Preprocessing

Min-Max: $X_{\text{norm}} = \dfrac{X - X_{\min}}{X_{\max} - X_{\min}}$

Range: $X_{\text{range}} = \dfrac{X - X_{\min}}{X_{\max} - X_{\min}}(b-a) + a$

Z-score: $Z = \dfrac{X - \mu}{\sigma}$ **Required for PCA**

PCA

Goal: Find orthogonal axes capturing max variance in fewer dimensions.

Steps

1. **Standardize** data (Z-score each feature).
2. **Compute covariance matrix C** ($n \times n$).
3. **Compute eigenvalues λ_i and eigenvectors $\mathbf{v}_i$** of **C**.
4. **Sort** eigenvectors by λ in decreasing order.
5. **Select top** k eigenvectors.
6. **Project** standardized data onto k eigenvectors.

Covariance

$$\text{Cov}(X_i, X_j) = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)$$

C is symmetric: $C_{ij} = \text{Cov}(X_i, X_j)$.

Eigen Decomposition

$\mathbf{C}\,\mathbf{v} = \lambda\,\mathbf{v}$

- λ = eigenvalue (variance captured)
- $\mathbf{v}$ = eigenvector (direction)

Variance Explained

$\text{Ratio}_i = \dfrac{\lambda_i}{\sum_j \lambda_j}$ Choose k: cumulative $\geq$ 90–95%

Projection

$\text{PC}_i = a_1x_1 + a_2x_2 + \cdots + a_nx_n$

a_j from eigenvector $\mathbf{v}_i$; x_j = standardized features.

Properties

- **Unsupervised** (ignores class labels)
- Components are **orthogonal**
- Max $k \leq \min(n_{\text{samples}}, n_{\text{features}})$

PCA vs. LDA

	PCA	LDA
Type	Unsupervised	Supervised
Goal	Max variance	Max between/min within
Max k	$\min(N, d)$	$\min(C-1, d)$

N=samples, d=features, C=classes.

Distance Measures

Euclidean: $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

Manhattan: $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}|x_i - y_i|$

K-Means Clustering

Algorithm

1. Select K initial centroids.
2. **Repeat:**
   a. **Assign** each point to nearest centroid.
   b. **Recompute** centroids as cluster means.
3. **Until** centroids stop changing.

Objective (SSE) & Centroid Update

$\text{SSE} = \sum_{i=1}^{K}\sum_{\mathbf{x}\in C_i}\|\mathbf{x} - \mathbf{m}_i\|^2$ $\quad$ $\mathbf{m}_i = \dfrac{1}{|C_i|}\sum_{\mathbf{x}\in C_i}\mathbf{x}$

Complexity: $O(n \cdot K \cdot I \cdot d)$ (n=pts, K=clusters, I=iters, d=dims)

Properties

- Convergence guaranteed (SSE monotonically $\downarrow$)
- Sensitive to initial centroids
- Finds **spherical/globular** clusters
- **Hard** clustering (1 cluster per point)

Hierarchical Clustering

Agglomerative (Bottom-Up)

1. Compute distance matrix.
2. Each point = its own cluster.
3. **Repeat:** merge two closest clusters; update distances.
4. **Until** single cluster remains.

Cut **dendrogram** at desired height for K clusters.

Linkage Methods

Method	Inter-Cluster Distance
MIN (Single)	Closest pair
MAX (Complete)	Farthest pair
Group Average	Avg of all cross-pairs
Centroid	Between centroids
Ward's	Min ΔSSE

MIN $\rightarrow$ elongated chains **MAX** $\rightarrow$ compact spheres **Avg** $\rightarrow$ compromise

DBSCAN

Params: ε (radius), MinPts (min points).

Point Types

- **Core**: $\geq$ MinPts within ε (incl. itself)
- **Border**: not core, but in ε-neighborhood of core
- **Noise**: neither core nor border

Algorithm

1. Label points as core, border, or noise.
2. Remove noise points.
3. Connect core points within ε.
4. Connected components of cores = clusters.
5. Assign border points to nearby core's cluster.

Properties

- Finds **arbitrary-shaped** clusters
- Resistant to noise/outliers
- **No** K required
- Struggles with varying density

EM / Gaussian Mixture Models

Soft clustering: each point has probability of belonging to each cluster.

Gaussian PDF

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi}\,\sigma_b}\,e^{-\frac{(x_i-\mu_b)^2}{2\sigma_b^2}}$$

E-Step (Expectation)

Membership weight via Bayes' rule:

$$w_{ib} = P(b \mid x_i) = \frac{P(x_i \mid b)\,P(b)}{\sum_k P(x_i \mid k)\,P(k)}$$

M-Step (Maximization)

$\mu_b = \dfrac{\sum_i w_{ib}\,x_i}{\sum_i w_{ib}}$ $\quad$ $\sigma_b^2 = \dfrac{\sum_i w_{ib}\,(x_i - \mu_b)^2}{\sum_i w_{ib}}$

Algorithm

1. Initialize K Gaussians with random μ, σ.
2. **E-step:** compute $P(\text{cluster} \mid x_i)$ $\forall$ points.
3. **M-step:** update μ, σ per cluster.
4. **Repeat** until convergence.

Cluster Validity

SSE – Cohesion (lower = tighter)

$\text{SSE} = \sum_{i=1}^{K}\sum_{\mathbf{x}\in C_i}(\mathbf{x} - \mathbf{m}_i)^2$

SSB – Separation (higher = better)

$\text{SSB} = \sum_{i=1}^{K} N_i\,(\mathbf{m}_i - \mathbf{m})^2$

$\mathbf{m}$ = grand mean, $N_i = |C_i|$. $\quad$ SSE + SSB = TSS

Silhouette Coefficient

- a = avg dist to points in *same* cluster
- b = min avg dist to points in *another* cluster

$$s(i) = \frac{b-a}{\max(a,b)}, \quad s \in [-1, 1]$$

$s \approx 1$	Well-clustered
$s \approx 0$	On boundary
$s \approx -1$	Misclassified

Overall = average $s(i)$ over all points.

Choosing K

- **Elbow**: plot SSE vs. K, pick the bend
- **Silhouette**: K with highest avg s
- **Dendrogram**: cut at natural gap

Hard vs. Soft Clustering

	Hard	Soft
Assign	1 cluster/point	Probabilities
Example	K-Means	EM / GMM

Conditional Probability

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}$$

Joint Probability

$$P(X \cap Y) = P(X \mid Y) \cdot P(Y)$$

Bayes' Theorem

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)}$$

Conditional Independence

X, Y conditionally independent given Z if:
$P(X \mid Y, Z) = P(X \mid Z)$

Naive Bayes Classifier

Goal: Find class Y that maximizes $P(Y \mid X_1, \ldots, X_D)$.

Steps

1. Compute **class priors** $P(Y_j)$ from training data.
2. For each feature X_i, compute $P(X_i \mid Y_j)$.
3. **Multiply** likelihoods (naive independence assumption).
4. **Classify** as class with highest score.

MAP Classification

$$\hat{Y} = \arg\max_Y \; P(X_1, \ldots, X_D \mid Y) \cdot P(Y)$$

Since $P(X)$ is constant across classes, ignore the denominator.

Naive Independence Assumption

$$P(X_1, \ldots, X_D \mid Y_j) = \prod_{i=1}^{D} P(X_i \mid Y_j)$$

Assumes all features are independent given the class.

Categorical Features

$$P(X_i = c \mid Y = y_j) = \frac{N_c}{N}$$

N_c = count of $X_i = c$ in class y_j; N = total in class y_j.

Continuous Features (Gaussian)

$$P(X_i \mid Y = y_j) = \frac{1}{\sqrt{2\pi}\,\sigma_{ij}} \exp\left(-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

Compute μ_{ij} and σ_{ij} from training data for feature i, class j.

Zero Probability Problem

If **any** $P(X_i \mid Y) = 0$, the entire product becomes 0, regardless of all other features.

Laplace Smoothing

$$P(X_i = c \mid Y) = \frac{N_c + 1}{N + V}$$

V = number of possible values for X_i.

m-Estimate Smoothing

$$P(X_i = c \mid Y) = \frac{N_c + \varepsilon \cdot p}{N + \varepsilon}$$

p = prior estimate; ε = smoothing strength.

Naive Bayes Properties

- Robust to noise and outliers
- Handles **missing values** (classify with available features)
- Works with partial information
- **Weakness**: correlated features violate independence
- Simple, fast, works well in practice

Bayesian Belief Network

Used when features are **dependent** (Naive Bayes assumption fails).

Structure

- **Directed Acyclic Graph (DAG)**
- Nodes = random variables
- Edges = dependency relationships
- Each node has a **conditional probability table**

Key Property

A node is **conditionally independent** of all non-descendants given its parents.

Probability Tables

- Root node (no parents): prior $P(X)$
- Child node: $P(X \mid \text{parents})$

Classification

For each class value, multiply:
$P(\text{parents}) \times P(\text{node} \mid \text{parents}) \times P(\text{children} \mid \text{node})$
Pick class with highest score.

Loss Functions

Regression Losses

MSE (L2 Loss): $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

MAE (L1 Loss): $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Mean Bias Error: $\text{MBE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$

Positive/negative errors cancel; detects directional bias.

Probabilistic Losses

Binary CE	2 classes (0/1 labels)
Categorical CE	Multi-class (one-hot)
Sparse Cat. CE	Multi-class (integer labels)

Hinge Losses

Used for SVMs. Labels: -1 and $+1$.
Maximizes margin between classes.

Gradient Descent

Weight Update

$$W_{\text{new}} = W_{\text{current}} - \alpha \cdot \frac{\partial L}{\partial W}$$

α = learning rate (hyperparameter).

Chain Rule

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W}$$

General: $\frac{dy}{dx} = \frac{dy}{df} \cdot \frac{df}{dg} \cdot \frac{dg}{dx}$

Parameters vs. Hyperparameters

	Parameters	Hyperparameters
Set by	Model (learned)	Engineer (chosen)
Examples	Weights, biases	LR, epochs, batch size