Modules 1: Logistic Regression

1. Supervised Learning

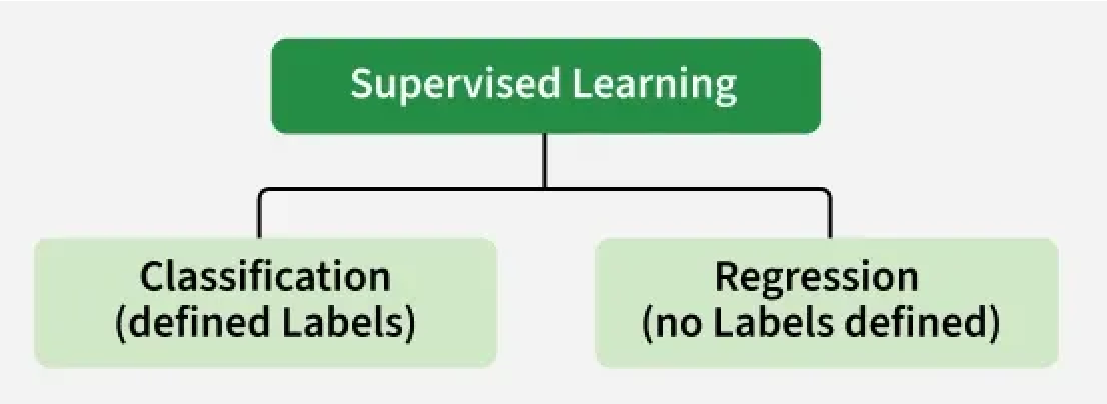
Definition:

A machine learning paradigm where the model learns from **labeled data** to predict new outputs.

Two main types:

Type	Output	Examples
Regression	Continuous variable	Predict house price, temperature, revenue
Classification	Categorical variable	Predict yes/no, male/female, spam/not spam

---



2. Classification Problem

Goal:

Assign an input to one of a set of predefined classes.

Example:

A flower shop wants to predict which type of flower a customer will buy based on past purchase history.

Requirements:

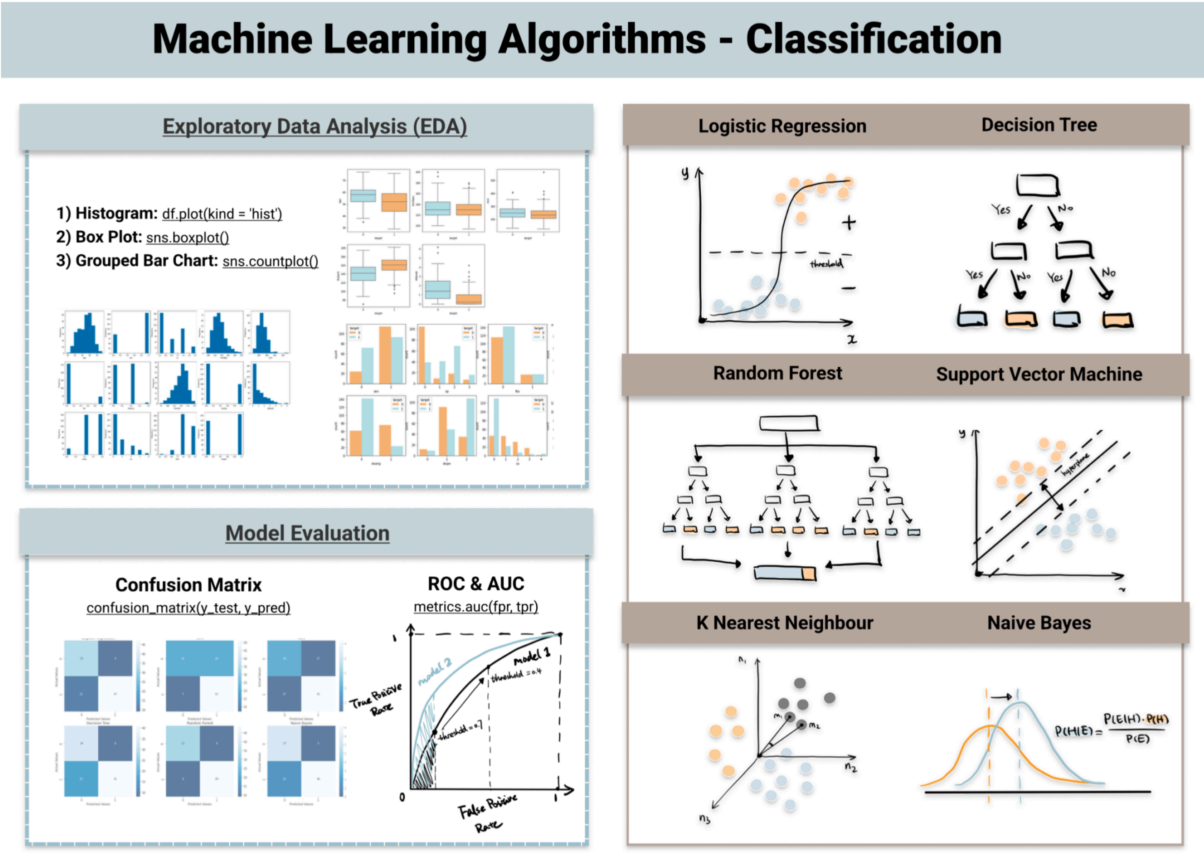
- **Features (X):** measurable attributes such as age, income, behavior.
- **Labels (y):** known targets such as class of flower or yes/no.
- **Measure of similarity:** a method to compare similarity between samples.

---

3. Common Models for Classification

Model	Main characteristics
Logistic Regression	Interpretable, uses sigmoid to predict probabilities
KNN	Based on distances between points
SVM	Finds the best separating hyperplane
Decision Tree / Random Forest	Tree-based, visually interpretable
Neural Networks	Powerful nonlinear models
Boosting / Ensemble	Combine multiple models to improve performance

---



4. Introduction to Logistic Regression

Purpose:

- Predict the **probability** that an observation belongs to class 1 (P(y=1|x)).
- Used for **binary classification** problems.

Comparison:

Aspect	Linear Regression	Logistic Regression
Prediction	Real value (−∞, +∞)	Probability (0–1)
Activation	None	Sigmoid
Problem type	Regression	Classification

5. Logistic Regression Formulas

5.1. Linear function

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Role:

- Combine input features into a single value z — the input to the sigmoid.

Parameter interpretation:

Symbol	Role	Meaning
β_0	Intercept	Log-odds when all $x_i = 0$
β_i	Weight	Effect size of feature x_i
x_i	Feature	Input data
z	Logit	Weighted sum, linear score

5.2. Sigmoid (logistic) function

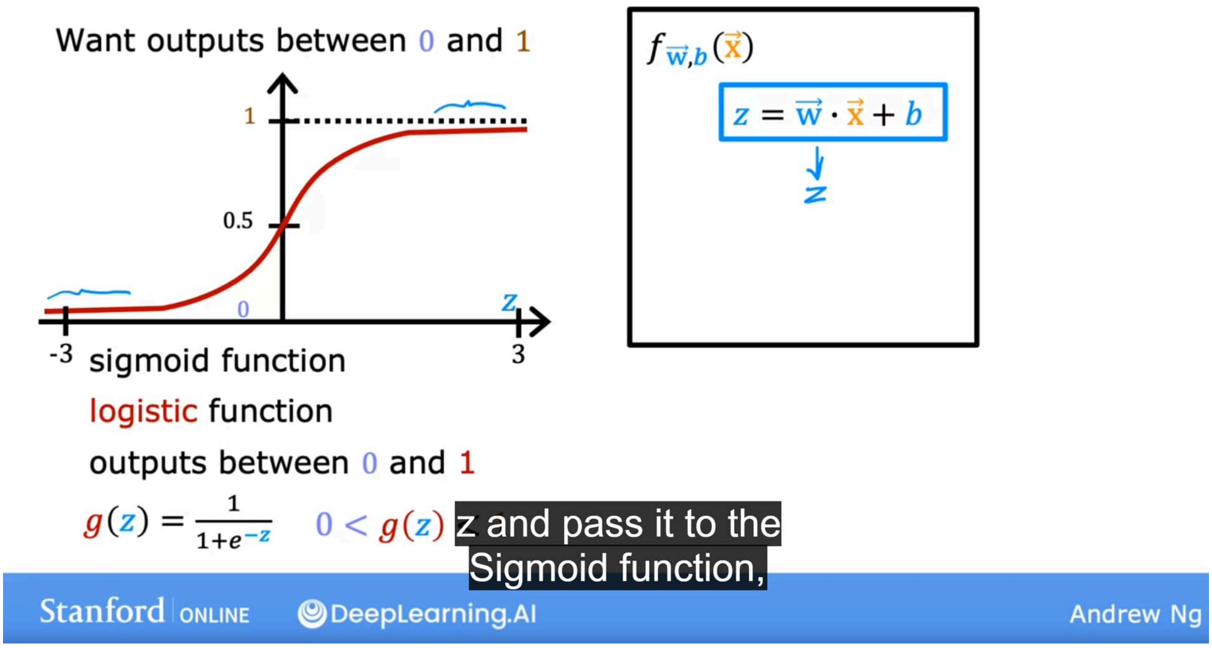
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Role:

- Map z from (−∞, +∞) to a **probability in [0,1]**.
- Lets the model interpret outputs as the chance of class 1.

Properties:

- $z \to +\infty \to \sigma(z) \to 1$
- $z \to -\infty \to \sigma(z) \to 0$
- $\sigma(0) = 0.5$ as the decision boundary.

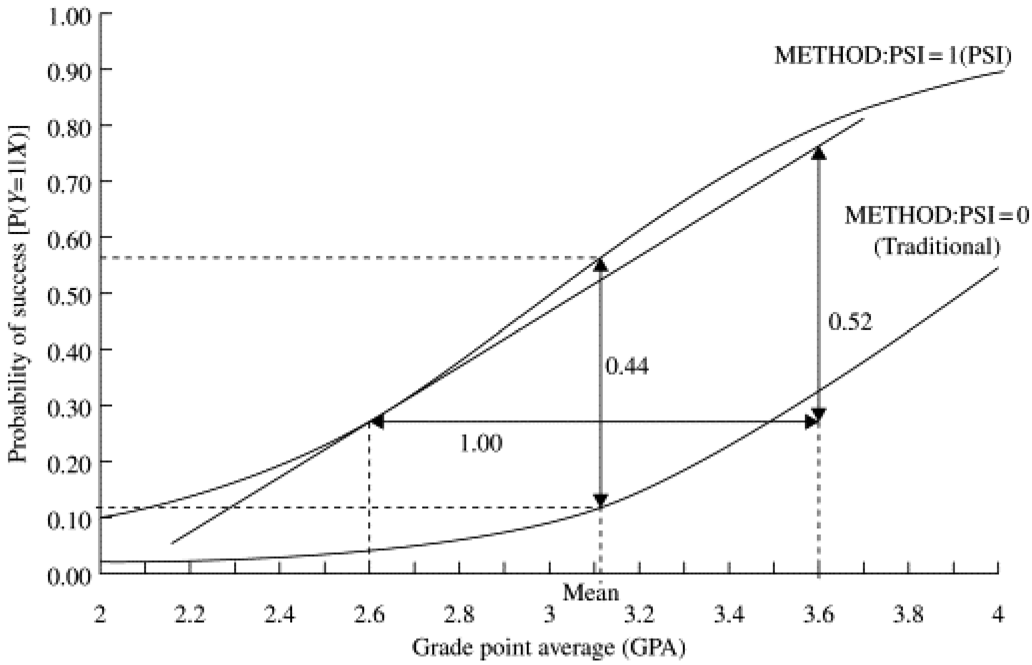


5.3. Predicted probability

$$P(y=1|x) = \sigma(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

Role:

- Main prediction function returning probability of class 1.
- With a 0.5 threshold:
  - If P > 0.5 → y = 1
  - If P ≤ 0.5 → y = 0



5.4. Logit (log-odds)

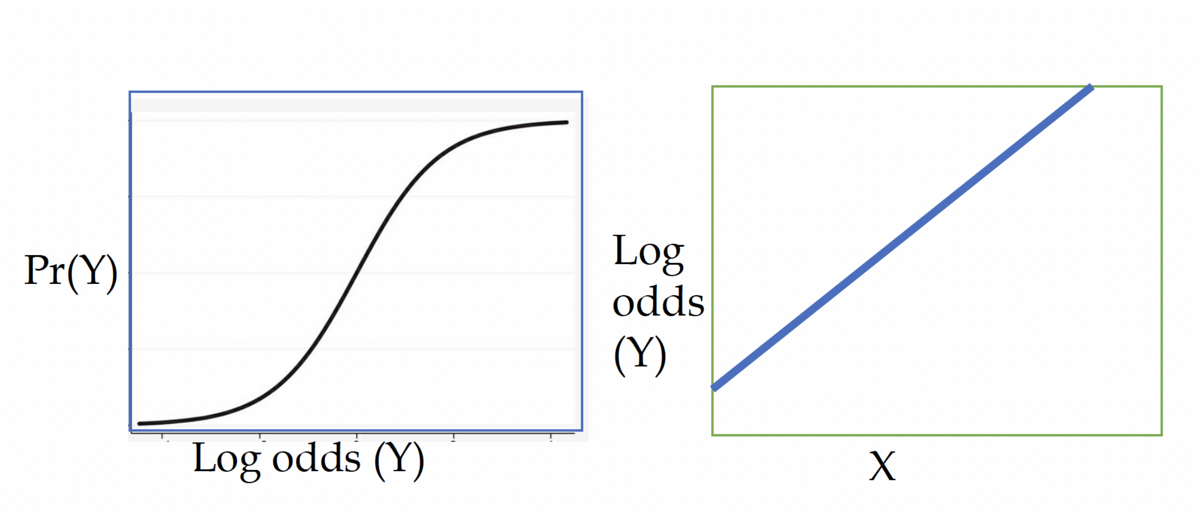
$$\text{logit}(P) = \ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

Role:

- Linear relationship between inputs and log-odds.
- Gives statistical meaning to each coefficient β_i.

Interpretation:

- Increasing x_i by 1 increases the log-odds by β_i.
- The odds are multiplied by e^{β_i} for a +1 change in x_i:
  - e^{β_i} > 1 → odds increase
  - e^{β_i} = 1 → no change
  - e^{β_i} < 1 → odds decrease
- Because σ(z) is strictly increasing, a higher z implies a higher P(y = 1 | x). The size of the probability change depends on the starting P.

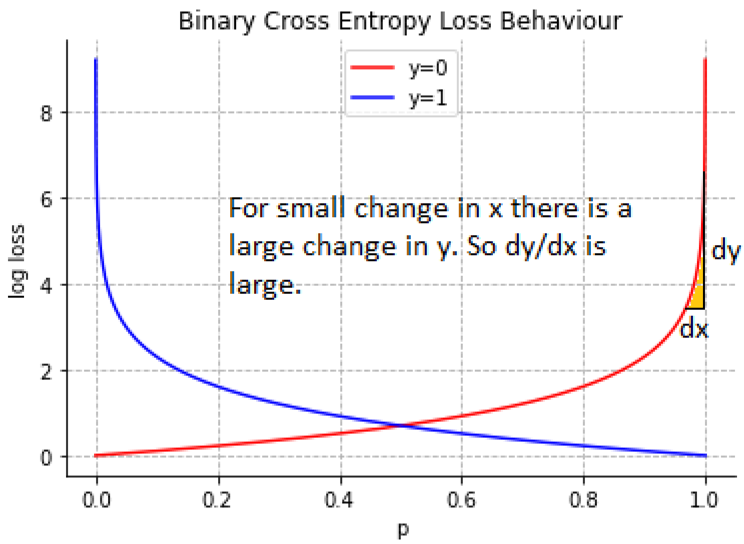


5.5. Loss function (Binary Cross Entropy)

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Role:

- Measures discrepancy between predicted probabilities and true labels.
- The model optimizes β by minimizing the loss with **Gradient Descent**.



5.6. Example

Task:

Predict **purchase (1)** vs **no purchase (0)** using:

- x_1: Website visits
- x_2: Income (million VND per month)

Model:

$\beta_0 = -4$, $\beta_1 = 0.6$, $\beta_2 = 0.04$

Customer:

- $x_1 = 5$, $x_2 = 60$

Step 1:

$z = -4 + 0.6 \cdot 5 + 0.04 \cdot 60 = 1.4$

Step 2:

$P(y=1|x) = 1/(1 + e^{-1.4}) \approx 0.802$

✅ **Conclusion:**

→ Purchase probability ≈ **80.2%** → Predict "likely to buy" (class = 1).

5.7. How the formulas work together

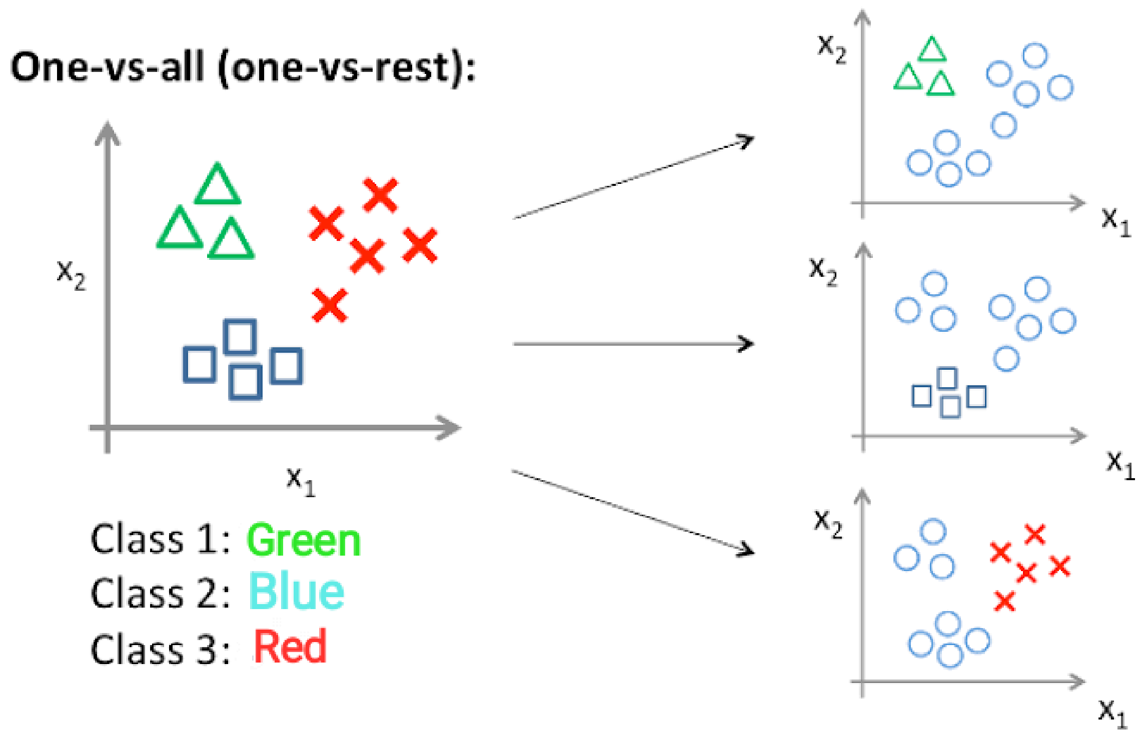
Formula	Role	Stage
$z = \beta_0 + \sum \beta_i x_i$	Linear combination of features	Initial step
$\sigma(z)$	Map linear score to probability	Nonlinearity
$P(y=1\|x)$	Final predicted probability	Output
$\ln(P/(1-P))$	Linear log-odds interpretation	Model analysis
$L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$	Error measure	Training

---

6. Multiclass Classification

Method 1: One-vs-All (OvA)

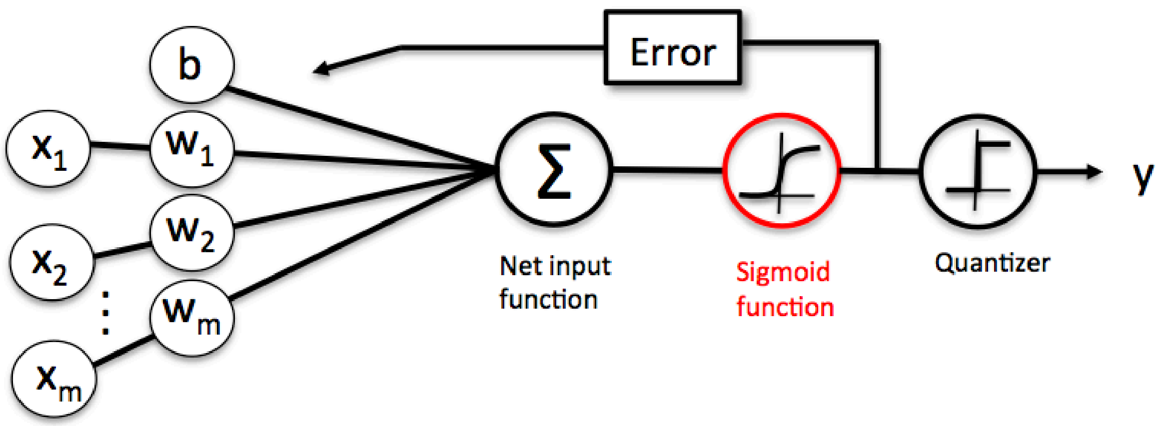
- Train **one logistic model per class**.
- Each model predicts probability of its class.
- Choose the class with highest probability.



Method 2: Softmax Regression

$$P(y=i|x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

→ Generalizes sigmoid to multiple classes.



---

7. Real-world Applications

Domain	Use case
Marketing	Predict purchase vs non-purchase
Finance	Default risk, bad credit prediction
E-commerce	Fraud detection
Healthcare	Disease presence prediction
Customer success	Churn prediction

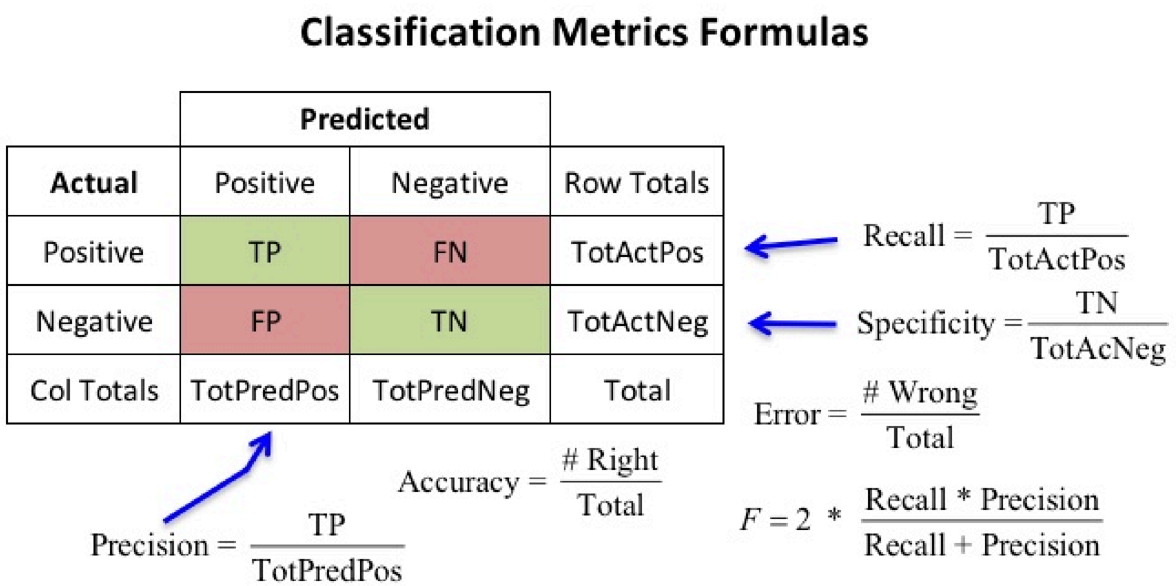
8. Sample Code – Python (Scikit-learn)

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pandas as pd

# 1. Load data
data = pd.read_csv('data.csv')
X = data[['feature1', 'feature2', 'feature3']]
y = data['label']

# 2. Chia tập train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Huấn luyện mô hình
model = LogisticRegression()
model.fit(X_train, y_train)

# 4. Dự đoán
y_pred = model.predict(X_test)

# 5. Đánh giá mô hình
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

9. Final Summary

Topic	Short summary
Goal	Predict probability of a binary outcome
Key functions	Sigmoid and Logit
Loss	Binary Cross-Entropy
Pros	Simple, interpretable, fast
Cons	Linear decision boundary, struggles with strong nonlinearity
Applications	Binary and multiclass classification
Output	Probability + class label

CLASSIFICATION ERROR METRICS – CHEATSHEET



1. LEARNING OBJECTIVES

Understand:

- **Error types** in classification
- How to **measure model performance**
- Use **error metrics** to select the best model

2. ERROR TYPES

Prediction	Reality	Outcome	Error type
Positive	Positive	**True Positive (TP)**	Correct positive prediction
Negative	Negative	**True Negative (TN)**	Correct negative prediction
Positive	Negative	**False Positive (FP)**	Type I Error
Negative	Positive	**False Negative (FN)**	Type II Error

---

3. CONFUSION MATRIX

$$\text{Confusion Matrix} = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

---

4. EVALUATION METRICS

Metric	Formula	Meaning	When to use
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall correctness	Balanced datasets
Recall (Sensitivity)	TP / (TP + FN)	Ability to catch positives	When missing positives is costly
Precision	TP / (TP + FP)	Quality of positive predictions	When false alarms are costly
Specificity (TNR)	TN / (TN + FP)	Ability to recognize negatives	When avoiding false alarms matters
FPR (False Positive Rate)	FP / (FP + TN)	False alarm rate	Used in ROC
F1-score	2 × (Precision × Recall) / (Precision + Recall)	Harmonic mean of precision and recall	When trading off precision and recall

---

5. REAL EXAMPLE – CANCER DETECTION

Assume:

- 1000 patients
- 10 sick (Positive), 990 healthy (Negative)

Model 1 – predict all "healthy"

Predicted	Actual	Count
Healthy	Healthy	990
Healthy	Sick	10

- TP=0, FP=0, FN=10, TN=990
- Accuracy = 99%, Recall = 0%, Precision = –

→ **Useless despite high accuracy**.

Model 2 – flag as sick when suspicious

	Sick	Healthy
Pred Sick	8	20
Pred Healthy	2	970

Calculations:

Metric	Value	Meaning
Accuracy	(8+970)/1000 = 97.8%	Overall correctness
Precision	8/(8+20)=0.285	28.5% of flagged positives are truly sick
Recall	8/(8+2)=0.8	80% of sick cases detected
Specificity	970/(970+20)=0.98	98% healthy correctly recognized
F1	2×(0.285×0.8)/(0.285+0.8)=0.42	Balanced performance

Analysis:

- **High recall**: good for disease detection.
- **Low precision**: many false alarms.

---

6. CHOOSING THE RIGHT METRIC

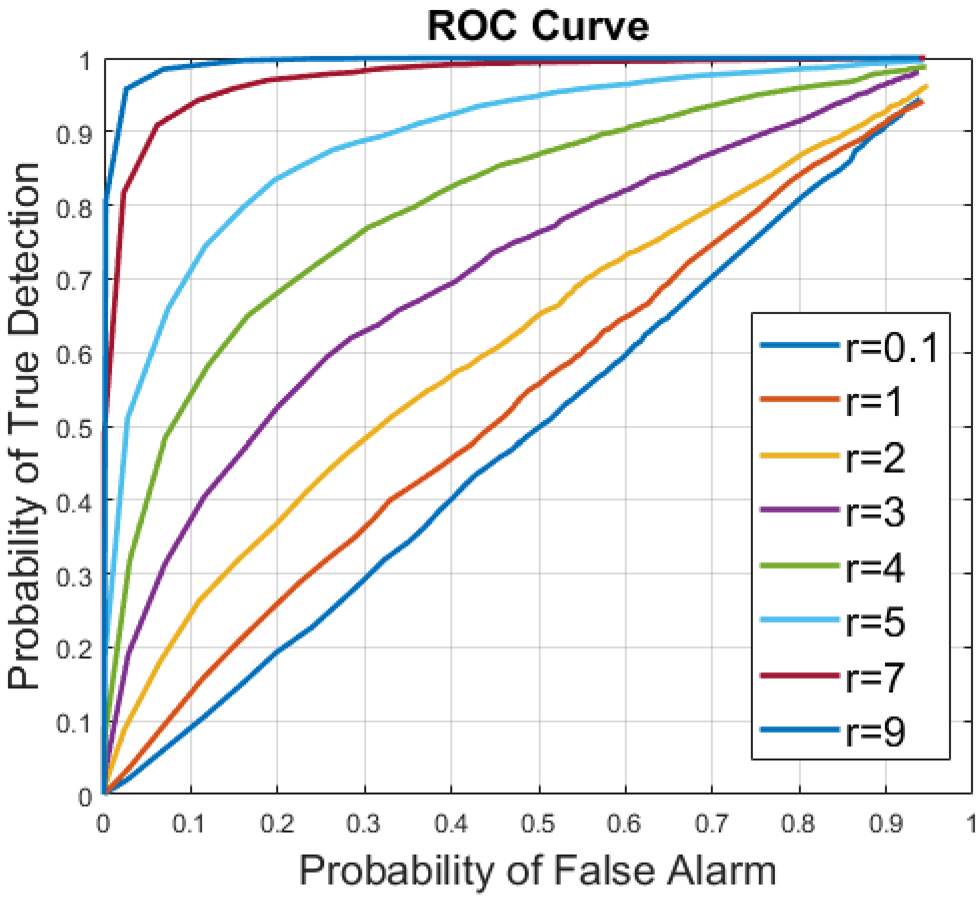
Objective	Priority
Cancer, fraud, terrorism detection	**Recall**
Spam filtering, annoying false alerts	**Precision**
Balanced data, low skew	**Accuracy**

---

7. ROC CURVE (Receiver Operating Characteristic)

- X-axis: **False Positive Rate** (1 – Specificity)
- Y-axis: **True Positive Rate** (Recall)
- Diagonal (AUC = 0.5): random guess
- **Higher AUC → better model**

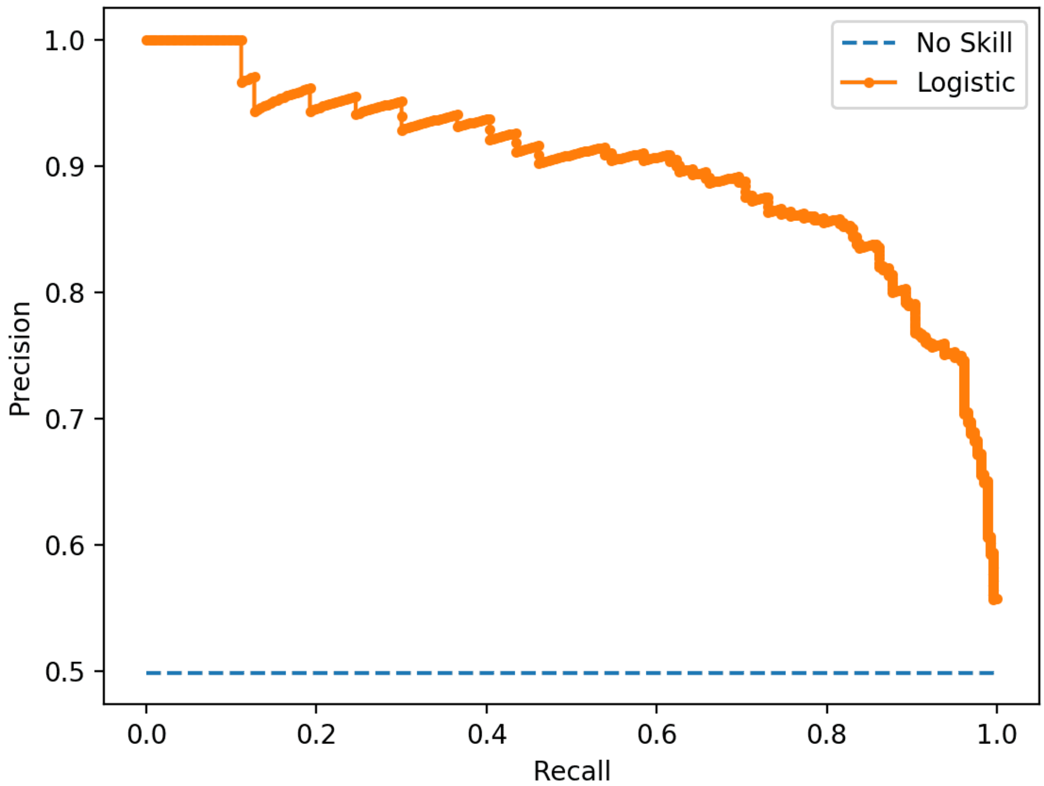
ROC is suitable for **balanced datasets**.



---

8. PRECISION–RECALL CURVE

- X-axis: **Recall**
- Y-axis: **Precision**
- Use when **data is imbalanced** (rare positives)
- Evaluates trade-off between catching positives and false alarms



9. ROC vs PR – Quick comparison

The Relationship Between Precision-Recall and ROC Curves

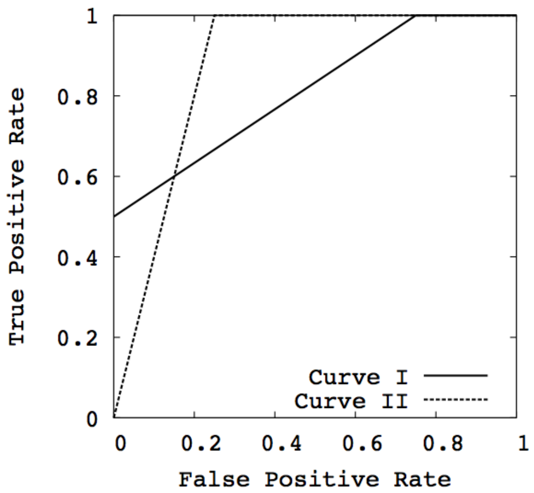


Figure 11. Comparing AUC-ROC for Two Algorithms

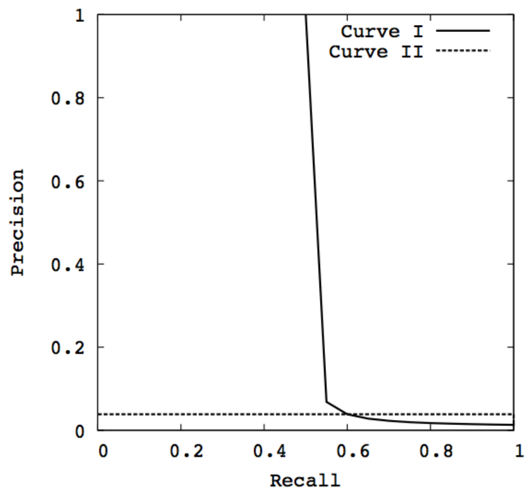


Figure 12. Comparing AUC-PR for Two Algorithms

Aspect	ROC Curve	Precision–Recall Curve
Best for	Balanced data	Imbalanced data
X-axis	False Positive Rate	Recall
Y-axis	True Positive Rate	Precision
Evaluates	Overall separability	Positive prediction quality
Area	AUC (Area Under Curve)	PR-AUC

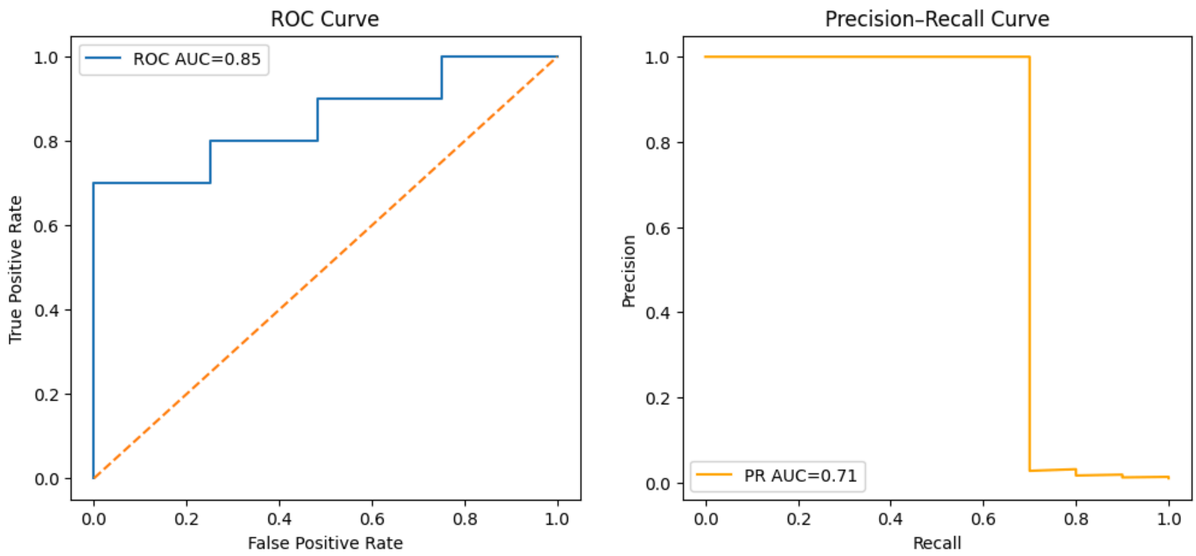
---

10. PYTHON CODE EXAMPLE

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, roc_curve, auc, precision_recall_curve
import matplotlib.pyplot as plt
import numpy as np

# Synthetic data
y_true = np.array([1]*10 + [0]*990)
y_score = np.array([0.9,0.8,0.7,0.85,0.6,0.5,0.4,0.3,0.2,0.1] + list(np.random.rand(990)*0.4))

# Confusion Matrix & Metrics
y_pred = (y_score > 0.5).astype(int)
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred))
print("Recall:", recall_score(y_true, y_pred))

# ROC Curve
fpr, tpr, _ = roc_curve(y_true, y_score)
roc_auc = auc(fpr, tpr)

# PR Curve
precision, recall, _ = precision_recall_curve(y_true, y_score)
pr_auc = auc(recall, precision)

plt.figure(figsize=(12,5))
```

```
plt.subplot(1,2,1)
plt.plot(fpr, tpr, label=f"ROC AUC={roc_auc:.2f}")
plt.plot([0,1],[0,1],'--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()

plt.subplot(1,2,2)
plt.plot(recall, precision, label=f"PR AUC={pr_auc:.2f}", color='orange')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision–Recall Curve")
plt.legend()
```

```
Confusion Matrix:
[[990   0]
 [  5   5]]
Accuracy: 0.995
Precision: 1.0
Recall: 0.5
```



11. MULTICLASS METRICS (Extensions)

Averaging methods:

1. **Macro-average:**
    - Compute metric per class then average.
    - Treat all classes equally.
2. **Micro-average:**
    - Aggregate TP, FP, FN across classes then compute.
    - Weighs frequent classes more.
3. **Weighted-average:**
    - Macro average weighted by class support.

Example (3 classes: A, B, C):

Macro F1 = (F1_A + F1_B + F1_C) / 3

Micro F1 = $\sum_i TP_i / (\sum_i TP_i + 0.5(FP_i + FN_i))$

---

12. TAKEAWAYS

Metric	Main role	When to use
Accuracy	Overall correctness	Balanced data
Recall	Catch positives	When missing positives is costly
Precision	Avoid false positives	When false alarms are costly
Specificity	Avoid false alarms	When true negatives matter
ROC-AUC	Overall separability	Model comparison
PR-AUC	Imbalanced data evaluation	Rare positives
F1-score	Balance precision and recall	When trading off both

---

13. REMEMBER

- **False Positive = Type I Error**
- **False Negative = Type II Error**
- **Do not rely on Accuracy alone**
- **Choose metrics based on objectives and error costs**
- **ROC suits balanced data, PR suits imbalanced data**

In short:

Classification error metrics help **objectively evaluate classification performance**, and prevent mistakes from relying only on Accuracy. Choosing the right metric (Precision, Recall, F1, ROC, PR-AUC, etc.) **depends on business objectives and data characteristics**.