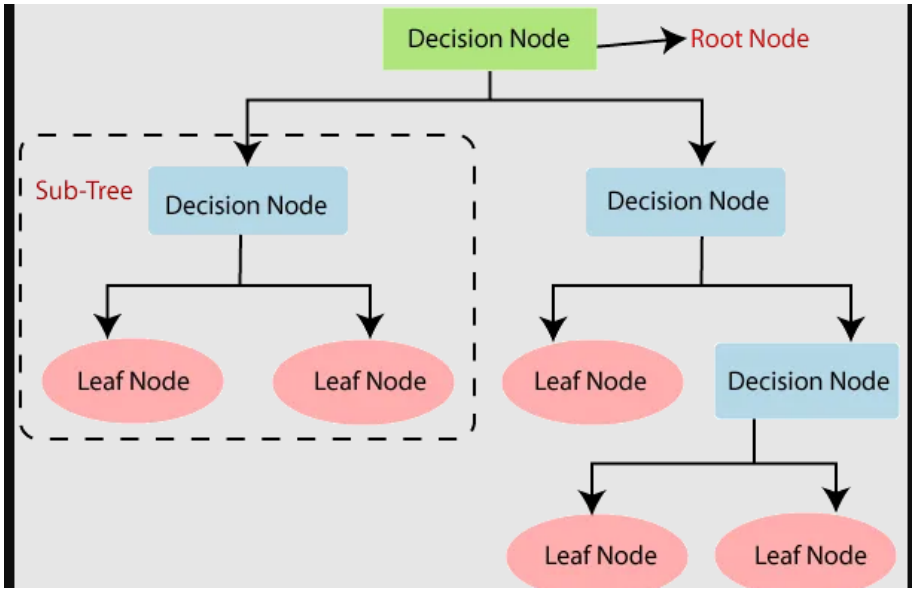
CHAPTER FOUR: CATEGORICAL AND MIXED-TYPE DATA

4.1 What is a Decision Tree?

A **decision tree** is a type of supervised machine learning algorithm used for both **classification** and **regression** tasks.

It works by splitting the dataset into smaller and smaller subsets based on certain conditions, forming a tree-like structure. The goal is to find the *best splits* that help separate the data points clearly based on the target variable.



Components of a Decision Tree

- **Root Node:** The topmost node. It represents the entire dataset, and the first decision (split) is made here.
- **Internal/Decision Nodes:** These are the points where the dataset is split based on a feature.
- **Branches:** These represent the outcome of a split (e.g., `Yes` or `No`, $\leq$ threshold or $>$ threshold).
- **Leaf/Terminal Nodes:** These are the final outcomes (classes or predicted values).

How Does It Work?

The decision tree algorithm chooses the best feature and split point using a criterion such as:

- **Information Gain** (used in ID3)
- **Gain Ratio** (used in C4.5)
- **Gini Impurity** (used in CART)

Predictors / Target

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

It then recursively splits the data until one of the following conditions is met:

- All instances in a node belong to the same class (i.e., the node is **pure**).
- A predefined stopping criterion is satisfied (e.g., maximum depth reached, or minimum number of samples per leaf).

How Splitting Happens in Decision Trees

Decision trees split the data based on impurity measures. Here are the common ones:

1. Gini Impurity (Used in CART)

$$\text{Gini} = 1 - \sum_{i=1}^{n} p_i^2$$

Where p_i is the probability of class i. Gini measures how often a randomly chosen element would be incorrectly labeled.

2. Entropy (Used in ID3)

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Entropy measures the impurity or randomness in the data.

3. Information Gain

Used to decide the best feature to split on.

$$\text{Information Gain} = \text{Entropy(Parent)} - \sum \left(\frac{|\text{Child}|}{|\text{Parent}|} \times \text{Entropy(Child)} \right)$$

4. Mean Squared Error (MSE) — Regression Trees

Used for continuous target variables.

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

Algorithms for Building Decision Trees

Algorithm	Key Idea	Use Case
ID3	Uses entropy and information gain	Classification
C4.5	Improvement over ID3; handles continuous data	Classification
CART	Uses Gini (classification) or MSE (regression)	Classification & Regression
CHAID	Uses statistical significance tests	Multi-level categorical splits

Pruning: Avoiding Overfitting

Decision trees can easily overfit the training data. **Pruning** helps by simplifying the model.

- **Pre-pruning (Early Stopping):** Stop growing the tree if a condition is met (e.g., minimum samples, max depth).
- **Post-pruning:** First grow the full tree, then remove nodes that provide little to no improvement.

Pros and Cons of Decision Trees

Advantages

- Easy to understand and visualize
- No need for feature scaling or normalization
- Works with both numerical and categorical data

Disadvantages

- Prone to overfitting
- Can be unstable — small data changes may produce different trees
- Biased toward features with many levels/categories

Real-world Applications

- Medical Diagnosis (e.g., classifying tumors as benign or malignant)
- Credit Scoring
- Customer Churn Prediction
- Fraud Detection

4.1.1 What ID3 Does

ID3 builds a decision tree top-down, starting from the root and recursively selecting the best attribute to split the data. It uses **information gain** to decide which attribute to split on.

Key Concepts

1. Entropy

Entropy is a measure of impurity or disorder in a dataset. It is calculated as:

$$\text{Entropy}(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

where:

- S is a set of examples,
- p_i is the proportion of examples in class i,
- c is the number of classes.

2. Information Gain

Information Gain measures how much "information" a feature gives us about the class. It is defined as:

$$IG(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$

where:

- A is the attribute,
- S_v is the subset of S for which attribute A has value v.

The attribute with the highest information gain is selected to split the node.

ID3 Algorithm Steps

1. Start with the full dataset.

2. Compute entropy of the dataset.

3. For each attribute, compute information gain.

4. Choose the attribute with the highest information gain to split the dataset.

5. Repeat recursively for each subset until:

- All examples in a subset belong to the same class,
- There are no more attributes to split on,
- The subset is empty.

Dataset Summary

Outlook	Temperature	Humidity	Windy	PlayGolf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

We have 14 instances with the target variable `PlayGolf`, which has 9 `Yes` and 5 `No` instances.

1. ID3 (Information Gain with Entropy)

Step 1: Total Dataset Summary

There are 14 records in total.

Play Golf	Count
Yes	9
No	5

Initial Entropy $E(S)$:

$$E(S) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$
$$= -0.643 \cdot \log_2(0.643) - 0.357 \cdot \log_2(0.357) \approx 0.940$$

Step 2: Compute Information Gain for Each Attribute

We now calculate the expected entropy for each attribute split.

Attribute: Outlook

- Sunny: 5 samples (2 Yes, 3 No)

$$E = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) \approx 0.971$$

- Overcast: 4 samples (4 Yes, 0 No)

$$E = 0$$

- Rainy: 5 samples (3 Yes, 2 No)

$$E = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) \approx 0.971$$

Information Gain:

$$Gain(S, Outlook) = 0.940 - \left(\frac{5}{14} \cdot 0.971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.971\right) = 0.940 - (0.693) = \boxed{0.247}$$

Attribute: Temperature

- Hot: 4 samples (2 Yes, 2 No), $E = 1.0$
- Mild: 6 samples (4 Yes, 2 No),

$$E = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) \approx 0.918$$

- Cool: 4 samples (3 Yes, 1 No), $E \approx 0.811$

$$Gain(S, Temp) = 0.940 - (0.286 + 0.393 + 0.232) = 0.029$$

Attribute: Humidity

- High: 7 samples (3 Yes, 4 No), $E \approx 0.985$
- Normal: 7 samples (6 Yes, 1 No), $E \approx 0.591$

$$Gain(S, Humidity) = 0.940 - \left(\frac{7}{14} \cdot 0.985 + \frac{7}{14} \cdot 0.591\right) = 0.940 - (0.788) = \boxed{0.152}$$

Attribute: Windy

- False: 8 samples (6 Yes, 2 No), $E \approx 0.811$
- True: 6 samples (3 Yes, 3 No), $E = 1.0$

$$Gain(S, Windy) = 0.940 - \left(\frac{8}{14} \cdot 0.811 + \frac{6}{14} \cdot 1.0\right) = 0.940 - 0.892 = \boxed{0.048}$$

Best attribute to split on: Outlook (Gain = 0.247)

Step 3: Split on Outlook

Outlook = Sunny

Subset size: 5 samples (2 Yes, 3 No) Entropy = 0.971

Split on **Humidity**:

- High: 3 samples (0 Yes, 3 No), $E = 0 \rightarrow$ No
- Normal: 2 samples (2 Yes, 0 No), $E = 0 \rightarrow$ Yes

Gain = 0.971

Outlook = Overcast

Subset size: 4 samples (4 Yes, 0 No) → Pure Node: Predict Yes

Outlook = Rainy

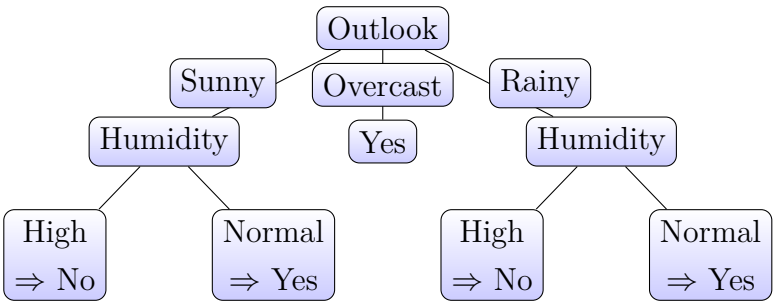
Subset size: 5 samples (3 Yes, 2 No) Entropy = 0.971

Try splitting on **Humidity**:

- High: 2 samples (0 Yes, 2 No), $E = 0 \rightarrow$ No
- Normal: 3 samples (3 Yes, 0 No), $E = 0 \rightarrow$ Yes

Gain = 0.971

Final Decision Tree



Summary of Leaf Nodes

Path	Prediction
Outlook = Overcast	Yes
Outlook = Sunny, Humidity = High	No
Outlook = Sunny, Humidity = Normal	Yes
Outlook = Rainy, Humidity = High	No
Outlook = Rainy, Humidity = Normal	Yes

4.1.2 CART (Gini Index)

Overview

The **CART (Classification and Regression Trees)** algorithm is a decision tree learning method introduced by *Breiman et al. (1986)*. It can be used for both:

- **Classification** (predicting class labels), and
- **Regression** (predicting continuous values).

Unlike algorithms such as ID3 or C4.5, CART produces strictly **binary trees**—each internal node splits into exactly two children.

Key Concepts

1. Gini Impurity (for classification)

CART uses **Gini Impurity** to measure the impurity of a dataset. It is calculated as:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Where:

- S is the dataset at a node,
- p_i is the proportion of samples belonging to class i,
- c is the number of classes.

A lower Gini impurity indicates a purer node.

2. Mean Squared Error (for regression)

For regression tasks, CART uses **Mean Squared Error (MSE)** as the splitting criterion:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2$$

Where:

- y_i is the actual value,
- $\hat{y}$ is the predicted value (typically the mean of values in the node),
- n is the number of samples in the node.

CART Algorithm Steps

1. Start with the full dataset at the root node.

2. For each feature and potential split value, compute the impurity (Gini or MSE).

3. Select the split that **minimizes impurity** in the resulting child nodes.

4. Split the dataset into two child nodes based on this criterion.

5. Recurse on each child node until a stopping condition is met:

- All samples belong to the same class,
- Maximum tree depth is reached,
- Minimum number of samples per node is not met,
- No further information gain is possible.

Pruning

CART uses a method called **cost-complexity pruning** (also known as *weakest link pruning*):

- First, a fully grown tree is generated.
- Then, nodes are pruned based on a complexity parameter α that balances accuracy with tree size.

Features of CART

- **Tree Type:** Binary decision tree.

- **Splitting Criteria:** Gini impurity (classification), MSE (regression).
- **Supports:** Both categorical and numerical attributes.
- **Output:** Class label or numeric value.
- **Pruning:** Cost-complexity pruning supported.

Step 1: Gini Index of Full Dataset

$$Gini(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

Step 1: Dataset Summary

The dataset has 14 instances.

Play Golf	Count
Yes	9
No	5

Gini Index for root:

$$Gini(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 1 - 0.41 - 0.13 = \boxed{0.459}$$

Step 2: Try Splits Using Binary CART

CART evaluates **binary splits only** (e.g., Outlook = Sunny? Yes/No).

We'll compute weighted Gini for each binary split.

Split: Outlook = Sunny?

- Yes branch (Sunny): 5 samples (2 Yes, 3 No),

$$Gini = 1 - (2/5)^2 - (3/5)^2 = 0.48$$

- No branch (Not Sunny): 9 samples (7 Yes, 2 No),

$$Gini = 1 - (7/9)^2 - (2/9)^2 \approx 0.346$$

$$Gini_{split} = \frac{5}{14} \cdot 0.48 + \frac{9}{14} \cdot 0.346 \approx 0.398$$

Split: Outlook = Overcast?

- Yes branch (Overcast): 4 samples (4 Yes), Gini = 0
- No branch: 10 samples (5 Yes, 5 No),

$$Gini = 1 - (5/10)^2 - (5/10)^2 = 0.5$$

$$Gini_{split} = \frac{4}{14} \cdot 0 + \frac{10}{14} \cdot 0.5 = 0.357$$

Split: Humidity = High?

- High: 7 samples (3 Yes, 4 No),

$$Gini = 1 - (3/7)^2 - (4/7)^2 \approx 0.489$$

- Not High: 7 samples (6 Yes, 1 No),

$$Gini = 1 - (6/7)^2 - (1/7)^2 \approx 0.245$$

$$Gini_{split} = \frac{7}{14} \cdot 0.489 + \frac{7}{14} \cdot 0.245 = 0.367$$

Split: Windy = True?

- True: 6 samples (3 Yes, 3 No), Gini = 0.5

- False: 8 samples (6 Yes, 2 No),

$$Gini = 1 - (6/8)^2 - (2/8)^2 = 0.375$$

$$Gini_{split} = \frac{6}{14} \cdot 0.5 + \frac{8}{14} \cdot 0.375 = 0.429$$

Best Split: Outlook = Overcast? (Gini = 0.357)

—

Step 3: First Split - Outlook = Overcast?

Overcast = Yes → 4 samples (4 Yes) → Pure → **Yes**

Overcast = No → 10 samples (5 Yes, 5 No) → Needs further splitting

—

Step 4: Further Split on Outlook Overcast

Now use the remaining 10 samples to test other binary splits.

Try **Humidity = High?** → Gives next best split for this subset.

- High: 5 samples (0 Yes, 5 No) → Gini = 0
- Not High: 5 samples (5 Yes, 0 No) → Gini = 0

$$Gini_{split} = 0 \Rightarrow \text{Perfect split}$$

4.2 Variable Importance: Permutation Tests and Partial Dependence Plots

1. Permutation Feature Importance

Objective

To quantify the importance of each feature by measuring the change in model performance when the feature's values are randomly shuffled, breaking the relationship between that feature and the target.

Algorithm

1. Train the model on training data.
2. Evaluate a baseline performance metric (e.g., accuracy, RMSE):

$$M_{\text{base}} = \text{metric}(y, \hat{y})$$

3. For each feature X_i:
    - Shuffle its values in the validation/test set.
    - Recalculate predictions and evaluate the performance metric M_i.
    - Compute the importance score as:

$$I_i = M_i - M_{\text{base}}$$

Key Points

- **Model-agnostic**: Works with any type of model.
- **Captures feature interactions**.
- **Limitation**: Sensitive to correlated features.

Pros and Cons

- **Pros**: Simple, intuitive, supports any model.

- **Cons**: Computationally expensive; may underestimate importance of correlated features.

2. Partial Dependence Plots (PDP)

Objective

To visualize the marginal effect of one or two features on the predicted outcome by averaging out the effects of all other features.

Mathematical Formulation

Let $f(x)$ be the trained prediction function. The partial dependence of feature subset X_s is given by:

$$PD(X_s) = \mathbb{E}_{X_c}[f(X_s, X_c)]$$

Where:

- X_s: The feature(s) of interest.
- X_c: The complement set of all other features.
- The expectation is taken over the joint distribution of X_c.

Procedure

1. Select a feature X_s and a range of values for it.
2. For each value, substitute it in all rows of the dataset and make predictions.
3. Average the predictions to estimate the marginal effect at that value.

Key Points

- **Visual interpretation**: 1D or 2D plots.
- **Assumes feature independence**: Can be misleading if features are correlated.

- Often used with tree-based models like Random Forests or Gradient Boosted Trees.

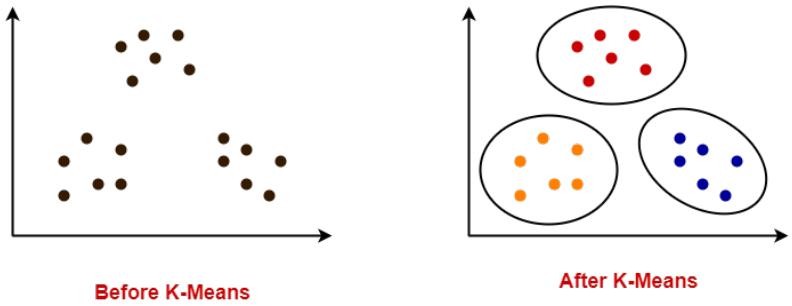
Comparison Table

Aspect	Permutation Importance	Partial Dependence Plot
Type	Quantitative metric	Visual interpretation
Output	Importance score	Plot of average model output
Captures Interactions	Yes	Yes (in 2D plots)
Feature Correlation Issue	Yes (underestimates importance)	Yes (can be misleading)
Model Compatibility	Any (model-agnostic)	Mostly tree-based models
Usefulness	Feature ranking	Understanding effect on prediction
Cost	Medium to High	Medium

Table 4.1: Comparison of Permutation Importance and PDP

4.3 K means clustering

K-Means clustering is an **unsupervised**, iterative clustering technique that partitions a given dataset into k predefined distinct clusters. A cluster is defined as a collection of data points exhibiting certain similarities.



Cluster Properties

The clustering is performed such that:

- Each data point belongs to a cluster whose mean (centroid) is nearest to it.
- Data points within the same cluster exhibit a high degree of similarity.

- Data points from different clusters show a high degree of dissimilarity.

K-Means Clustering Algorithm

The K-Means clustering algorithm involves the following steps:

Step-1: Choose the number of clusters k.

Step-2: Initialize cluster centers:

- Randomly select k data points as the initial cluster centers.
- Try to select them such that they are as far apart as possible.

Step-3: Assign data points to clusters:

- Calculate the distance between each data point and each cluster center.
- Commonly used distance metric: Euclidean distance.

Step-4: Cluster assignment:

- Assign each data point to the cluster with the nearest centroid.

Step-5: Recompute centroids:

- For each cluster, compute the new centroid by taking the mean of all data points assigned to that cluster.

Step-6: Repeat:

- Repeat steps 3 to 5 until one of the following conditions is met:
  - Cluster centers do not change.
  - Data point assignments remain unchanged.
  - A maximum number of iterations is reached.

Advantages

K-Means Clustering offers the following advantages:

- **Efficiency:** Time complexity is $\mathcal{O}(nkt)$, where:
  - n = number of data points,
  - k = number of clusters,
  - t = number of iterations.
- **Fast convergence:** The algorithm often converges quickly, although it may reach a local optimum.
- **Extensibility:** Techniques such as Simulated Annealing or Genetic Algorithms can be used to escape local optima and seek a global optimum.

Disadvantages

Despite its efficiency, K-Means has several limitations:

- Requires the number of clusters k to be specified in advance.
- Sensitive to noisy data and outliers.
- Poor performance with non-convex shaped clusters or clusters of varying density and size.

Given

- Number of clusters: $K = 2$
- Initial centroids:
  - $C_1 = (185, 72)$
  - $C_2 = (170, 56)$
- Data points:

Point	Coordinates
P_1	$(185, 72)$
P_2	$(170, 56)$
P_3	$(168, 60)$
P_4	$(179, 68)$
P_5	$(182, 72)$
P_6	$(188, 77)$

Iteration 1: Euclidean Distance

The Euclidean distance is calculated using:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Point	Coordinates	$d(P, C_1)$	$d(P, C_2)$	Cluster
P_1	$(185, 72)$	0	$\sqrt{15^2 + 16^2} = 21.93$	C_1
P_2	$(170, 56)$	21.93	0	C_2
P_3	$(168, 60)$	$\sqrt{17^2 + 12^2} = 20.80$	$\sqrt{2^2 + 4^2} = 4.47$	C_2
P_4	$(179, 68)$	$\sqrt{6^2 + 4^2} = 7.21$	$\sqrt{9^2 + 12^2} = 15.00$	C_1
P_5	$(182, 72)$	3.00	20.00	C_1
P_6	$(188, 77)$	$\sqrt{3^2 + 5^2} = 5.83$	27.66	C_1

Clusters after Iteration 1:

- C_1: P_1, P_4, P_5, P_6
- C_2: P_2, P_3

New Centroids

- For C_1:

$$\text{Mean} = \left(\frac{185 + 179 + 182 + 188}{4}, \frac{72 + 68 + 72 + 77}{4} \right) = (183.5, 72.25)$$

- For C_2:

$$\text{Mean} = \left(\frac{170+168}{2}, \frac{56+60}{2}\right) = (169, 58)$$

Iteration 2: Recalculate Distance

Point	Coordinates	$d(P, C_1')$	$d(P, C_2')$	Cluster
P_1	$(185, 72)$	$\sqrt{1.5^2 + 0.25^2} \approx 1.52$	$\sqrt{16^2 + 14^2} \approx 21.26$	C_1
P_2	$(170, 56)$	$\sqrt{13.5^2 + 16.25^2} \approx 21.13$	$\sqrt{1^2 + 2^2} \approx 2.24$	C_2
P_3	$(168, 60)$	$\sqrt{15.5^2 + 12.25^2} \approx 19.71$	$\sqrt{1^2 + 2^2} \approx 2.24$	C_2
P_4	$(179, 68)$	$\sqrt{4.5^2 + 4.25^2} \approx 6.20$	$\sqrt{10^2 + 10^2} \approx 14.14$	C_1
P_5	$(182, 72)$	$\sqrt{1.5^2 + 0.25^2} \approx 1.52$	$\sqrt{13^2 + 14^2} \approx 19.10$	C_1
P_6	$(188, 77)$	$\sqrt{4.5^2 + 4.75^2} \approx 6.55$	$\sqrt{19^2 + 19^2} \approx 26.87$	C_1

Clusters after Iteration 2:

- C_1: P_1, P_4, P_5, P_6
- C_2: P_2, P_3

Conclusion

After two iterations, the clusters did not change. Hence, the algorithm has converged. The final clusters are:

- **Cluster 1 ($\mathbf{C}_1$)**: (185, 72), (179, 68), (182, 72), (188, 77)
- **Cluster 2 ($\mathbf{C}_2$)**: (170, 56), (168, 60)

Problem Statement

Cluster the following eight 2D points using the K-Means algorithm with $K = 3$ and perform 2 iterations.

Data Points:

Point	Coordinates
A_1	$(2, 10)$
A_2	$(2, 5)$
A_3	$(8, 4)$
A_4	$(5, 8)$
A_5	$(7, 5)$
A_6	$(6, 4)$
A_7	$(1, 2)$
A_8	$(4, 9)$

Initial Cluster Centers:

- Cluster 1 (C1): $A_1 = (2, 10)$
- Cluster 2 (C2): $A_4 = (5, 8)$
- Cluster 3 (C3): $A_7 = (1, 2)$

Iteration 1: Distance Calculation

Distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Point	Coordinates	$d(C1)$	$d(C2)$	$d(C3)$	Cluster
A_1	$(2, 10)$	0.00	3.61	8.06	$C1$
A_2	$(2, 5)$	5.00	3.61	3.16	$C3$
A_3	$(8, 4)$	8.48	5.00	7.28	$C2$
A_4	$(5, 8)$	3.61	0.00	6.71	$C2$
A_5	$(7, 5)$	7.62	3.61	6.08	$C2$
A_6	$(6, 4)$	7.21	4.47	5.39	$C2$
A_7	$(1, 2)$	8.06	6.71	0.00	$C3$
A_8	$(4, 9)$	2.24	1.41	7.28	$C2$

Clusters after Iteration 1:

- Cluster 1 (C1): A_1

- Cluster 2 (C2): A_3, A_4, A_5, A_6, A_8
- Cluster 3 (C3): A_2, A_7

New Centroids After Iteration 1

C1: Only $A_1 \rightarrow (2, 10)$

C2:

$$x = \frac{8+5+7+6+4}{5} = 6, \quad y = \frac{4+8+5+4+9}{5} = 6$$
$$\Rightarrow C2 = (6,6)$$

C3:

$$x = \frac{2+1}{2} = 1.5, \quad y = \frac{5+2}{2} = 3.5$$
$$\Rightarrow C3 = (1.5, 3.5)$$

Iteration 2: Distance Calculation

Point	Coordinates	$d(C1:2,10)$	$d(C2:6,6)$	$d(C3:1.5,3.5)$	Cluster
A_1	$(2,10)$	0.00	5.00	6.54	$C1$
A_2	$(2,5)$	5.00	4.12	1.80	$C3$
A_3	$(8,4)$	8.48	2.83	6.55	$C2$
A_4	$(5,8)$	3.61	2.24	5.70	$C2$
A_5	$(7,5)$	7.62	1.41	5.70	$C2$
A_6	$(6,4)$	7.21	2.00	4.53	$C2$
A_7	$(1,2)$	8.06	5.00	1.58	$C3$
A_8	$(4,9)$	2.24	3.61	6.40	$C1$

Clusters after Iteration 2:

- Cluster 1 (C1): A_1, A_8
- Cluster 2 (C2): A_3, A_4, A_5, A_6
- Cluster 3 (C3): A_2, A_7

New Centroids After Iteration 2

C1:

$$x = \frac{2+4}{2} = 3, \quad y = \frac{10+9}{2} = 9.5 \Rightarrow C1 = (3, 9.5)$$

C2:

$$x = \frac{8+5+7+6}{4} = 6.5, \quad y = \frac{4+8+5+4}{4} = 5.25 \Rightarrow C2 = (6.5, 5.25)$$

C3:

$$x = \frac{2+1}{2} = 1.5, \quad y = \frac{5+2}{2} = 3.5 \Rightarrow C3 = (1.5, 3.5)$$

Final Answer

After 2 iterations, the final cluster centers are:

- Cluster 1: (3, 9.5)
- Cluster 2: (6.5, 5.25)
- Cluster 3: (1.5, 3.5)

4.4 Clustering Mixed-Type Data: K-Means and Distance Metrics

Handling **mixed-type data** (numerical + categorical variables) requires specialized approaches since standard K-means and Euclidean distance are incompatible with categorical features.

Data Preprocessing & Distance Metrics

One-Hot Encoding + Euclidean Distance

- **Approach**: Convert categorical variables to binary columns
- **Issues**:
  - High dimensionality and sparsity distort distance
  - Centroids become non-interpretable (fractional values)

– Suitable only for low-cardinality categorical features

4.4.1 Gower Distance

Ideal unified metric for mixed data:

- **Numerical**: Manhattan distance normalized by range
- **Categorical**: Dice similarity (1 if same, 0 if different)

$$d_{\text{Gower}}(\mathbf{x_i}, \mathbf{x_j}) = \frac{\sum_{k=1}^{p} w_k \delta_{ijk}}{\sum_{k=1}^{p} w_k}$$

where δ_{ijk} is feature-specific distance and w_k is optional weight.

Advantages: Natural mixed-type handling, normalized [0,1] distances
Limitation: $O(n^2)$ computational complexity

Other Metrics

- **Hamming Distance**: Mismatch count for categorical features
- **Weighted Combinations**: Custom blends (e.g., Euclidean + Hamming)

Clustering Algorithms

4.4.2 K-Prototypes

Hybrid of K-Means (numerical) and K-Modes (categorical):

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{k \in \text{num}} (x_{ik} - x_{jk})^2 + \gamma \sum_{k \in \text{cat}} \mathbb{1}(x_{ik} \neq x_{jk})$$

where γ balances feature-type contributions.

Implementation:

```
from kmodes.kprototypes import KPrototypes
# Preprocess: Scale numericals, encode categoricals as integers
kp = KPrototypes(n_clusters=3, init='Cao')
clusters = kp.fit_predict(data, categorical=[0, 2])
```

K-Medoids with Gower Distance

PAM algorithm using actual data points as centroids:

- Robust to noise
- Compatible with any distance matrix

Implementation:

```
import gower
from sklearn_extra.cluster import KMedoids
dist_matrix = gower.gower_matrix(data)
kmed = KMedoids(n_clusters=3, metric="precomputed")
labels = kmed.fit_predict(dist_matrix)
```

FAMD + K-Means

Dimensionality reduction for mixed data:

- FAMD projects data into continuous latent space
- Preserves variance in both feature types

Implementation:

```
from prince import FAMD
from sklearn.cluster import KMeans

famd = FAMD(n_components=5).fit(data)
kmeans = KMeans(n_clusters=3).fit(famd.row_coordinates_)
```

Practical Considerations

4.4.3 Preprocessing

- Scale numerical features (e.g., StandardScaler)
- Encode categoricals as integers or one-hot

Balancing Feature Types

- Use γ parameter in K-Prototypes
- Apply weights in Gower distance

Cluster Validation

- Silhouette Score with Gower distance
- Inspect cluster characteristics (categorical modes)

Limitations

- K-Prototypes assumes spherical clusters
- Gower distance scales poorly for large datasets
- No universal solution - requires experimentation

Method Comparison

Table 4.2: Summary of Clustering Methods for Mixed-Type Data

Method	**Distance Metric**	**Algorithm**	**Scalability**
K-Prototypes	Hybrid (Euclidean + Dissimilarity)	Custom	High
Gower + K-Medoids	Gower	PAM	Low
One-Hot + K-Means	Euclidean	K-Means	Medium
FAMD + K-Means	Euclidean (latent space)	K-Means	Medium

Recommendations

- **K-Prototypes**: Best for efficiency and implementation simplicity
- **Gower + K-Medoids**: Preferred for accuracy in small-to-medium datasets
- Always validate with domain knowledge and multiple metrics

4.5 Logistic Regression

What is Logistic Regression?

Logistic Regression is a supervised machine learning algorithm used primarily for classification tasks.

Unlike Linear Regression, which predicts continuous values, Logistic Regression estimates the probability that a given input belongs to a particular class.

It is especially useful for binary classification problems where the target variable has only two classes:

- Yes / No
- 0 / 1
- True / False

Types of Logistic Regression

Type	Description
Binomial Logistic Regression	Target variable has two categories (e.g., Yes/No)
Multinomial Logistic Regression	Target has 3+ unordered categories (e.g., cat/dog/sheep)
Ordinal Logistic Regression	Target has 3+ ordered categories (e.g., Low/Med/High)

Assumptions of Logistic Regression

- **Independent Observations:** Each sample is independent.
- **Binary Output (for binary case):** Target variable should be 0 or 1.
- **Linearity of Logit:** Linear relationship between predictors and log-odds.
- **No Outliers:** Outliers can distort parameter estimation.
- **Sufficient Sample Size:** Needed for stability and accuracy.

The Sigmoid Function

The sigmoid function transforms any real-valued number into a value between 0 and 1:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\lim_{z\to\infty} \sigma(z) = 1, \quad \lim_{z\to-\infty} \sigma(z) = 0$$

Decision Rule:

$$\sigma(z) \geq 0.5 \Rightarrow \text{Class 1}; \quad \text{Else Class 0}$$

Mathematical Derivation

Step 1: Linear Function

$$z = w \cdot X + b$$

Where:

- X = feature vector
- w = weights
- b = bias/intercept

Step 2: Apply Sigmoid

$$p(X) = \sigma(z) = \frac{1}{1+e^{-z}}$$

This gives the probability of belonging to Class 1.

Logistic Regression as Log-Odds

Odds:

$$\text{Odds} = \frac{p(x)}{1-p(x)} = e^{z}$$

Log-Odds (Logit):

$$\log\left(\frac{p(x)}{1-p(x)}\right) = w \cdot X + b$$

Likelihood Function

Likelihood:

$$L(w,b) = \prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$$

Log-Likelihood:

$$\log L(w,b) = \sum_{i=1}^{n} [y_i \log(p(x_i)) + (1-y_i)\log(1-p(x_i))]$$

Gradient for Optimization (Gradient Ascent)

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{n}(y_i - p(x_i))x_{ij}$$

Weight Update Rule:

$$w_j := w_j + \alpha \cdot \frac{\partial J}{\partial w_j}$$

Softmax in Multiclass Logistic Regression

Given K classes, the softmax function:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Final probability for class c:

$$P(y = c \mid x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{k=1}^{K} e^{w_k \cdot x + b_k}}$$

Logistic vs Linear Regression

Aspect	Linear Regression	Logistic Regression
Output	Continuous value	Probability (0 to 1)
Type of Problem	Regression	Classification
Estimation Technique	Least Squares	Maximum Likelihood
Model	Best-fit line	Sigmoid curve (S-shaped)
Output Range	$(-\infty, +\infty)$	$(0, 1)$
Assumptions	Linearity between X and Y	Linearity between X and log-odds

Common Terminologies

Term	Description
Independent Variables (X)	Input features or predictors
Dependent Variable (Y)	Target class label (binary or multiclass)
Odds	$\frac{P}{1-P}$
Logit	$\log\left(\frac{P}{1-P}\right)$
Coefficients	Weights showing effect of input features
Intercept	Constant term in the logit equation
MLE	Maximum Likelihood Estimation for parameter learning

Evaluation Metrics

Metric	Formula	Use Case
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	General performance
Precision	$\frac{TP}{TP+FP}$	Importance of positive class
Recall	$\frac{TP}{TP+FN}$	Catching all positive cases
F1-Score	$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$	Trade-off between P and R
AUC-ROC	Area under ROC Curve	Probabilistic ranking quality

Example: Numerical Calculation

Consider a dataset with a single feature:

x	y
2	0
4	0
6	1
8	1

Let the initial parameters be:

$$\theta_0 = -4, \quad \theta_1 = 1$$

Prediction for $x = 6$

Step 1: Compute z

$$z = \theta_0 + \theta_1 x = -4 + 1 \cdot 6 = 2$$

Step 2: Apply sigmoid function

$$h_\theta(x) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-2}} \approx \frac{1}{1 + 0.1353} \approx 0.88$$

Interpretation: There is an 88% probability the class is 1 $\Rightarrow$ **Predict 1**

Prediction for $x = 2$

Step 1: Compute z

$$z = \theta_0 + \theta_1 x = -4 + 1 \cdot 2 = -2$$

Step 2: Apply sigmoid function

$$h_\theta(x) = \frac{1}{1 + e^{-(-2)}} = \frac{1}{1 + e^{2}} \approx \frac{1}{1 + 7.389} \approx 0.119$$

Interpretation: About 11.9% chance the class is 1 $\Rightarrow$ **Predict 0**

Applications

Logistic Regression is widely used in various domains due to its simplicity and interpretability. Some common applications include:

- **Email Spam Detection**
- **Disease Prediction (e.g., Diabetes)**
- **Credit Risk Scoring**
- **Marketing Campaign Success Prediction**

Advantages and Disadvantages

Advantages

- Simple and fast to implement.
- Provides a probabilistic interpretation of the outcome.
- Works well with linearly separable data.

Disadvantages

- Assumes a linear decision boundary, which may not hold for complex datasets.
- May underperform when the relationship between features and target is non-linear or intricate.