Data Mining: Supervised Methods

Sunil Regmi, Lecturer, DoAI

June 12, 2025

Linear Regression

- Equation of linear regression: $Y = mx + b$
- Y represents the dependent variable
- X represents the independent variable
- m is the slope of the line (how much Y changes for a unit change in X)
- b is the intercept (the value of Y when X is 0)

Predicting Pizza Prices

1. Data Collection
2. Calculations
3. Prediction
4. Visualization
5. Evaluation

Data and Calculations

Diameter(X) In Inches	Price(Y) In Dollars	Mean(X)	Mean(Y)	Deviations(X)	Deviations(Y)	Product of Deviations	Square of Deviations for X
8	10	10	12.33	-2	-2.33	4.66	4
10	13	10	12.33	0	0.67	0	0
12	15	10	12.33	2	2.67	5.34	4

- Slope $m = \frac{10}{8} = 1.25$
- Intercept $b = 12.33 - (1.25 \cdot 10) = -0.17$
- Regression equation: $\hat{Y} = 1.25X - 0.17$

Prediction and Verification

- Predicted values:
  - $X = 8$: $\hat{Y} = 9.83$
  - $X = 10$: $\hat{Y} = 12.33$
  - $X = 12$: $\hat{Y} = 14.83$
- Deviations: $10 - 9.83 = 0.17$, $13 - 12.33 = 0.67$, $15 - 14.83 = 0.17$

Model Evaluation

- Mean Squared Error (MSE) $= \frac{0.0289+0.4489+0.0289}{3} \approx 0.169$
- R-squared $(R^2) = 1 - \frac{0.5067}{13.0067} \approx 0.961$
- Interpretation:
  - MSE indicates average error; low value suggests good fit.
  - $R^2 \approx 0.961$ means 96.1% of variance in price is explained by diameter.

Model Training Process (OLS)

Data Preparation:

- Dataset: $(8, 10), (10, 13), (12, 15)$
- Feature: X (Diameter), Target: Y (Price)
- Means: $\bar{X} = 10$, $\bar{Y} = 12.33$

Model Definition:

- Linear equation: $Y = mx + b$

Loss Function:

- $MSE = \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2$

Model Training Process (Contd.)

Optimization with OLS:

- Deviations from mean: $X - \bar{X}, Y - \bar{Y}$
- Slope: $m = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2} = 1.25$
- Intercept: $b = \bar{Y} - m \cdot \bar{X} = -0.17$

Model Fitting and Evaluation:

- Final model: $\hat{Y} = 1.25X - 0.17$
- MSE: 0.169, $R^2 = 0.961$

Machine Learning Perspective:

- Supervised Learning: Learn from labeled data
- OLS: Closed-form solution, fast training
- Generalization: Predict prices for unseen diameters

Model Training with Gradient Descent

Why Gradient Descent?

- OLS provides a direct formula, but gradient descent is useful for larger or more complex models.
- Gradient Descent is an iterative optimization algorithm to minimize the loss function.

Objective:

- Minimize the Mean Squared Error (MSE):

$$J(m, b) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - (mX_i + b))^2$$

Gradient Descent Steps in Our Pizza Price Example

1. **Initialize parameters:**
   - Start with arbitrary values, e.g., $m = 0$, $b = 0$
2. **Compute Gradients:**
   - Partial derivatives:

$$\frac{\partial J}{\partial m} = -\frac{2}{n}\sum X_i(Y_i - \hat{Y}_i) \quad , \quad \frac{\partial J}{\partial b} = -\frac{2}{n}\sum (Y_i - \hat{Y}_i)$$

3. **Update Rules:**
   - Use learning rate α, update:

$$m := m - \alpha\frac{\partial J}{\partial m}, \quad b := b - \alpha\frac{\partial J}{\partial b}$$

4. **Repeat:**
   - Until convergence (minimal change in loss)

Gradient Descent: Manual Step-by-Step

Initial Setup:

- Data: $(8, 10), (10, 13), (12, 15)$
- Initialize: $m = 0, b = 0$
- Learning Rate: $\alpha = 0.01$
- Hypothesis: $\hat{Y} = mX + b$

Step 1: Compute Predictions

$$\hat{Y}_i = 0 \text{ for all } i \quad \Rightarrow \text{Errors: } -10, -13, -15$$

Step 2: Compute Gradients

$$\frac{\partial J}{\partial m} = -\frac{2}{3}(8 \cdot (-10) + 10 \cdot (-13) + 12 \cdot (-15)) = \frac{2}{3}(80 + 130 + 180) = \frac{2}{3}(390)$$

$$\frac{\partial J}{\partial b} = -\frac{2}{3}(-10 - 13 - 15) = \frac{2}{3}(38) \approx 25.33$$

Gradient Descent: Update Parameters

Step 3: Update Parameters

$$m := 0 - 0.01 \cdot 260 = -2.6, \quad b := 0 - 0.01 \cdot 25.33 \approx -0.2533$$

Step 4: New Predictions

- For $X = 8$: $\hat{Y} = -2.6 \cdot 8 - 0.2533 = -20.0533$
- For $X = 10$: $\hat{Y} = -26.2533$
- For $X = 12$: $\hat{Y} = -32.4533$

Observation:

- Predictions went further from true values.
- Suggests the learning rate is too high — might need $\alpha = 0.001$
- Repeat steps until convergence (i.e., until MSE stops decreasing)

Logistic Regression

Supervised classification model

- Predicts binary outcomes (0 or 1)
- Suitable when the dependent variable is categorical and binary
- Learns a decision boundary using the sigmoid function

Example Dataset:

Study Hours	Exam Result
2	0
3	0
4	0
5	1
6	1
7	1
8	1

Model aims to estimate:

$$P(\text{Pass}) = \frac{1}{1 + e^{-(mX+b)}}$$

Sigmoid Function and Cross-Entropy Loss

- Logistic regression outputs probabilities:

$$\hat{y} = \frac{1}{1 + e^{-(mx+b)}}$$

- Binary Cross-Entropy Loss:

$$\mathcal{L}(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

Gradient Descent Updates

- Gradients:

$$\frac{\partial \mathcal{L}}{\partial m} = (\hat{y} - y) \cdot x \quad , \quad \frac{\partial \mathcal{L}}{\partial b} = (\hat{y} - y)$$

- Update rules:

$$m = m - \alpha \cdot \frac{\partial \mathcal{L}}{\partial m} \quad , \quad b = b - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b}$$

- α: Learning rate

Example: One Iteration (x = 5, y = 1)

- Initialize: $m = 0,\ b = 0,\ \alpha = 0.1$
- Compute $z = mx + b = 0$
- Sigmoid: $\hat{y} = \frac{1}{1+e^{-0}} = 0.5$
- Loss: $\mathcal{L} = -\log(0.5) = 0.693$
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial m} = -2.5, \quad \frac{\partial \mathcal{L}}{\partial b} = -0.5$$

- Updates:

$$m = 0.25, \quad b = 0.05$$

Summary of Logistic Regression Training

- Logistic Regression uses sigmoid function to model probabilities
- Loss minimized using cross-entropy
- Gradient Descent is used to update m and b
- Model generalizes to new input data to make binary predictions

k-Nearest Neighbors (k-NN)

- A simple, intuitive classification algorithm
- Classifies a data point based on the majority label of its k nearest neighbors
- Requires a distance metric (e.g., Euclidean distance)
- Non-parametric and lazy learning algorithm

Example: Predicting Movie Genre (k-NN)

IMDb Rating	Duration	Genre
8.0 (Mission Impossible)	160	Action
6.2 (Gadar 2)	170	Action
7.2 (Rocky & Rani)	168	Comedy
8.2 (OMG 2)	155	Comedy

Query: Predict the genre of "**Barbie**" with IMDb rating **7.4** and duration **114 min**

Distance Calculation (Euclidean)

Barbie: IMDb $= 7.4$, Duration $= 114$

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

- To (8.0, 160): $d = \sqrt{(7.4 - 8.0)^2 + (114 - 160)^2} \approx 46.003$
- To (6.2, 170): $d \approx 56.029$
- To (7.2, 168): $d \approx 54.000$
- To (8.2, 155): $d \approx 41.001$

Closest Neighbor: (8.2, 155) $\rightarrow$ Genre: Comedy

Prediction: Genre of "Barbie"

- If $k = 1$, the closest movie is (8.2, 155) $\rightarrow$ **Comedy**
- If $k = 3$, nearest movies:
  1. (8.2, 155) $\rightarrow$ Comedy
  2. (8.0, 160) $\rightarrow$ Action
  3. (7.2, 168) $\rightarrow$ Comedy
- Majority genre: **Comedy (2 out of 3)**

Therefore, Barbie is predicted as a Comedy movie.

k-NN: Summary

- Distance-based classification technique
- Sensitive to scale of features $\rightarrow$ normalization often needed
- Choice of k affects bias-variance trade-off:
  - Low k: more flexible, more variance
  - High k: more stable, more bias
- Works well with small datasets and intuitive interpretation

Bayes Theorem: Basics

- Bayes' Theorem allows us to update probabilities based on new evidence.
- Formula:

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

- Where:
  - $P(A|B)$ = Posterior probability
  - $P(B|A)$ = Likelihood
  - $P(A)$ = Prior
  - $P(B)$ = Evidence

Sample Dataset: Flu, COVID, Fever

Person	COVID	Flu	Fever
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	Yes	Yes
4	No	No	No
5	Yes	No	Yes
6	No	No	Yes
7	Yes	No	Yes
8	Yes	No	No
9	No	Yes	Yes
10	No	Yes	No

Prior and Conditional Probabilities

- Prior:

$$P(Fever = Yes) = \frac{7}{10}, \quad P(Fever = No) = \frac{3}{10}$$

- Conditional Probabilities:
  - $P(COVID = Yes|Fever = Yes) = \frac{4}{7}$
  - $P(COVID = Yes|Fever = No) = \frac{2}{3}$
  - $P(Flu = Yes|Fever = Yes) = \frac{3}{7}$
  - $P(Flu = Yes|Fever = No) = \frac{2}{3}$

Naive Bayes Classification

Goal: Predict Fever given Flu and COVID.

$$P(Fever = Yes|Flu, COVID) \propto P(Flu|Yes) \cdot P(COVID|Yes) \cdot P(Yes)$$

$$P(Fever = No|Flu, COVID) \propto P(Flu|No) \cdot P(COVID|No) \cdot P(No)$$

Example: Flu = Yes, COVID = Yes

$$P(Yes) = 0.7, \quad P(Flu|Yes) = 0.43, \quad P(COVID|Yes) = 0.57$$

$$P(Yes|Flu, COVID) \propto 0.43 \cdot 0.57 \cdot 0.7 \approx 0.172$$

$$P(No|Flu, COVID) \propto 0.67 \cdot 0.33 \cdot 0.3 \approx 0.066$$

Conclusion: Predict Fever = Yes

What is a Bayesian Network?

- A probabilistic graphical model representing conditional dependencies.
- Variables as nodes, dependencies as directed edges.
- Joint probability decomposed as:

$$P(COVID, Flu, Fever) = P(COVID) \cdot P(Flu) \cdot P(Fever|COVID, Flu)$$

How to Calculate a Conditional Probability Table (CPT)

- A CPT defines probabilities of a variable given its parents in the Bayesian Network.
- To calculate:
  1. Count how many times each parent configuration occurs.
  2. Count how often the child variable is `Yes` or `No` for each configuration.
  3. Compute conditional probability as:

$$P(\text{Child} = \text{Yes} \mid \text{Parents}) = \frac{\text{Count of Yes with Parents}}{\text{Total Count of Parents}}$$

- Example:

$$P(\text{Fever} = \text{Yes} \mid \text{COVID} = \text{Yes}, \text{Flu} = \text{No}) = \frac{3}{4}$$

Bayesian Network Structure and CPT

Structure:

$$COVID \rightarrow Fever \leftarrow Flu$$

Conditional Probability Table (CPT):

COVID	Flu	Fever	P(Fever)
Yes	Yes	Yes	1.0
Yes	No	Yes	0.75
Yes	No	No	0.25
No	Yes	Yes	0.67
No	No	Yes	0.5
No	No	No	0.5

Bayesian Network Inference Example

Query: $P(Fever = Yes | COVID = Yes, Flu = No)$
From CPT: Lookup directly

$$P(Fever = Yes | COVID = Yes, Flu = No) = 0.75$$

No need to assume independence as in Naive Bayes

Summary: Naive Bayes vs Bayesian Network

- **Naive Bayes:** Simpler, assumes conditional independence.
- **Bayesian Network:** Flexible, models dependencies explicitly.
- Both use Bayes' Theorem at their core.
- Dataset used to demonstrate both methods with Flu, COVID, and Fever.

What is a Decision Tree?

- A decision tree is a supervised machine learning algorithm used for classification and regression.
- It splits data recursively based on feature values to make decisions.
- Each internal node represents a test on a feature.
- Each branch represents the outcome of the test.
- Each leaf node represents a class label or regression value.

How Decision Trees Work

- At each node, choose the feature and split that best separates the data.
- Common splitting criteria:
  - Information Gain (based on entropy)
  - Gini Impurity
- Continue splitting until stopping conditions:
  - All samples in a node belong to the same class.
  - Maximum tree depth reached.
  - Minimum number of samples in a node.

Sample Sports Dataset

Person	Weather	Temperature	Play
1	Sunny	Hot	No
2	Sunny	Mild	Yes
3	Rainy	Hot	Yes
4	Rainy	Mild	Yes
5	Sunny	Hot	No

Calculate Entropy of Target Variable

Play distribution:

- Yes: 3
- No: 2

Entropy formula:

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

Calculate:

$$p(Yes) = \frac{3}{5}, \quad p(No) = \frac{2}{5}$$

$$H(S) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) = 0.971$$

Information Gain: Split by Weather

Weather values: Sunny, Rainy
Entropy for Sunny (3 samples: 1 Yes, 2 No):

$$H(Sunny) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = 0.918$$

Entropy for Rainy (2 samples: 2 Yes, 0 No):

$$H(Rainy) = -(1 \times \log_2 1 + 0) = 0$$

Weighted entropy after split:

$$H_{Weather} = \frac{3}{5} \times 0.918 + \frac{2}{5} \times 0 = 0.551$$

Information gain:

$$IG(Weather) = H(S) - H_{Weather} = 0.971 - 0.551 = 0.42$$

Information Gain: Split by Temperature

Temperature values: Hot, Mild

Entropy for Hot (3 samples: 1 Yes, 2 No):

$$H(Hot) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = 0.918$$

Entropy for Mild (2 samples: 2 Yes, 0 No):

$$H(Mild) = 0$$

Weighted entropy after split:

$$H_{Temp} = \frac{3}{5} \times 0.918 + \frac{2}{5} \times 0 = 0.551$$

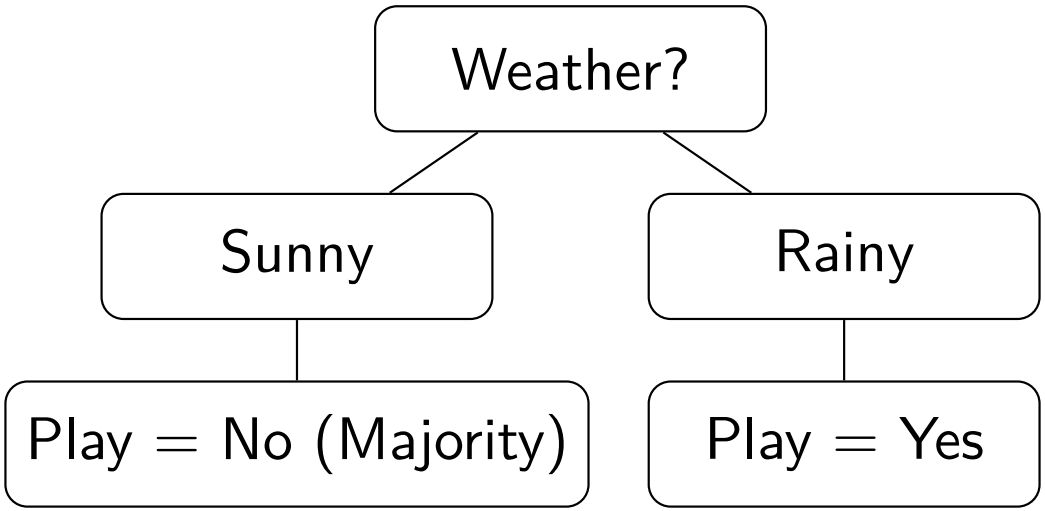
Information gain:

$$IG(Temp) = H(S) - H_{Temp} = 0.971 - 0.551 = 0.42$$

Both features have same IG here.

Building the Decision Tree

- Choose feature with highest information gain (Weather or Temperature, both 0.42 here).
- Suppose we choose **Weather** as root node.
- Split data into two groups: Sunny and Rainy.
- For Rainy branch: all Yes $\rightarrow$ leaf node "Play = Yes".
- For Sunny branch: further split or assign majority class.



Advantages and Disadvantages of Decision Trees

Advantages:

- Easy to understand and interpret.
- Handles both numerical and categorical data.
- Can model nonlinear relationships.

Disadvantages:

- Can easily overfit if not pruned.
- Unstable: small data changes can lead to different trees.
- Can be biased towards features with more levels.

Gini Impurity: An Alternative to Entropy

- Measures the probability of misclassification.
- Formula:

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2$$

  where p_i is the fraction of class i in node t.
- The goal is to find splits that minimize weighted Gini impurity.

Calculate Gini Impurity for Target Variable

Play distribution:

- Yes: 3
- No: 2

Calculate class probabilities:

$$p(Yes) = \frac{3}{5} = 0.6, \quad p(No) = \frac{2}{5} = 0.4$$

Gini impurity:

$$Gini(S) = 1 - (0.6^2 + 0.4^2) = 1 - (0.36 + 0.16) = 0.48$$

Gini Impurity for Split on Weather

Split by Weather:
- Sunny (3 samples): 1 Yes, 2 No

$$p(Yes) = \frac{1}{3}, p(No) = \frac{2}{3}$$

$$Gini(Sunny) = 1 - \left(\frac{1}{3}^2 + \frac{2}{3}^2\right) = 1 - \left(\frac{1}{9} + \frac{4}{9}\right) = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

- Rainy (2 samples): 2 Yes, 0 No

$$Gini(Rainy) = 1 - (1^2 + 0) = 0$$

Weighted Gini:

$$Gini_{Weather} = \frac{3}{5} \times 0.444 + \frac{2}{5} \times 0 = 0.266$$

What is a Random Forest?

- An ensemble learning method for classification and regression.
- Combines multiple decision trees to improve accuracy and control overfitting.
- Each tree is trained on a random subset (bootstrap sample) of the data.
- At each split, a random subset of features is considered.
- Final prediction by majority voting (classification) or averaging (regression).

Advantages of Random Forests

- Reduces variance compared to a single decision tree.
- Robust to noise and overfitting.
- Works well with high-dimensional data.
- Can estimate feature importance.

Random Forest Algorithm Steps

1. For each tree:
   - Draw a bootstrap sample from the training data.
   - Build a decision tree using a random subset of features at each split.
2. Repeat to build multiple trees (e.g., 100 trees).
3. Aggregate predictions from all trees:
   - Classification: majority vote.
   - Regression: average prediction.

Example Dataset Recap

Person	Weather	Temperature	Play
1	Sunny	Hot	No
2	Sunny	Hot	No
3	Rainy	Mild	Yes
4	Sunny	Mild	Yes
5	Rainy	Mild	Yes

Random Forest Intuition

- Each decision tree might pick different splits because of random samples and feature subsets.
- Trees can disagree, but ensemble voting leads to more reliable predictions.
- For example, predicting if a person will play based on weather and temperature:
  - Tree 1 might focus on Weather.
  - Tree 2 might focus on Temperature.
- Final prediction: majority vote across trees.

Random Forest: Numerical Example Setup

Dataset:

Person	Weather	Temperature	Play
1	Sunny	Hot	No
2	Sunny	Hot	No
3	Rainy	Mild	Yes
4	Sunny	Mild	Yes
5	Rainy	Mild	Yes

Goal: Predict if "Play" = Yes or No based on Weather and Temperature.

Step 1: Create Bootstrap Samples

Randomly sample with replacement to create training subsets for each tree:

- Tree 1 Sample: Persons 1, 3, 4, 4, 5
- Tree 2 Sample: Persons 2, 3, 3, 5, 5
- Tree 3 Sample: Persons 1, 2, 4, 5, 5

Note: Some samples repeat due to replacement.

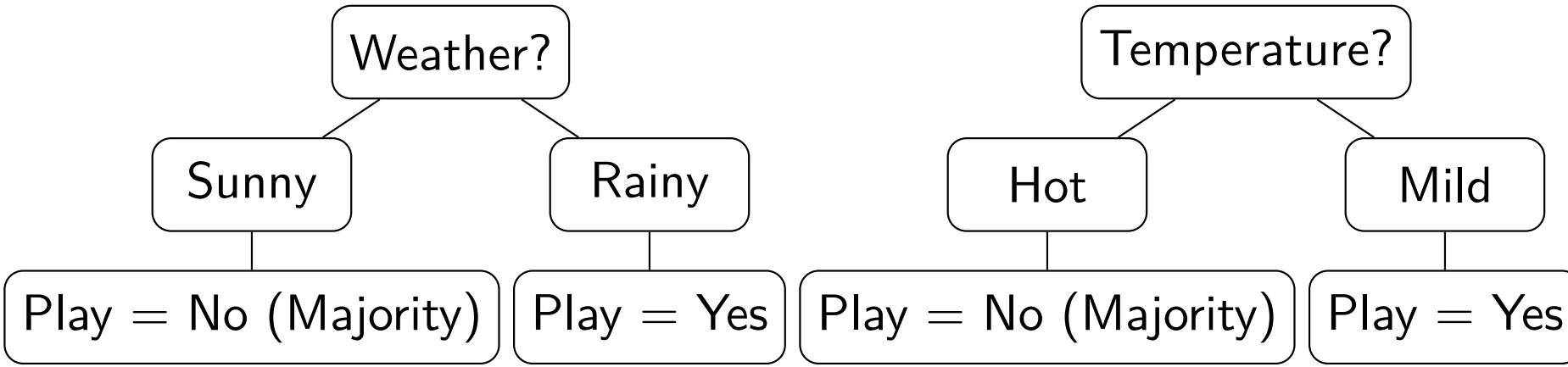
Step 2: Random Feature Selection

At each split, randomly select a subset of features to consider.
Assume: Only one feature is selected at random per split.
Example for Tree 1:

- First split: feature = Weather
- Split on Weather: Sunny vs Rainy

Example for Tree 2:

- First split: feature = Temperature
- Split on Temperature: Hot vs Mild



Step 3: Build Trees and Predict

Tree 1:

- Weather = Sunny $\Rightarrow$ Play = No (majority)
- Weather = Rainy $\Rightarrow$ Play = Yes (majority)

Tree 2:

- Temperature = Hot $\Rightarrow$ Play = No
- Temperature = Mild $\Rightarrow$ Play = Yes

Tree 3:

- Weather = Sunny $\Rightarrow$ Play = No
- Weather = Rainy $\Rightarrow$ Play = Yes

Step 4: Aggregate Predictions by Majority Voting

Predict Play for a new example:

Weather = Sunny, Temperature = Mild

- Tree 1 predicts: No (Sunny $\rightarrow$ No)
- Tree 2 predicts: Yes (Mild $\rightarrow$ Yes)
- Tree 3 predicts: No (Sunny $\rightarrow$ No)

Final prediction: No (2 out of 3 trees vote No)

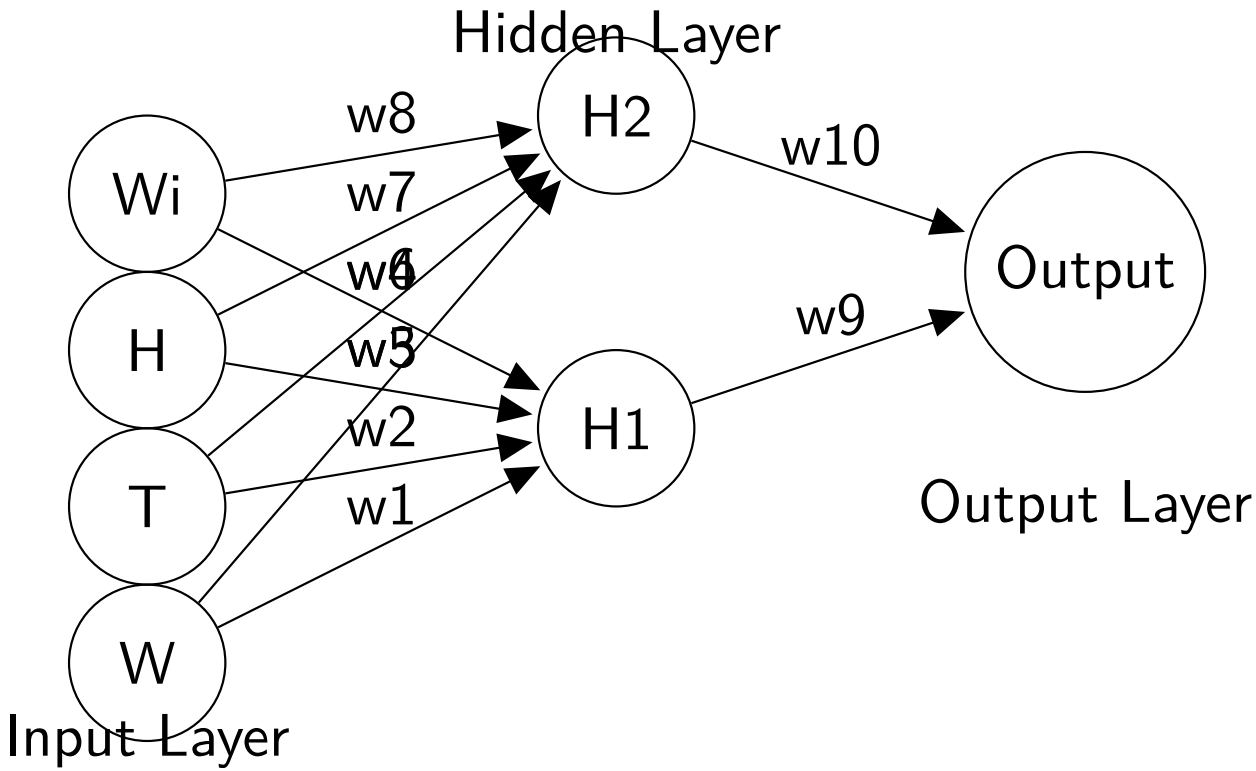
Summary

- Random Forest creates multiple trees with different samples and features.
- Each tree makes a prediction independently.
- Final output is based on majority voting (classification).
- This reduces overfitting and increases robustness.

Deep Introduction to ANN

- ANNs are inspired by the human brain's neural networks.
- Comprise layers: Input, Hidden, Output.
- Key components:
  - Neurons with weights and biases.
  - Activation functions (e.g., Sigmoid, ReLU).
  - Learning via backpropagation and gradient descent.
- Applications: Regression, Classification.

ANN Structure Visualization



- 4 inputs: Weather, Temp, Humidity, Wind.
- 2 hidden neurons for feature processing.
- 1 output for prediction.

Dataset for ANN

Day	Weather	Temperature	Humidity	Wind	Play Score / Play?
1	1 (Sunny)	1 (Hot)	1 (High)	0 (Weak)	0.2 / 0
2	1 (Sunny)	1 (Hot)	1 (High)	1 (Strong)	0.3 / 0
3	0 (Cloudy)	1 (Hot)	1 (High)	0 (Weak)	0.7 / 1
4	-1 (Rain)	0 (Mild)	1 (High)	0 (Weak)	0.6 / 1
5	-1 (Rain)	-1 (Cool)	-1 (Normal)	0 (Weak)	0.8 / 1

- Encoded: Sunny=1, Cloudy=0, Rain=-1; Hot=1, Mild=0, Cool=-1; High=1, Normal=-1; Weak=0, Strong=1.
- Regression target: Play Score (0 to 1).
- Classification target: Play? (0/1).

ANN for Regression

- **Task**: Predict Play Score for Day 1 (1, 1, 1, 0).
- **Initial Weights**: $w_1 = 0.2, w_2 = 0.3, w_3 = 0.1, w_4 = 0.4$, $b_1 = 0.1$ (H1); $w_5 = 0.1, w_6 = 0.2, w_7 = 0.3, w_8 = 0.2$, $b_2 = 0.2$ (H2); $w_9 = 0.5, w_{10} = 0.5$, $b_3 = 0.1$ (Output).
- **Forward Pass**:
  - H1: $z_1 = (1 \cdot 0.2) + (1 \cdot 0.3) + (1 \cdot 0.1) + (0 \cdot 0.4) + 0.1 = 0.7$
  - $a_1 = \sigma(0.7) \approx 0.67$
  - H2: $z_2 = (1 \cdot 0.1) + (1 \cdot 0.2) + (1 \cdot 0.3) + (0 \cdot 0.2) + 0.2 = 0.6$
  - $a_2 = \sigma(0.6) \approx 0.65$
  - Output: $z_3 = (0.67 \cdot 0.5) + (0.65 \cdot 0.5) + 0.1 \approx 0.765$
  - $a_3 = \sigma(0.765) \approx 0.68$
- Target $= 0.2$, Predicted $= 0.68$.

Regression: Loss and Backpropagation

- **Loss (MSE)**: $L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(0.2 - 0.68)^2 \approx 0.1156$
- **Gradient Descent** (Learning rate $\eta = 0.1$):
  - $\frac{\partial L}{\partial a_3} = (a_3 - y) \cdot a_3(1 - a_3) = (0.68 - 0.2) \cdot 0.68 \cdot 0.32 \approx 0.0109$
  - $\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial a_3} \cdot a_1 \approx 0.0109 \cdot 0.67 \approx 0.0073$
  - Update $w_9 = w_9 - \eta \cdot \frac{\partial L}{\partial w_9} = 0.5 - 0.1 \cdot 0.0073 \approx 0.4993$
  - Similarly for other weights (simplified here).
- Repeat for all weights and biases.

ANN for Classification

- **Task**: Predict Play? for Day 1 (1, 1, 1, 0).
- Same network, output interpreted as probability (0.68).
- Threshold: $> 0.5 = 1$ (Yes), $\quad 0.5 = 0$ (No).
- Target $= 0$, Predicted $= 1$ (error).
- Loss (Cross-Entropy): $L = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \approx -[0 \cdot \log(0.68) + 1 \cdot \log(1 - 0.68)] \approx 0.385$

Classification: Backpropagation

- **Gradient (Cross-Entropy)**: $\frac{\partial L}{\partial a_3} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \approx \frac{0.68-0}{0.68 \cdot 0.32} \approx 3.13$
- $\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial a_3} \cdot a_1 \approx 3.13 \cdot 0.67 \approx 2.097$
- Update $w_9 = 0.5 - 0.1 \cdot 2.097 \approx 0.2797$
- Adjust other weights similarly.

Support Vector Machines (SVMs)

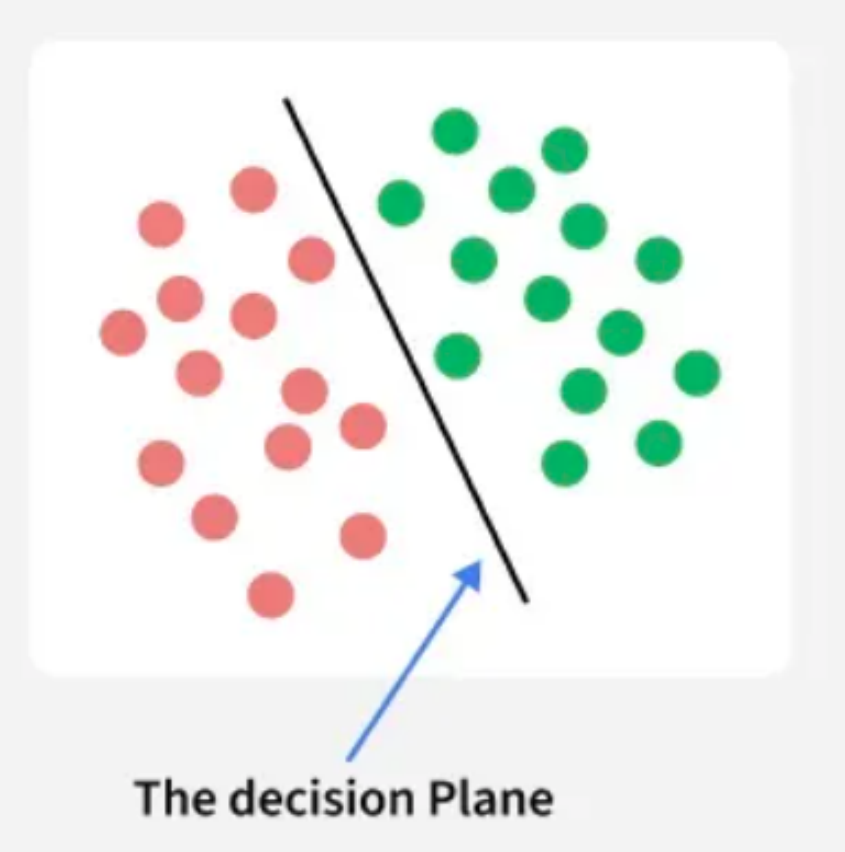
- **Support Vector Machines (SVMs)** are supervised machine learning algorithms used for classification and regression tasks.
- **Hyperplane:** A hyperplane is a decision boundary that separates data points of different classes in the feature space.
- **Mathematical Representation:** $\mathbf{w} \cdot \mathbf{x} + b = 0$

Example:
If we classify animals into cats $(+1)$ and dogs (-1) based on weight and height, the hyperplane could be:

$$3 \times \text{Weight} + 2 \times \text{Height} - 50 = 0$$

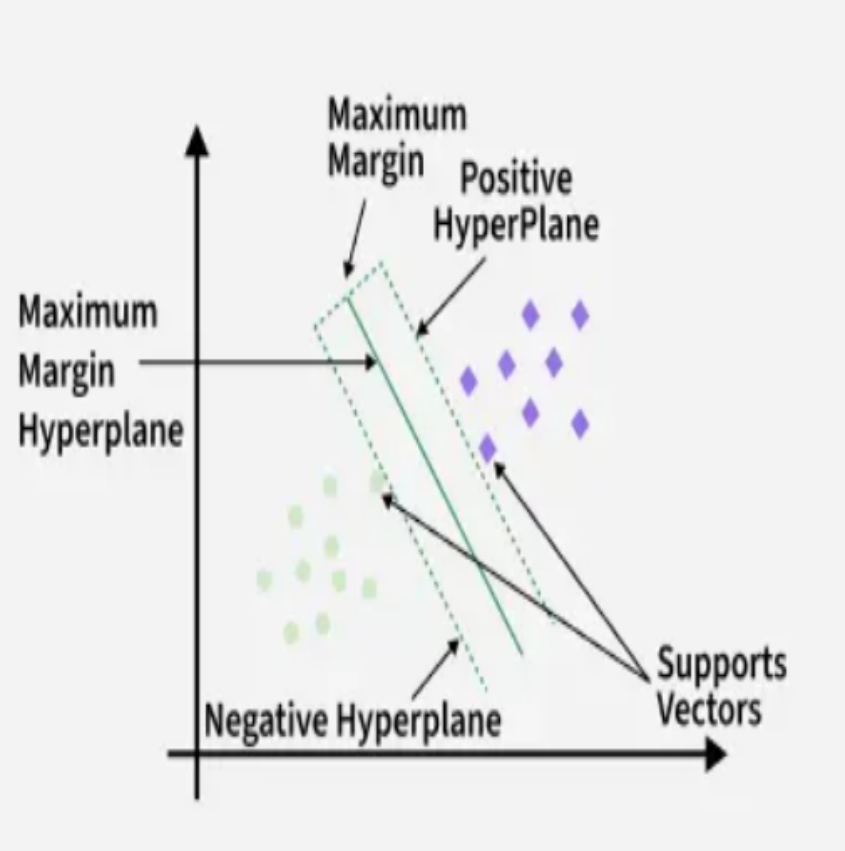
What is SVM?

- SVM (Support Vector Machine) is a supervised machine learning algorithm used for classification and regression tasks.
- SVM is based on the concept of decision planes that define decision boundaries.
- A decision plane is one that separates between a set of objects having different class memberships.

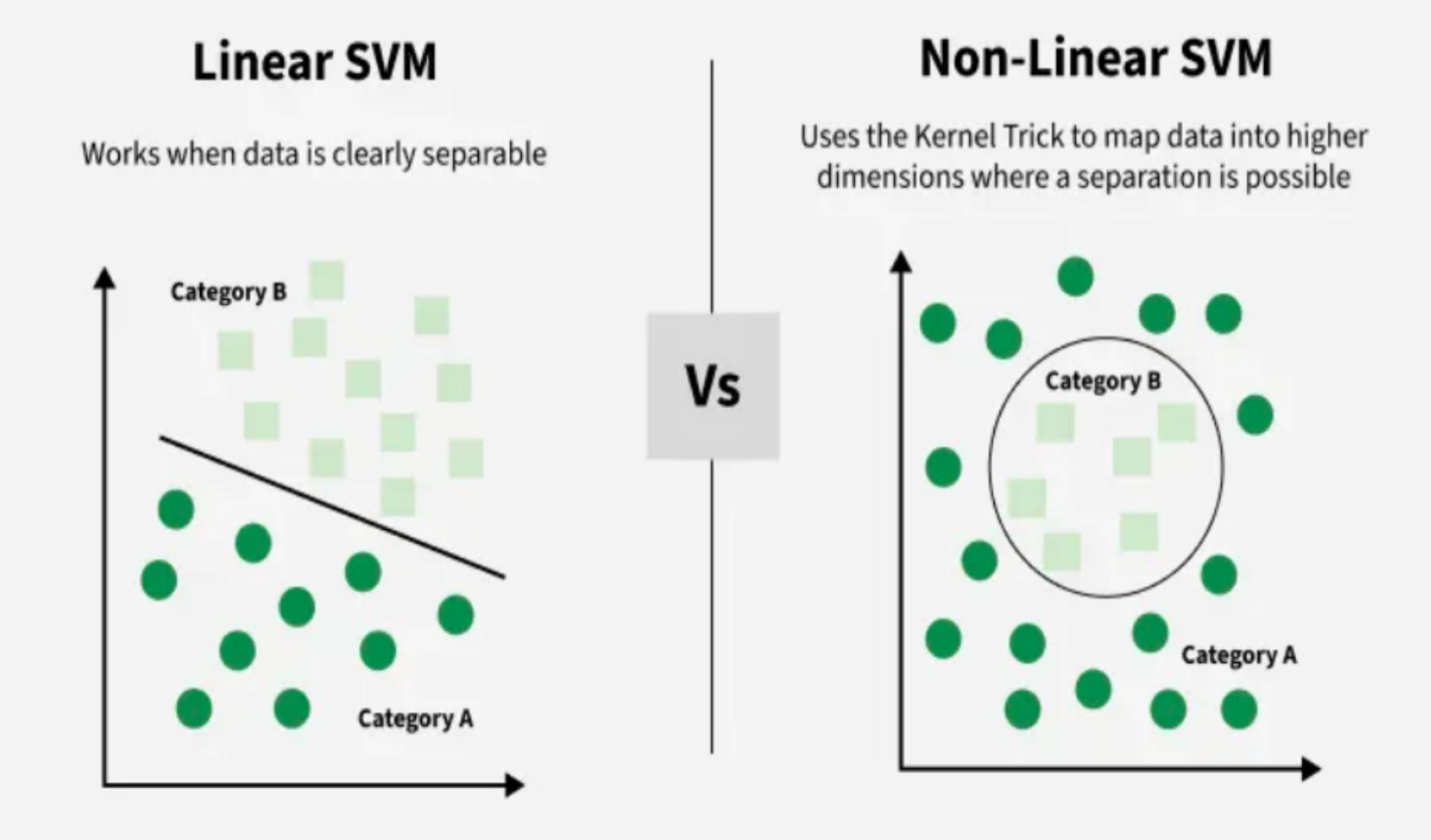


Support Vectors & Hyperplane

- **Hyperplane:** A line or plane that separates classes.
- **Support Vectors:** The closest data points to the hyperplane. These points define the boundary and help maximize the margin between classes.



Linear SVM vs Non-Linear SVM



Advantages and Disadvantages of SVM

Advantages

- ✓It works well with a clear margin of separation.
- ✓It is effective in high-dimensional spaces.
- ✓It is effective in cases where the number of dimensions is greater than the number of samples.

Disadvantages

- It doesn't perform well when we have large data set because the required training time is higher.
- It also doesn't perform very well when the data set has more noise, i.e., target classes are overall.
- SVM doesn't directly provide probability estimates; these are calculated using expensive five-fold cross-validation.

SVM: Core Concepts

- **Support Vectors:** These are the data points that are closest to the hyperplane. These points influence the orientation and position of the hyperplane.
- **Margin:** The margin is the distance between the hyperplane and the nearest data point of either class. SVM tries to maximize this margin to improve classification confidence.
- **Hard Margin:** Assumes data is perfectly separable by the hyperplane. All points must lie outside the margin.
- **Soft Margin:** Allows some misclassification or margin violations for non-linearly separable data.
- **Kernel Function:** A kernel function transforms data into a higher dimensional space to make it linearly separable.

Types of Kernels

- **Linear Kernel:** Suitable for linearly separable data.
- **Polynomial Kernel:** For curved boundaries.
- **Radial Basis Function (RBF) Kernel:** Captures complex relationships.

Our Dataset (Play Football)

Weather and Temperature affect whether we play football.

Weather	Temperature	Play
Sunny	Hot	No (-1)
Sunny	Mild	Yes (+1)
Rainy	Hot	Yes (+1)
Rainy	Mild	Yes (+1)
Sunny	Hot	No (-1)

We want to build a model to predict Play or Not.

Step 1: Encoded Dataset

Feature Encoding:

- x_1: Weather (Sunny = 1, Rainy = 2)
- x_2: Temperature (Hot = 1, Mild = 2)
- y: Play (Yes = +1, No = -1)

x_1	x_2	y
1	1	-1
1	2	+1
2	1	+1
2	2	+1
1	1	-1

Step 2: Choose Support Vectors

Support Vectors (on boundary):

$$\mathbf{x}^{(1)} = [1, 1], \; y^{(1)} = -1 \quad \text{(Sunny, Hot)}$$

$$\mathbf{x}^{(2)} = [1, 2], \; y^{(2)} = +1 \quad \text{(Sunny, Mild)}$$

We'll build a separating hyperplane using these.

Step 3: Weight Vector Calculation

We assume $\vec{w} = [w_1, w_2]$
Let's find $\vec{w}$ by subtracting vectors:

$$\vec{w} = \mathbf{x}^{(2)} - \mathbf{x}^{(1)} = [1, 2] - [1, 1] = [0, 1]$$

This means: decision depends only on x_2 (Temperature)

Step 4: Compute Bias b

We use the equation:

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) = 1$$

Using $\vec{w} = [0, 1]$
From $\mathbf{x}^{(1)} = [1, 1], y^{(1)} = -1$:

$$-1(0 \cdot 1 + 1 \cdot 1 + b) = 1 \Rightarrow -(1 + b) = 1 \Rightarrow b = -2$$

Step 5: Final SVM Equation

Now we know:

$$\vec{w} = [0, 1], \quad b = -2$$

Hyperplane:

$$f(x) = w_1 x_1 + w_2 x_2 + b = 0x_1 + 1x_2 - 2 = 0 \Rightarrow x_2 = 2$$

Decision Rule:

$$x_2 > 2 \Rightarrow +1 \quad \text{(Play)}$$

$$x_2 < 2 \Rightarrow -1 \quad \text{(Don't Play)}$$

Step 6: Predict Using Hyperplane

Test: Weather = Rainy (2), Temperature = Hot (1)

$$f(x) = 0 \cdot 2 + 1 \cdot 1 - 2 = -1 \Rightarrow \text{Predict: No (-1)}$$

Test: Weather = Sunny (1), Temperature = Mild (2)

$$f(x) = 0 \cdot 1 + 2 - 2 = 0 \Rightarrow \text{On boundary: Yes (+1)}$$

Test: Weather = Rainy (2), Temp = Mild (2)

$$f(x) = 2 - 2 = 0 \Rightarrow \text{Predict: Yes (+1)}$$

Model Evaluation

- **Overview:** Model evaluation assesses how well a machine learning model performs on unseen data, ensuring it generalizes beyond the training set.
- **Importance:** Critical for validating model reliability, avoiding overfitting, and guiding improvements.
- **Topics Covered:**
  - Training and Testing
  - Holdout, Cross-validation, Leave-One-Out Cross-validation, Bootstrap
  - Accuracy, Precision, Recall
  - ROC Curves, Recall-Precision Curves
  - Loss Functions
- Goal: Achieve a balance between bias and variance for robust predictions.

Training and Testing

- **Training Phase:** The model learns patterns from a labeled dataset, adjusting parameters to minimize error (e.g., using gradient descent).
- **Testing Phase:** Evaluates performance on a separate, unseen dataset to estimate generalization.
- **Process Details:**
  - Typical split: 70-80% for training, 20-30% for testing.
  - Example: Train on 700 samples, test on 300 samples from a 1000-sample dataset.
  - Random splitting ensures representativeness, but stratification is key for imbalanced data.
- **Challenges:** Small datasets may lead to high variance in test results; data leakage can inflate performance.
- Solution: Use multiple evaluation methods for validation.

Holdout, Cross-validation, Leave-One-Out, Bootstrap

- **Holdout Method:** Splits data into a single training and test set (e.g., 80/20). Simple but sensitive to split randomness.
- **Cross-validation:** Divides data into k folds (e.g., 5 or 10). Trains on $k-1$ folds, tests on the remaining fold, repeating k times. Average performance across folds.
  - Example: 5-fold CV with 1000 samples means 800 train, 200 test per fold.
- **Leave-One-Out Cross-validation (LOO):** Uses $n-1$ samples for training, 1 for testing, repeated for all n samples. Computationally expensive but unbiased for small datasets.
- **Bootstrap:** Resamples data with replacement to create multiple training sets, testing on out-of-bag samples. Useful for estimating confidence intervals.
- Advantage: These methods maximize data usage and reduce overfitting risk.

Accuracy, Precision, Recall

- **Accuracy:** Fraction of correct predictions. Formula: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.
  - Example: 80/100 correct predictions = 80% accuracy.
- **Precision:** Proportion of positive predictions that are correct. Formula: Precision $= \frac{TP}{TP+FP}$.
- Example: 50/60 positive predictions correct = 83.3% precision.
- **Recall:** Proportion of actual positives correctly identified. Formula: Recall $= \frac{TP}{TP+FN}$.
- Example: 50/70 actual positives found = 71.4% recall.
- **Trade-off:** Increasing precision may decrease recall, depending on the classification threshold.
- Limitation: Accuracy is misleading in imbalanced datasets (e.g., 90% negative class inflates accuracy).

ROC Curves, Recall-Precision Curves

- **ROC Curve:** Plots True Positive Rate (Recall) vs. False Positive Rate ($\frac{FP}{FP+TN}$) across different thresholds.
  - AUC-ROC (Area Under Curve) ranges from 0.5 (random) to 1.0 (perfect).
  - Example: AUC = 0.9 indicates strong model performance.
- **Recall-Precision Curve:** Plots Precision vs. Recall, highlighting trade-offs at various thresholds. Preferred for imbalanced data.
  - Example: High recall with low precision may indicate over-prediction of positives.
- **Visualization:**

- Insight: ROC is threshold-invariant; use with care in highly imbalanced cases.

Loss Functions

- **Definition:** A loss function quantifies the error between predicted and actual values, guiding model optimization.
- **Common Examples:**
  - **Mean Squared Error (MSE):** $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Used in regression to penalize larger errors quadratically.
    - Example: Predict 5, actual 7, MSE contribution = $(7\text{-}5)^2 = 4$.
  - **Cross-Entropy Loss:** $L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$. Common in classification.
  - **Hinge Loss:** $L = \max(0, 1 - y_i \cdot \hat{y}_i)$. Used in SVM to maximize margin.
- **Role:** Minimized during training using gradient-based methods (e.g., SGD).
- **Choice:** Depends on task—MSE for regression, cross-entropy for binary/multiclass, hinge for SVM.
- Consideration: Regularization (e.g., L2 penalty) can be added to prevent overfitting.

Any Queries???