (k) looking at the graph SSE over k using the Elbow method.

Definition (Between Clusters Sum of Squared)(BCSE, also BSS)(IM): is a metric that represent the separation between clusters. With the formulation:

$$\sum_i m_i (c - c_i)^2 \tag{28}$$

where m_i is the size of the i-th cluster

Definition (Within Clusters Sum of Squared)(WCSE, also WSS)(IM): Cohesion of clusters. Given a dataset of n data points partitioned into K clusters $C_1, C_2, \ldots, C_K$, and with μ_j as the centroid of cluster C_j, the WCSS is defined as:

$$\text{WCSS} = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \tag{29}$$

With x_i is a data point in cluster C_j and μ_j is the centroid (mean) of cluster C_j.

Definition (Silhouette index)(IM): The silhouette index measures how similar a data point is to its own cluster compared to other clusters. For each data point x_i, the silhouette score $s(i)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{30}$$

where:

- $a(i)$ is the average distance from x_i to all other points in the same cluster,
- $b(i)$ is the minimum average distance from x_i to all points in the nearest neighboring cluster.

The silhouette score ranges from -1 to 1:

- $s(i) \approx 1$: the point is well-matched to its own cluster and poorly matched to others,
- $s(i) \approx 0$: the point lies on the border between clusters,
- $s(i) < 0$: the point may be assigned to the wrong cluster.

The overall silhouette index is the average of $s(i)$ over all data points:

$$\text{Silhouette Index} = \frac{1}{n} \sum_{i=1}^{n} s(i) \tag{31}$$

// TODO: add silhouette graph where each line is an observation

3.1.1 Hierarchical Clustering

Hierarchical clustering is a method of cluster analysis that seeks to build a hierarchy of clusters. Unlike partition-based methods such as k-means, hierarchical clustering does not require specifying the number of clusters in advance. Instead, it creates a tree-like structure called a **dendrogram**, which represents nested groupings of data points and their similarity levels.

// TODO: add dendrogram image

There are two main approaches:

- **Agglomerative (bottom-up):** Starts with each data point as its own cluster and iteratively merges the closest pairs of clusters.
- **Divisive (top-down):** Starts with all data points in a single cluster and recursively splits them into smaller clusters.

It uses a distance matrix (that contains the distance of every pair of observations) to implement groupings. Hierarchical clustering requires a stopping criterion.

The steps to create the dendrogram with agglomerate approach are:

1. Compute the proximity matrix
2. Let each data point to be a cluster
3. Repeat: Merge the two clotest clusters and update the proximity matrix using the new clusters. It depends on **Linkage method**, i.e. the method adopted for merging clusters, that deining the proximity among clusters.
4. Repeat until only a single cluster remains

Linkage methods: In hierarchical clustering, the choice of **linkage method** determines how the distance between clusters is calculated during the merging process. The linkage criterion directly influences the shape and structure of the resulting dendrogram.

- **Single Linkage (Minimum Linkage):** The distance between two clusters is defined as the shortest distance between any two points in the two clusters.

  $$D(A, B) = \min_{x \in A,\, y \in B} \|x - y\|$$

  It can handle non-spherical clusters but is sensitive to noise and outliers.
- **Complete Linkage (Maximum Linkage):** The distance between two clusters is the maximum distance between any two points in the clusters.

  $$D(A, B) = \max_{x \in A,\, y \in B} \|x - y\|$$

  This method tends to create compact and spherical clusters. It tends to break large clusters, but it is less susceptible to noise and outliers.
- **Average Linkage:** The distance between two clusters is the average of all pairwise distances between points in the two clusters.

  $$D(A, B) = \frac{1}{|A||B|} \sum_{x \in A} \sum_{y \in B} \|x - y\|$$

  It provides a balance between single and complete linkage.

With that algorithm there are great computational complexity and, after the merging of two clusters, the decision cannot be reversed. So, how to choose k (number of clusters)? Looking for stability areas in the dendrogram.

3.1.2 K-Means

K-means is a widely used partitional clustering algorithm that aims to divide a dataset into K non-overlapping, clusters by minimizing the variance within each cluster. Unlike hierarchical clustering, K-means requires the number of clusters K to be specified in advance.

Goal: Partition n data points into K clusters $C_1, C_2, \ldots, C_K$ such that the **WCSS** is minimized.

K-means Algorithm:

1. Random assignment of each observation to one of the K cluster and computation of centroid (mean point) of each cluster
2. Assign each observation to the cluster whose centroid is closer and update the partition matrix
3. Compute centroids again. Then repeat previous step until no point changes the cluster of belonging or until other stopping criteria are satisfied.

Properties and Notes:

- The algorithm is guaranteed to converge in a finite number of steps, but the solution may be a local minimum.
- It is sensitive to the initial choice of centroids; multiple runs or K-means++ initialization is recommended.
- It assumes clusters are convex and isotropic (e.g., spherical in Euclidean space).
- The number of clusters K can be chosen using evaluation metrics such as the Elbow Method, Silhouette Score, or Gap Statistic.
- Use **Medoids** (median points) instead of centroids..
- The use of Euclidean distance makes the algorithm sensitive to outliers.

K-means is simple and computationally efficient, making it suitable for large datasets when clusters are approximately spherical and well-separated.

Making a recap,tThe objective function of K-means clustering can be written as the total within-cluster sum of squared distances:

$$\sum_{k=1}^{K} \sum_{i \in C_k} \frac{1}{2\delta^2} \|x_i - \mu_k\|^2, \delta > 0$$

Minimizing this is equivalent (up to a constant) to maximizing the log-likelihood under a probabilistic model where each data point $x_i \in C_k$ is assumed to be drawn from a Gaussian distribution:

$$x_i \mid i \in C_k \sim \mathcal{N}(\mu_k, \delta^2 I)$$

Here, $\delta > 0$ is a constant and $\delta^2 I$ denotes isotropic Gaussian noise, which is the same for all clusters.

3.1.3 Latent Variable Representation

We introduce latent variables $z_1, \ldots, z_n$, where:

$$z_i \in \{0,1\}^K, \quad \text{with } z_i[ij] = 1 \text{ iff } x_i \in C_k$$

Example:

$$z_i = [0,0,1,0,0] = \pi_3 \text{ i.e. prior probability that j-th point belongs to cluster k}$$

The log-likelihood of the data, given the assignments, becomes:

$$\log p(X \mid Z) \propto -\sum_{k=1}^{K} \sum_{i \in C_k} \frac{1}{2\delta^2} \|x_i - \mu_k\|^2$$

This can be rewritten as:

$$\prod_{k=1}^{K} \prod_{i \in C_k} \mathcal{N}(x_i \mid \mu_k, \delta^2 I) = \prod_{i=1}^{n} \prod_{k=1}^{K} \mathcal{N}(x_i \mid \mu_k, \delta^2 I)^{z_i[k]} = p(x|z)$$

To generalize, we define the conditional distribution of data given latent variable z:

$$p(x \mid z) := \text{“conditional distribution of data } X \text{ given } z''$$

We also define a prior distribution over z:

$$p(z) = \prod_{i=1}^{n} p(z_i) = \prod_{i=1}^{n} \prod_{k=1}^{K} \pi_k^{z_i[k]}$$

where π_k is the prior probability that a point belongs to cluster k. This leads to the **marginal distribution**:

$$p(x) = \sum_{\text{all possible } z} p(x \mid z)p(z)$$

which corresponds to marginalizing over the latent distribution.
The complete likelihood of the observed data under the GMM (gaussian mixture model) is:

$$p(x_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

This model is known as the **Gaussian Mixture Model** (GMM).

Each cluster is modeled as:

$$\mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

If we assume that $\Sigma_k = \delta^2 I$, the clusters are isotropic (spherical), as shown in the circular contour plot.
The full parameter set of a GMM is:

$$\Theta = (\pi, \mu, \Sigma)$$

3.1.4 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model is a probabilistic model for representing normally distributed subpopulations within an overall population. It assumes that each data point x_i is generated by one of K clusters, each following a multivariate Gaussian distribution.

Model Definition. Let $x_1, \ldots, x_n \overset{\text{iid}}{\sim} p(x)$, with:

$$p(x) = \prod_{i=1}^{n} \{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \}$$

where:

1. $\pi_k \geq 0$ are the mixture weights (or prior probabilities),
2. $\sum_{k=1}^{K} \pi_k = 1$,
3. $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ is the Gaussian density with mean μ_k and covariance Σ_k

The overall multiplication becames from indipendence.

Joint Distribution. The full joint distribution over data and cluster assignments is:

$$p(x, z) = \prod_{i=1}^{n} \prod_{k=1}^{K} \left[\mathcal{N}(x_i \mid \mu_k, \Sigma_k)\pi_k \right]^{z_i[k]}.$$

Posterior Probability (Responsibilities). We can compute the posterior probability (or responsibility) of cluster k given x_i:

$$\gamma_{i,k} = p(z_i[k] = 1 \mid x_i) = \frac{p(z[k] = 1)p(x_i | z_i[k] = 1)}{\sum_{j=1}^{K} p(z_j[k] = 1)p(x_i | z_j[k] = 1)} = \text{ using bayes theorem } = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}.$$

This corresponds to the probability that x_i belongs to cluster k.

Parameter Estimation via EM Algorithm. We aim to find the parameters $\theta = (\pi, \mu, \Sigma)$ that maximize the likelihood of the data (i.e. $p(x|\pi, \mu, \Sigma)$). Direct maximization of the log-likelihood:

$$\log p(x \mid \theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

is intractable due to the log-sum. Instead, we use the Expectation-Maximization (EM) algorithm:

E-step: Compute the responsibilities $\gamma_{i,k}$.

M-step: Update parameters:

$$\mu_k^{\text{new}} = \frac{1}{\sum_{i=1}^{n} \gamma_{i,k}} \sum_{i=1}^{n} \gamma_{i,k} x_i,$$

$$\Sigma_k^{\text{new}} = \frac{1}{\sum_{i=1}^{n} \gamma_{i,k}} \sum_{i=1}^{n} \gamma_{i,k} (x_i - \mu_k^{\text{new}})(x_i - \mu_k^{\text{new}})^\top,$$

$$\pi_k^{\text{new}} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{i,k}.$$

Initialization and Convergence. To obtain something, you have to:

1. Initialize $\theta^{(0)}$.
2. Alternate E and M steps until convergence (based on likelihood or parameter stability).
3. Each iteration increases the likelihood (or keeps it the same).

Relation to Soft Clustering. Unlike k-means, where each point belongs to a single cluster, GMM allows *soft clustering*, where each point is assigned to clusters with probabilities (i.e., $\gamma_{i,k}$):

$$p(z \mid x) = \frac{p(x, z)}{p(x)} \propto p(x \mid z) p(z)$$

Given the form:

$$p(x_i) = \sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)$$

we can compute the posterior for each component:

$$\pi_i^*[k] = p(z_i = k \mid x_i)$$

These values $\pi_i^*[k]$ represent the **responsibilities**, i.e. the soft assignments of data points to clusters.

Note on Degeneracy. If we set $\mu_k = x_i$ and $\Sigma_k = \delta^2 I$ with $\delta \to 0$, the likelihood becomes arbitrarily large. This leads to overfitting, and is why we need the EM method instead of naive MLE.

4 Supervised learning

Definition (Supervised Learning): Supervised learning is the process of learning a mapping from input variables to a known output (target variable). The target can be:

- **Continuous** – in which case the task is a *regression* problem.
- **Categorical** – in which case the task is a *classification* problem.

The learning process is really simple. It starts training the model (find relationship between inputs and target using techniques like linear regression, decision trees...) and after that evaluate the model (using for example cross validation). A representation of the technique is in 1.

Mathematically, we introduce a vector $\underline{x} \in \mathbb{X}$ that is $\underline{x} = [x_1, x_2, \ldots, x_n]$, the features and another vector $\underline{y} \in \mathbb{Y}$ that is $\underline{y} = [y_1, y_2, \ldots, y_n]$,the target.

Goal: learn how to compose f in $f : \mathbb{X} \to \mathbb{Y}$

We can do this in two different way depending on the spaces of y and x:

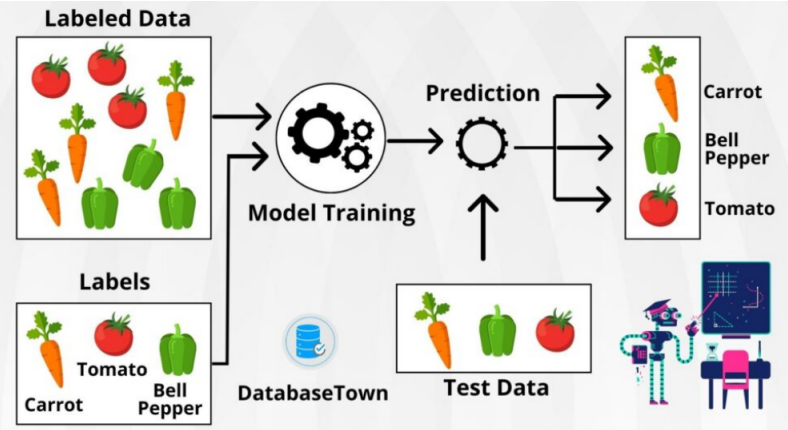


Figure 1: Supervised learning algorithm

1. **Regression problem**: If $y \in \mathbb{R}$ and $x \in \mathbb{R}^{\text{l}}$.
2. **Classification problem**: If y a finite set like $y = \{L_1, \ldots, L_g\}$, labels (similar to slustering, but here you already know the cluster)

Recall that in a Gaussian Mixture Model (GMM), the latent variable z_i represents the cluster assignment for sample x_i. In supervised classification, we now have pairs of observed data $(x_1, y_1), \ldots, (x_n, y_n)$, with each y_i implicitly related to some latent cluster z_i. The goal is to predict the labels $y_{n+1}, y_{n+2}, \ldots$ for new, unseen inputs $x_{n+1}, x_{n+2}, \ldots$.

To do this, we use the training data (x_i, y_i) to estimate the parameters of the data distribution for each class. Since the class label z_i is known for $i = 1, \ldots, n$, we can estimate the class-conditional parameters by maximizing the likelihood:

$$\hat{\theta}_k = \arg\max_{\theta} p\big(x_i : y_i = L_k \mid \theta_k\big)$$

where θ_k are the parameters specific to class k, such as the mean μ_k and covariance matrix Σ_k of a Gaussian distribution. Thus, the data belonging to class k are assumed to follow the distribution $p(x \mid \hat{\theta}_k)$.

Once we have estimated all $\hat{\theta}_1, \ldots, \hat{\theta}_g$, we can predict the label of a new data point x_{n+1}. Recalling Step 1 of the EM algorithm, we compute the posterior probability of each class using Bayes' theorem:

$$p(y_{n+1} = L_j \mid x_{n+1}) = \frac{p(x_{n+1} \mid \hat{\theta}_j)\, \pi_j}{\sum_{k=1}^{g} p(x_{n+1} \mid \hat{\theta}_k)\, \pi_k} \tag{32}$$

where π_j is the prior probability of class j, estimated as:

$$\pi_j = \frac{\text{number of observations in class } j}{n}$$

It is important to note that this formulation assumes a large amount of training data relative to the number of new data points. This naturally leads us into the domain of **decision theory**.

4.0.1 Formulation as a Decision Theory Problem

For simplicity, consider the case where $g = 2$. The goal is to find a decision function:

$$f : \mathcal{X} \to \{1, 2\}$$

that minimizes the expected cost (or *expected loss*) of misclassification.

Definition (Cost Function): A cost function quantifies the penalty for incorrect predictions. It maps model decisions to a real-valued loss. The notation $c_{a,b}$ denotes the cost of predicting class a when the true class is b.

We define the expected cost of a classifier f as:

$$\begin{aligned} EC(f) = \int_{R_2} c_{2,1}\, P_1(x)\pi_1\, dx + \int_{R_1} c_{1,2}\, P_2(x)\pi_2\, dx \\ \text{where } \mathcal{X} = R_1 \cup R_2, \quad R_j = \{x : f(x) = j\} \end{aligned} \tag{33}$$

Here, $P_i(x)$ is the probability density function of class i, and R_j is the region of input space assigned to class j.

To minimize $EC(f)$, we proceed in two steps:

1. Optimize with respect to R_1 and R_2. It can be shown that the optimal decision rule is:

$$R_1 = \{x \in \mathcal{X} : c_{1,2}P_2(x)\pi_2 \leq c_{2,1}P_1(x)\pi_1\}, \quad R_2 = \mathcal{X} \setminus R_1$$

2. The optimal classifier is:

$$f_{\text{optimal}}(x) = \begin{cases} 1 & \text{if } x \in R_1 \\ 2 & \text{if } x \in R_2 \end{cases}$$

4.0.2 Special Cases: QDA and LDA

If we assume that the class-conditional densities are Gaussian, i.e.,

$$P_i(x) = \mathcal{N}(x \mid \mu_i, \Sigma_i)$$

and that the costs are symmetric ($c_{2,1} = c_{1,2}$), then the decision rule reduces to:

$$f(x) = \arg\max_j \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)$$

This method is called **Quadratic Discriminant Analysis (QDA)**, as the decision boundary is defined by a quadratic surface due to differing covariances Σ_j.

If we further assume that all classes share the same covariance matrix, i.e., $\Sigma_i = \Sigma_j = \Sigma$, the decision boundaries become linear, and the resulting method is known as **Linear Discriminant Analysis (LDA)**.

// TODO: ADD figure for QDA and LDA

4.1 Classification

Classification is one of the fundamental tasks in supervised learning, where the objective is to assign input data points to one of a set of predefined categories or classes. Formally, given a dataset of n labeled examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where each $x_i \in \mathbb{R}^p$ represents a feature vector and $y_i \in \{L_1, \ldots, L_g\}$ is the corresponding class label, the goal is to learn a function (the classification function):

$$f : \mathbb{R}^p \to \{L_1, \ldots, L_g\}$$

that can accurately predict the label y for new, unseen input data x. Note that: $R_i \cap R_j = \emptyset$ and $\bigcup_{i=1}^g R_i = R^p$. Infact, $R_i = f(\{i\})^{-1} = \{x \in R^p : f(x) = i\}$.

Learn f ideal setting: we know the per-class distribution of the features $x|y = i \sim f_i$ and we know some "prior" probabilities π_j with $j = 1, \ldots, g$ and $P(y = j) = \pi_j$. By Bayes:

$$P(y = j|x) = \frac{P(x|y = j)P(y = j)}{P(x)} = \text{law of total prob.} = \frac{P_j(x)\pi_j}{\sum_{l=1}^g P_l\pi_l} \tag{34}$$

Definition (Bayes classifier): The **Bayes classifier** is a probabilistic classification rule that assigns each input $x \in \mathbb{R}^p$ to the class with the highest **posterior probability** given the input. Formally, for a classification problem with classes $\{L_1, L_2, \ldots, L_g\}$, the Bayes classifier is defined as:

$$f_{\text{Bayes}}(x) = \arg\max_{k \in \{1,\ldots,g\}} \mathbb{P}(Y = L_k \mid X = x)$$

Using Bayes' Theorem, the posterior can be written as:

$$\mathbb{P}(Y = L_k \mid X = x) = \frac{p(x \mid Y = L_k) \cdot \pi_k}{\sum_{j=1}^g p(x \mid Y = L_j) \cdot \pi_j}$$

where:

- $p(x \mid Y = L_k)$ is the class-conditional density for class L_k,
- $\pi_k = \mathbb{P}(Y = L_k)$ is the prior probability of class L_k,
- The denominator is the total marginal likelihood of x and serves as a normalization factor.

The Bayes classifier is optimal in the sense that it minimizes the probability of misclassification—called the **Bayes error rate**—under the assumption that the true distributions $p(x \mid Y)$ and priors π_k are known.

4.1.1 Incorporating Costs into Classification

In many real-world scenarios, misclassification errors do not all carry the same cost. For example, in medical diagnosis (with $g = 2$):

- Class 1: healthy (no disease)
- Class 2: ill (has disease)

In this case:

- Predicting "healthy" when the person is ill (**false negative**) may be dangerous and costly.
- Predicting "ill" when the person is healthy (**false positive**) may cause unnecessary stress or tests, but is usually less costly.
- Therefore: $C_{12} \gg C_{21}$

We aim to define a decision rule $f(x)$ that minimizes the **expected cost of classification errors**.

Definition (Cost Function): The cost function C_{ij} represents the penalty of assigning an observation to class j when the true class is i. Usually:

- $C_{ii} = 0$: no cost for correct classification,
- $C_{ij} > 0$: cost for misclassifying class i as class j,
- Often $C_{ij} \neq C_{ji}$: the cost is not symmetric.

Assume $g = 2$ classes and define decision regions R_1, R_2 where:

$$R_1 = \{x : f(x) = 1\}, \quad R_2 = \{x : f(x) = 2\}$$

The total expected cost is:

$$\begin{aligned} EC(f) &= C_{21} \int_{R_2} P_1(x)\pi_1 \, dx + C_{12} \int_{R_1} P_2(x)\pi_2 \, dx \\ &= C_{21}\pi_1 + \int_{R_1} [C_{12}\pi_2 P_2(x) - C_{21}\pi_1 P_1(x)] \, dx \end{aligned} \tag{35}$$

To minimize this cost, define:

$$g(x) = C_{12}\pi_2 P_2(x) - C_{21}\pi_1 P_1(x)$$

and choose:

$$R_1^* = \{x \in \mathbb{R}^p : g(x) < 0\} \quad \Rightarrow \quad f(x) = \begin{cases} 1 & \text{if } x \in R_1^* \\ 2 & \text{otherwise} \end{cases}$$

In practice, we do not know the true class distributions, so we estimate them. A common assumption is that the data are normally distributed within each class. You can use QDA or LDA.

4.1.2 Bayes Error Rate and Evaluation Metrics

The theoretical minimum error rate (Bayes error rate) is:

$$\text{AER (Actual Error Rate)}(f^*) = \sum_{j=1}^{g} \sum_{k \neq j} \int_{R_k} P_1(x)\pi_1 \, dx$$

But we can also use another metric that is an empirical approximation of misclassification rate:

$$\text{APER} = \frac{1}{n} \sum_{j=1}^{g} \sum_{k \neq j} n_{ik}$$

where n_{ik} is the number of samples from class i predicted as class k. APER is a simple but often noisy estimator of true error.

4.1.3 Confusion Matrix and Performance Metrics (for $g = 2$)

// TODO: add a section for the confusion matrix

Note that $n_{i,j}$ is the number of samples of class i attributed to class j by the classifier. There are some other important metrics coming from the confusion matrix that are:

- **Definition (Precision)**: $\frac{n_{11}}{n_{11}+n_{21}} = \frac{\text{TP}}{\text{TP+FP}}$
- **Definition (Recall)**: $\frac{n_{11}}{n_{11}+n_{12}} = \frac{\text{TP}}{\text{TP+FN}}$
- **Definition (F1-score)**:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

4.2 Regression

Regression is a fundamental task in supervised learning where the objective is to model the relationship between a set of input variables and a continuous output variable. Given a dataset of input-output pairs $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the goal is to learn a function:

$$f : \mathbb{R}^p \to \mathbb{R}$$

such that the output can be approximated by:

$$y_i = f(x_i) + \epsilon_i$$

Here, f captures the systematic relationship between inputs and outputs, and ϵ_i represents random noise or unexplained variability, typically assumed to be independent with mean zero.

The learned function $\hat{f}$ can then be used to predict outcomes for new, unseen inputs. Regression is widely used in applications such as forecasting, risk modeling, and scientific research, where understanding and predicting continuous outcomes is essential.

The regression function f can be expressed as a linear combination of fixed basis functions:

$$f(\underline{x}) = \sum_{j=1}^{q} \beta_j g_j(\underline{x}) \tag{36}$$

where:

- $g_j(\underline{x})$ are known basis functions (e.g., polynomials, splines),
- β_j are unknown coefficients to be estimated from the data.

This formulation allows for great flexibility. The input dimension p (number of features) can be any value. For instance, if $p = 2$, the model will consider two input variables. To keep the notation simple, let us now consider the case where $p = 1$, i.e., univariate regression.

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_q x^q$$

This is an example of **polynomial regression**, where each $g_j(x) = x^j$. In this setup, we may preprocess the inputs by computing transformed features:

$$x_i \to (z_{i1}, z_{i2}, \ldots, z_{iq}), \quad \text{where } z_{ij} = g_j(x_i)$$

So the regression problem becomes:

$$y_i = \sum_{j=1}^{q} \beta_j z_{ij} + \epsilon_i$$

4.2.1 Ordinary Least Squares

Definition (Ordinary Least Squares) (OLS): is the most common method to estimate the coefficients β_j by minimizing the total squared error between the observed values and the model predictions.

Given a dataset $\{(x_i, y_i)\}_{i=1}^n$, and the transformed input matrix $Z \in \mathbb{R}^{n \times q}$ with $z_{ij} = g_j(x_i)$, the OLS objective is:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{q} \beta_j z_{ij} \right)^2$$

In matrix form:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - Z\boldsymbol{\beta}\|^2$$

The closed-form solution is given by:

$$\hat{\boldsymbol{\beta}} = (Z^\top Z)^{-1} Z^\top \mathbf{y}$$

Assumptions of OLS:

- Linearity: The relationship between features and target is linear in the parameters. Model is correct
- Independence: Errors ϵ_i are independent, i.e. $\epsilon_i \perp \epsilon_j$ and $E[\epsilon_i] = 0$
- Homoscedasticity: Constant variance of errors, i.e. $VAR[\epsilon_i] = \sigma^2$.
- Normality (optional for inference): Errors are normally distributed, i.e. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

OLS is efficient and interpretable, and serves as the foundation for more advanced regression techniques such as ridge regression, LASSO, and generalized linear models.

Using those properties, we can say:

$$E[\hat{\beta}] = \beta$$

So $\hat{\beta}$ is the unbiases estimator for β. From now on we're considering only examples with $p = 1$ for simplicity (a simple linear regression for $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$).

$$\begin{aligned} VAR[\hat{\beta_1}] &= \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})} \text{ with } \bar{x} \text{ mean} \\ VAR[\hat{\beta_0}] &= \sigma^2 (\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})}) \\ \hat{\sigma^2} &= \frac{1}{n-2} \sum_{i=1}^{n} (y_i - (\hat{\beta_0} + \hat{\beta_1} x_i))^2 \end{aligned} \tag{37}$$

Where $\hat{\sigma^2}$ is the estimator for the unknown σ^2. We're doing this to obtain a valid start for our statistical test about $H_0 : \beta_j = 0$ while $H_1 : \beta_j \neq 0$. (... demonstration from notes...).

This is the test for the specific case of linear regression (p=1), but it can also be more general. This test can be optain at the end testing only the gaussianity of β, done with shapiro-wilk normality test (you can look only at the p-value of that test that has to be greater than 0.05 to NOT reject H_0).

The value of $\bar{x}$ we've seeing that determine the confidence interval for our test.

We are interested in the $(1 - \alpha)$-level confidence interval for the expected value of a new response Y^* given a fixed value x^*:

$$CI_{1-\alpha} \text{ for } \mathbb{E}[Y^* \mid X = x^*]$$

For simple linear regression ($p = 1$), where:

$$Y^* \mid X = x^* = f(x^*) + \epsilon^*, \quad \epsilon^* \sim \mathcal{N}(0, \sigma^2)$$
$$f(x^*) = \beta_0 + \beta_1 x^*$$

The confidence interval becomes:

$$CI_{1-\alpha}\left(\mathbb{E}[Y \mid X = x^*]\right) = \hat{\beta}_0 + \hat{\beta}_1 x^* \pm t_{1-\alpha/2}^{(n-2)} \cdot \hat{\sigma} \cdot \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

This confidence interval is **narrowest** when $x^* = \bar{x}$, i.e., when the new input is near the center of the observed data.

We can also consider the prediction interval (larger that CI) that will be:

$$PI_{1-\alpha}\left(Y \mid X = x^*\right) = \hat{\beta}_0 + \hat{\beta}_1 x^* \pm t_{1-\alpha/2}^{(n-2)} \cdot \hat{\sigma} \cdot \sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

that represent the accurate prediction when values of covariance are nearby values of covariance already seen.

4.2.2 Goodness of Fit of the Regression Model

Considering the general regression problem:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \epsilon$$

In which β_j is what I expect value of y to change if I maintain all x_j, except changing just x_j by 1.

A measure for the accuracy of a regression model that allows comparison between different models is the **coefficient of determination**, denoted as R^2. It quantifies how much of the total variability in the response variable is explained by the model.

Let:

- y_i: observed values,
- $\hat{y}_i$: predicted values from the model,
- $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$: mean of the observed values.

We can decompose the total variability of the data using the following identity:

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Which can be rewritten as:

$$\text{TSS} = \text{SS}_{\text{Reg}} + \text{RSS}$$

where:

- **TSS** (Total Sum of Squares): total variability present in the data,
- $\mathbf{SS_{Reg}}$ (Regression Sum of Squares): variability explained (captured) by the model,
- **RSS** (Residual Sum of Squares): variability that the model fails to explain.

This decomposition is based on the Pythagorean theorem (it doesn't depend on probabilistic assumptions).

Definition (R^2):

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} \in [0, 1]$$

- $R^2 = 1$: The model explains all the variability in the data (perfect fit).
- $R^2 = 0$: The model explains none of the variability (only residuals are left).

Interpretation: R^2 measures the proportion of variance in the dependent variable that is predictable from the independent variables. It is a fundamental metric for assessing the *goodness of fit* of a regression model.

4.2.3 Categorical variables

In regression analysis, input variables X_j may be **categorical**, such as gender, ethnicity, or age group. Since regression models operate on numerical inputs, categorical variables must be converted to a numerical format using a process called **dummy coding** (also known as one-hot encoding). For a binary categorical variable (e.g., "young or not", "citizen or not"), we can represent the categories using a single dummy variable. For example:

$$x_j = \begin{cases} 1 & \text{if person not young} \\ 0 & \text{otherwise} \end{cases}$$

This single dummy variable captures the presence or absence of the category.

More in general, for a categorical variable with K categories (e.g., ethnicity: Caucasian, African American, Asian, etc.), we create $K-1$ dummy variables. The process will be:

- Select one category as the **reference category** (e.g., African American).
- Create dummy variables $D_1, D_2, \ldots, D_{K-1}$ for the remaining categories.

The regression model becomes:

$$Y = \beta_0 + \beta_1 D_1 + \beta_2 D_2 + \cdots + \beta_{K-1} D_{K-1} + \varepsilon$$

- Each β_j measures the effect of category j relative to the reference. - **Important:** Do not include a dummy variable for the reference category! Its effect is implicitly captured in the intercept β_0.
Example: If ethnicity has 3 categories and "African American" is the reference, then:

- β_1: effect of being Caucasian vs. African American
- β_2: effect of being Hispanic vs. African American

This approach ensures that the model is identifiable and avoids perfect collinearity.

Interaction terms: Interaction terms allow the model to capture relationships between variables that are not purely additive. They increase model flexibility by allowing the effect of one variable to depend on the level of another.

Example:

If X_1 is categorical and X_2 is continuous, you can include interaction terms:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \cdot X_2) + \varepsilon$$

More generally, when using dummy variables D_k for the categorical variable, the interaction terms can be written as:

$$X_2 \cdot D_k \Rightarrow \text{Include terms like } X_2 \cdot D_1, X_2 \cdot D_2, \ldots$$

This lets the slope for X_2 differ across categories of X_1. For each change in a coefficient β, you get a different p-value. This allows you to make statistical inference or regression on any sub-model.
—

Collinearity issue
Although we can include as many variables and interactions as we want, this can lead to problems such as **collinearity**.

Collinearity Definition: Collinearity occurs when two or more predictor variables convey very similar information. For example:

$$X_2 \sim X_3 \quad \text{(or very close)}$$

Perfect Collinearity: we're gonna have perfect collinearity if:

$$X_j = \alpha_0 \sum_{l \neq j} \alpha_l x_j$$

i.e., X_j is a linear combination of other variables, then:

$$X^\top X \text{ is not invertible} \quad \text{(the design matrix is rank-deficient)}$$

this brings different problems like:

- Difficult to interpret the effect of individual coefficients.
- The variance of $\hat{\beta}_j$ increases under normality hp (4-th hypotesis of OLS)):

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2), \quad \Rightarrow \quad \hat{\beta}_j \sim \mathcal{N}(\beta_j, \text{Var}(\hat{\beta}_j))$$

- Variance of $\hat{\beta}_j$:

$$\text{Var}(\hat{\beta}_j) = \sigma^2 \left[(X^\top X)^{-1}\right]_{jj}$$

  This variance explodes in presence of collinearity.

A possible solution is the use of VIF

Variance Inflation Factor (VIF): To measure collinearity, we define the Variance Inflation Factor:

$$\text{VIF}(\hat{\beta}_j) = \frac{1}{1 - R_j^2}$$

Where R_j^2 is the R^2 value from regressing X_j on all other predictors:

$$X_j = \alpha_0 + \sum_{k \neq j} \alpha_k X_k + \varepsilon$$

We should use it in:

$$\text{Var}(\hat{\beta}_j) = \frac{\sigma^2}{\sum_{i=1}^n (x_{ij} - \bar{x_j})^2} * \text{VIF}$$

Rule of Thumb: If $\text{VIF}(\hat{\beta}_j) > 5$, there is a potential collinearity issue. It's a bad idea to include linearly dependent variables in your model. For this main reason, we have to consider to use some solution:

1. Dimensionality reduction in X using PCA
2. Variable selection
3. Schrinkage methods

4.2.4 Variable selection process

In many regression problems, we are given a response variable y and a set of potential predictors $X_1, X_2, \ldots, X_p$. The goal of **variable selection** is to find a subset of predictors $\{Z_1, Z_2, \ldots, Z_k\} \subseteq \{X_1, X_2, \ldots, X_p\}$ and fit a smaller model:

$$y = \beta_0 + \beta_1 Z_1 + \cdots + \beta_k Z_k + \varepsilon,$$

with $k < p$.

There are several strategies for variable selection:

- **Brute force:** Examine all possible subsets of the p variables (which are 2^p combinations), fit each model, and choose the best one. While exhaustive, this method is computationally expensive for large p.
- **Forward selection:** Start from the *empty model* $y = \beta_0 + \varepsilon$. For each variable X_j, fit the model:

$$y = \beta_0 + \beta_j X_j + \varepsilon,$$

  and choose the variable that most improves the model (e.g., lowest residual sum of squares). Denote this first selected variable as X_{j_1}. Then, for each remaining variable X_j, fit:

$$y = \beta_0 + \beta_{j_1} X_{j_1} + \beta_j X_j + \varepsilon,$$

  and again select the one that adds the most explanatory power. Continue this process until a stopping criterion is met (e.g., performance plateaus or some metric threshold is satisfied). This method is simple, intuitive, and adds variables one by one.
- **Backward selection:** This approach starts from the *full model* (i.e., all p predictors included) and removes variables one at a time. At each step, the variable contributing the least is removed, and the process is repeated until only significant variables remain.

4.2.5 Model Comparison Metrics.

Once models are selected, we need a way to compare them. One useful metric is the coefficient of determination, R^2, which measures the proportion of variance explained by the model. However, R^2 always increases as more variables are added, even if they are not useful. Therefore, we need better methods:

- **Adjusted** R^2: Denoted R^2_{adj}, this adjusts for the number of predictors p and data points n:

$$R^2_{\text{adj}} = 1 - (1 - R^2)\frac{n-1}{n-p-1}.$$

  This penalizes the addition of irrelevant variables and provides a more accurate comparison between models with different numbers of covariates.

- **Mallow's** C_p: Another useful metric defined as:

$$C_p = \frac{1}{n}\left(\text{RSS} + 2k\hat{\sigma}^2\right),$$

  where k is the number of parameters in the model, RSS is the residual sum of squares, and $\hat{\sigma}^2$ is an estimate of the error variance.

- **AIC** estimates the relative quality of a statistical model for a given set of data.

$$\text{AIC} = 2k - \log \mathcal{L} \tag{38}$$

  where $\mathcal{L}$ is the likelihood of the model and k is the number of estimated parameters. Lower AIC values indicate a better model — balancing fit and complexity. However, it does not give an absolute measure of model quality, only relative comparisons. It penalizes the number of parameters linearly, so it is more forgiving in allowing extra variables than BIC. AIC is commonly used when the goal is prediction and you want to avoid underfitting.

- **BIC:** Like AIC, BIC is used to compare models but includes a stronger penalty for complexity.

$$\text{BIC} = k\log(n) - 2\log \mathcal{L}, \tag{39}$$

  Also prefers lower values. It heavily penalizes more complex models, especially when the dataset is large. The penalty term grows with $\log(n)\log(n)$, so BIC tends to favor simpler models as n increases. More conservative than AIC. It's often preferred when model interpretability or parsimony is critical.

In general, one can use plots of metrics such as R^2 or R^2_{adj} against the number of covariates to find an "elbow" point, indicating the optimal number of predictors. While such plots provide useful intuition (as in k-means), it is often preferable to use formal criteria like AIC, BIC, or C_p for automated and objective model selection.

4.2.6 Shrinkage methods

Shrinkage methods are used in linear regression when we have a large number of features, and we want to prevent overfitting or perform variable selection. These methods work by constraining or regularizing the coefficients, effectively shrinking them toward zero. The two most common shrinkage methods are **Ridge Regression** and **Lasso Regression**.

Standard Linear Regression Given data $y_i \sim \mathcal{N}(x_i^\top \beta, \sigma^2)$, the ordinary least squares (OLS) estimate minimizes the residual sum of squares:

$$\hat{\beta}^{\text{OLS}} = \arg\min_\beta \sum_{i=1}^n (y_i - x_i^\top \beta)^2$$

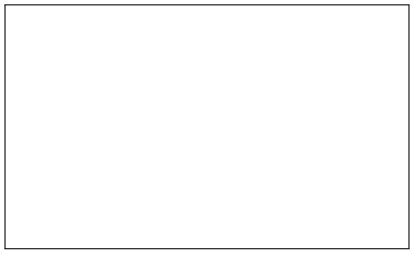
Ridge Regression Ridge adds an ℓ_2 penalty on the size of the coefficients:

$$\hat{\beta}^{\text{ridge}} = \arg\min_\beta \left\{ \sum_{i=1}^n (y_i - x_i^\top \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

This shrinks the coefficients, but never sets them exactly to zero. It is good for handling multicollinearity. It's good if λ is high, not if it is low.

Standardized Version: Before applying ridge, standardize the predictors so each has mean 0 and variance 1.

Effect of increasing λ on ridge regression:

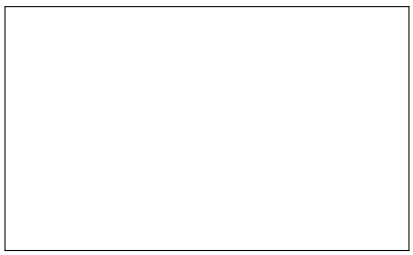


Lasso Regression Lasso adds an ℓ_1 penalty:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

This both shrinks coefficients and sets some exactly to zero, performing variable selection.

Effect of increasing λ on lasso regression:



The Effect of Shrinkage We define the **shrinkage factor**:

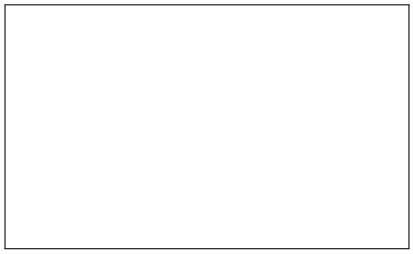
$$s_\lambda = \frac{\text{Var}(\hat{\beta}_j^{\text{Ridge}})}{\text{Var}(\hat{\beta}_j^{\text{OLS}})} \in (0, 1)$$

As λ increases, the variance decreases, but bias increases. The goal is to find a good bias-variance tradeoff.

Geometric Interpretation Shrinkage methods can also be interpreted geometrically:

1. Ridge: constraint region is a circle (or sphere in high dimensions).
2. Lasso: constraint region is a diamond (or hypercube).

The optimal $\hat{\beta}$ is where the elliptical contours of the OLS loss intersect the constraint region.



This visualization explains why lasso can set coefficients to exactly zero — the corners of the diamond make it easier to land on axes, while the smooth edge of a circle makes it less likely.

4.2.7 Generalized Linear Models (GLMs)

Generalized Linear Models are an extension of classical linear models that allow for more flexibility in modeling the relationship between inputs and outputs—particularly when the response variable y does not follow a normal distribution.

In a classical linear model, we often expect $\mathbb{E}[y_i] \in \mathbb{R}$, and the model assumes:

$$\mathbb{E}[y_i] = X_i^T \beta$$

However, in a GLM, we model a **transformation** of $\mathbb{E}[y_i]$ instead:

$$g(\mathbb{E}[y_i]) = X_i^T \beta$$

where:

- $g : \Omega \to \mathbb{R}$ is a monotonic, differentiable function known as the **link function**,
- Ω is the domain of $\mathbb{E}[y_i]$.

Examples:

1. **Binary Data:** If $y_i \in \{0, 1\}$, assume $y_i \sim \text{Bernoulli}(p_i)$. Then, the link function is:

$$g(p) = \log\left(\frac{p}{1-p}\right) \quad \text{(logit)}$$

So the model becomes:

$$g(\mathbb{E}[y_i]) = \log\left(\frac{p_i}{1-p_i}\right) = X_i^T \beta \quad \Rightarrow \quad \mathbb{E}[y_i] = \frac{e^{X_i^T \beta}}{1 + e^{X_i^T \beta}}$$

2. **Count Data:** If $y_i \in \mathbb{N}$, assume $y_i \sim \text{Poisson}(\lambda_i)$ with $\lambda_i > 0$. Use the link function:

$$\log(\lambda_i) = X_i^T \beta \quad \Rightarrow \quad \lambda_i = e^{X_i^T \beta}$$

Model Fitting. GLMs are typically fitted using **Maximum Likelihood Estimation (MLE)** rather than OLS. Since the likelihood is often not available in closed form, numerical optimization techniques such as **Iteratively Reweighted Least Squares (IRLS)** are used.

Definition (Maximum Likelihood Estimation): MLE chooses parameters β that maximize the likelihood function:

$$L(\beta) = \prod_{i=1}^{n} p(y_i \mid x_i, \beta)$$

This corresponds to choosing parameters that make the observed data most probable.

MLE for Logistic Regression: Assume $y_i \sim^{iid} \text{Bernoulli}(p_i)$, with:

$$p_i = \sigma(X_i^T \beta) = \frac{1}{1 + e^{-X_i^T \beta}}$$

Then the likelihood is:

$$L(\beta \mid y) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i} = \prod_{i=1}^{n} \sigma(X_i^T \beta)^{y_i} (1 - \sigma(X_i^T \beta))^{1 - y_i}$$

The log-likelihood becomes:

$$\ell(\beta \mid y) = \log L(\beta \mid y) = \sum_{i=1}^{n} \left[y_i X_i^T \beta - \log(1 + e^{X_i^T \beta}) \right]$$

This form is easier to optimize numerically. Its gradient, called the **score function**, is:

$$\nabla_\beta \ell(\beta \mid y) = \sum_{i=1}^{n} (y_i - \sigma(X_i^T \beta)) X_i = X^T (\mathbf{y} - \hat{\mathbf{p}})$$

This is the canonical gradient used in gradient descent to estimate parameters.

Definition (Odds): In the Bernoulli case, the **odds** are defined as:

$$\text{odds} = \frac{p_i}{1 - p_i}$$

It represents how much more likely the outcome is 1 rather than 0. Example: if $p_i = 0.75$, then odds = 3 (3 times more likely to be 1 than 0).

$$\log(\text{odds}) = X_i^T \beta \quad \text{(logit)}$$

Definition (Odds Ratio): Fix all variables except x_{ij}, and increase x_{ij} by 1. The odds ratio becomes:

$$\frac{\text{odds after increase}}{\text{original odds}} = e^{\beta_j}$$

It tells us how much the odds change when x_{ij} increases by one unit.

Prediction and Thresholding. GLMs output probabilities $\hat{p}_i \in [0, 1]$. To make a final decision in classification tasks, a threshold $r \in (0, 1)$ must be selected:

$$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_i > r \\ 0 & \text{otherwise} \end{cases}$$

This allows construction of a confusion matrix.

AUC-ROC: Area Under the Receiver Operating Characteristic Curve

Definition: AUC-ROC measures the quality of a binary classifier across all possible thresholds $r \in (0, 1)$. For each threshold, compute:

$$\text{TPR} = \frac{\text{TP}}{\text{Number of positives}}$$
$$\text{FPR} = \frac{\text{FP}}{\text{Number of negatives}}$$

Plot TPR (True Positive Rate) vs. FPR (False Positive Rate). The area under this curve (AUC) quantifies classifier performance. An AUC of 1.0 means perfect classification; an AUC of 0.5 corresponds to random guessing.

Logistic regression : Logistic regression is a fundamental classification technique, originally designed for binary outcomes. When dealing with more than two classes ($g > 2$), the model can be extended using **one-vs-rest** (OvR) or **multinomial logistic regression**.

In the one-vs-rest approach, we fit g separate binary classifiers. Each model estimates the probability that an observation belongs to class k versus all other classes:

$$\hat{p}_i^{(k)} = \mathbb{P}(y_i = k \mid x_i) \quad \text{for } k = 1, \ldots, g$$

The predicted class $\hat{y}_i$ is then assigned as the class with the highest estimated probability:

$$\hat{y}_i = \arg\max_k \hat{p}_i^{(k)}$$

Alternatively, for multiclass problems—especially when g is large—we use **multinomial logistic regression**, also known as **softmax regression**. In this case, a single model is trained to output a probability distribution over the g classes using the softmax function. The input features are often one-hot encoded when categorical variables are present.

This framework allows the model to compute class probabilities and assign the most likely class to each observation.

4.3 Regression and Classification trees

Fino a questo punto, we've seeing the most relevant method to do classification and regression, but it has got some weak points about the use of linear models:

1. Parametric assumptions
2. Lack of interaction among variables (some with the assumption to be iid)

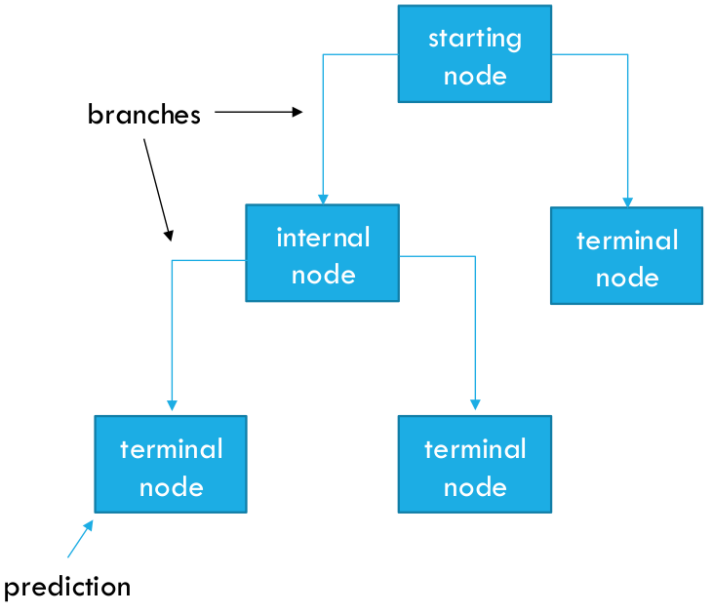


Figure 2: Example of a decision tree

From this simple concept that tree-based methods was borned, i.e. they use relaxing the parametric assumptions and they allow interaction between variables.

Tree-based classification and regression methods involve stratifying the predictor space into a number of simple regions where constant predictions can be made. This segmentation of the space can be represented through a **decision tree** (you can see an example in figure 2). In general, following the boxes from bottom to top, the importance of the splitting variable increase and, with it, their training.

How to split into boxes?: using CARTs. They identify a number of simple regions of the covariate space through binary splits. Those regions (high-dimensional rectangles) are named $R_1, R_2 \dots$ and, graphically, they represent a section of the covariate space (we can see an example in figure 3).

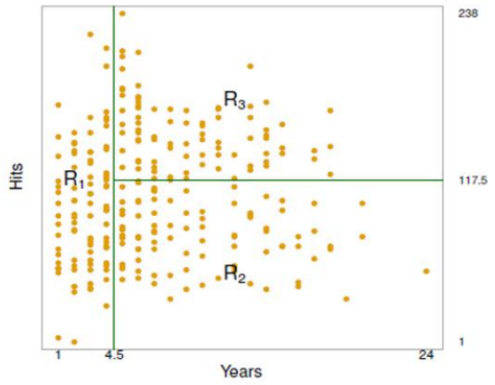


Figure 3: Example of boxes subdivision in the covariate space

Aim: for each box created, predict values for which you split results.

Process to build a regression tree: The main process involves two steps:

1. Divide the predictor space, that is the set of possible values for $X_1, X_2, \dots, X_p$ into a number J of distinct and non-overlapping regions $R_1, R_2, \dots R_p$.
2. For every observation that falls into region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

The rule to find the best regions is minimize the **Residual sum of squares (RSS)**:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - y_{\hat{R}_j})^2$$

Where $y_{\hat{R}_j}$ is the mean response for the training observations within the j-th box. Since it is computationally infeasible to consider every possible partition of the covariate space into J boxes, regression trees take a top-down, greedy approach that is known as **recursive binary splitting**.

Note that exists only squared cuts for regions, not diagonal ones. This because is a lot more complex to understand trees like that.

At each binary splitting, the algorithm first selects the predictor X_j and the threshold s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j > s\}$ leads to the greatest possible reduction in RSS.

We can define the **Half-planes** such:

$$R_1(j, s) = \{X \mid X_j < s\}$$

$$R_2(j, s) = \{X \mid X_j > s\}$$

For each of this split, we divide dataset in R_1 and R_2 and we're going to compute:

$$y_{\hat{R}_1} = \frac{1}{|R_1|} \sum_{i: x_i \in R_1(j,s)} y_i$$

And of course same for computing $y_{\hat{R}_2}$. After this, we can compute the total RSS and minimize it.

Note that the process of binary splitting continues until a stopping criterion is reached.

4.3.1 Tree pruning

When a decision tree is grown to its full depth, it tends to overfit the training data. This results in poor generalization to new, unseen observations. To address this issue, we apply a process called **tree pruning**, which simplifies the tree by removing splits that do not provide substantial predictive power.

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

where:

- $|T|$: number of terminal nodes (leaves) in tree T, (i.e. each subtree)
- $\alpha \geq 0$: tuning parameter controlling the trade-off between subtree size (complexity) and data fit to training data.

Interpretation:

1. A larger α penalizes complex trees more heavily, favoring smaller trees.
2. When $\alpha = 0$, we recover the fully grown tree T_0.

Algorithm:

1. Grow a large tree T_0 that overfits the training data using a recursive binary splitting alg. This until stopping criteria is reached.
2. Generate a sequence of subtrees $T_1, T_2, \ldots$ by cost complexity pruning.
3. Use k-fold cross-validation to select the subtree T_{α^*} that minimizes the estimated test error. For each $k = k_1, \ldots K$ repeat step 1 and 2 on all but the k-th fold of the training data.
4. Average the results for each value of α and pick one to minimize the avg error.
5. Finally, return the subtree from step 2 that corresponds to the chosen value of α.

Pruning enhances both interpretability and predictive performance by removing parts of the tree that capture noise rather than signal. It can capture very complex shapes in datas.

4.3.2 Classification trees

Classification trees are a special case of decision trees where the response variable Y is categorical. The goal is to predict the class label of a given observation by learning decision rules inferred from the data features.

As with regression trees, the predictor space is recursively partitioned into distinct, non-overlapping regions using binary splits. However, since the outcome is categorical, we need different criteria to measure the quality of a split.

Splitting Criteria. Instead of minimizing the residual sum of squares (RSS) as in regression, classification trees aim to maximize the **purity** of the resulting child nodes. Several impurity measures are commonly used:

- **Classification Error:**
$$E_j = 1 - \max_k \hat{p}_{jk}$$
where $\hat{p}_{jk}$ is the proportion of observations in region R_j that belong to class k.
- **Gini Index:**
$$G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}) = 1 - \sum_{k=1}^{K} \hat{p}_{jk}^2$$
The Gini index is a measure of total variance across the classes in a node. It reaches its minimum (0) when the node is pure. (less variance, more purity)
- **Cross-Entropy (Deviance):**
$$D_j = -\sum_{k=1}^{K} \hat{p}_{jk} \log(\hat{p}_{jk})$$
This measure comes from information theory and is particularly sensitive to class probabilities.

Although classification error is intuitive, Gini and cross-entropy are typically preferred during tree construction due to their better sensitivity to changes in node purity.

Prediction. Once the tree is built, to classify a new observation:

- Follow the decision path based on feature values to a terminal node.
- Assign the observation to the majority class in that node.

Advantages:

- Easy to interpret and visualize
- Can handle both numerical and categorical predictors
- Naturally captures feature interactions
- No need for feature scaling or transformation

Limitations:

- High variance — prone to overfitting without pruning
- Often outperformed by ensemble methods (e.g., Random Forests, Boosting)

Unfortunally, trees have got some disadvantages w.r.t. a parametric approach, that they suffer from high variance, predictor masking and are computationally expensive. For this reason, by aggregating many decision trees, we can see that the predictive performance of trees improved. We have to use trees as building blocks to build more powerful prediction models and average them.

4.3.3 Bagging (Bootstrap Aggregating)

Definition: Bagging is an ensemble technique that aims to reduce variance by averaging the predictions of multiple decision trees trained on different bootstrap samples of the original dataset.

- Given a training set of size n, draw B bootstrap samples of size n (with replacement).
- Train an independent tree T_b on each bootstrap sample.
- Aggregate predictions:
  - **Regression:** average the predictions:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

  - **Classification:** majority vote among the B classifiers.

Out-of-Bag (OOB) Estimation: There is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach. On average, each bagged tree makes use of around 2/3 of the observations. The remaining 1/3 of the observations are referred to as the out-of-bag (OOB) observations.These **OOB observations** can be used to:

- Evaluate model accuracy without the need for a separate validation set.
- Compute the average OOB error as an estimate of test error.

Advantages:

- Reduces model variance.
- Improves prediction accuracy compared to a single tree.
- Easy to parallelize.

We can obtain an OOB prediction for each of the n observations. With B sufficiently large, OOB error is virtually equivalent to leave-one-out CV error.

4.3.4 Random Forests

Definition: Random Forests are an enhancement of bagging that further reduce correlation between individual trees by injecting additional randomness during tree construction.

Key idea: At each split in a tree, instead of considering all p predictors, we randomly select a subset of $m < p$ predictors and only search for the best split among them. This gives a chance to each predictor and deals with correlated predictors.

- For classification, a typical choice is $m = \sqrt{p}$
- For regression, a common choice is $m = \frac{p}{3}$

This modification decorrelates the trees, resulting in better variance reduction when aggregating their predictions.

Feature Importance: Random forests naturally provide a measure of variable importance:

- One method is to compute the total decrease in node impurity attributable to each variable, averaged over all trees.
- Another is the **permutation importance**, which measures the drop in accuracy when the values of a variable are randomly permuted.

Advantages:

- High accuracy
- Resistant to overfitting
- Handles large numbers of features and observations well
- Provides insights into variable importance

So, if p = m, we have to use bagging. Instead, if $m < p$, we have to use random forest.

Note in general that there is the **Bias-Variance tradeoff**, i.e. a simple model will have high bias (adapts bad to data), but low variance (stable answers to data not in the training data). A complex model the opposite, low bias but high variance.

4.3.5 Boosting

Definition: Boosting is a powerful ensemble technique that builds models sequentially, with each new model trained to correct the errors made by the previous ones.

Unlike bagging and random forests, where trees are grown independently, boosting fits trees in sequence and combines them into a single strong learner. Boosting does not involve bootstrap sampling, but each tree is fitted on a modified version of the original data.

Gradient Boosting for Regression:

- Start with an initial prediction, usually the mean of y: $f_0(x) = \bar{y}$
- For $b = 1, \ldots, B$:
  1. Compute residuals (pseudo-residuals): $r_i^{(b)} = y_i - f_{b-1}(x_i)$
  2. Fit a regression tree T_b to the residuals (A large residual means the current ensemble is still doing a poor job on that point, so it contributes more to the loss we're trying to minimise).
  3. Update the model:
$$f_b(x) = f_{b-1}(x) + \lambda T_b(x)$$
- Output the boosted model.

λ is a learning rate (shrinkage parameter), typically $\lambda \in [0.01, 0.2]$

Important aspects:

- Trees are usually shallow (e.g., 1–5 splits)
- Small learning rate improves generalization but requires more trees
- Regularization techniques (like early stopping) help avoid overfitting

Boosting has 3 tuning parameters: number of trees B (can overfit if high, use CV), the shrinkage parameter λ and the number of splits d in each tree.

Comparison with Bagging/Random Forests:

- Boosting reduces bias and variance; bagging mostly reduces variance
- Boosting is sequential and more sensitive to noise
- Random forests are easier to parallelize

Summary Table: Tree-Based Methods Comparison

Method	Variance Reduction	Bias Reduction	Interpretability
Single Tree	No	No	High
Bagging	Yes	No	Medium
Random Forest	Yes (stronger)	No	Medium
Boosting (best)	Yes	Yes	Low

We can also measure the variables importance measure (how much each variable is important) or how much single covariant influences. Decision trees are simple and interpretable tools for building regression and classification models and Randomized methods (bagging, random forests) allow to significantly improve the predictive accuracy.