Logistic regression

✅ Simplified Logistic Regression with NumPy

```python
import numpy as np

# Sigmoid activation
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Train logistic regression using gradient descent
def logistic_regression(X, y, lr=0.1, epochs=1000):
    w = np.zeros(X.shape[1])
    for _ in range(epochs):
        y_pred = sigmoid(X @ w)
        grad = X.T @ (y_pred - y) / len(y)
        w -= lr * grad
    return w

# Generate simple 2D data
np.random.seed(0)
X = np.random.randn(100, 2)
y = (X[:, 0] + X[:, 1] > 0).astype(int)

# Add bias term
X = np.c_[np.ones(len(X)), X]

# Train model
w = logistic_regression(X, y)

# Predict and evaluate
probs = sigmoid(X @ w)
preds = (probs >= 0.5).astype(int)
acc = np.mean(preds == y)

print("Accuracy:", acc)
```

🧠 Recap of the Core Formulas

Sigmoid:

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

Binary Cross-Entropy Loss:

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

Gradient:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{m} X^T (\hat{y} - y)$$