Name: Shivam Bhatt

Enrollment No: 92301733046

Code:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# 1. Generate Dummy Student Data

np.random.seed(0)

N = 60

data = pd.DataFrame({

'hours\_studied': np.random.randint(1, 10, N), # 1 to 9 hours/day

'attendance\_pct': np.random.randint(50, 101, N), # 50% to 100% attendance

'assignments\_done': np.random.randint(5, 21, N) # 5 to 20 assignments

})

# Generate exam score (continuous)

data['exam\_score'] = (

data['hours\_studied'] \* 8 +

data['attendance\_pct'] \* 2 +

data['assignments\_done'] \* 3 +

np.random.normal(0, 15, N) # Noise

)

# Discretize target: Low=0, Medium=1, High=2

bins = [0, 500, 800, np.inf]

labels = [0, 1, 2]

data['score\_class'] = pd.cut(data['exam\_score'], bins=bins, labels=labels).astype(int)

X = data[['hours\_studied', 'attendance\_pct', 'assignments\_done']]

y = data['score\_class']

n\_classes = len(y.unique())

# 2. Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y

)

# Convert to NumPy

X\_train\_np = X\_train.values

y\_train\_np = y\_train.values

X\_test\_np = X\_test.values

y\_test\_np = y\_test.values

# 3. Logistic Regression (One-vs-Rest)

def sigmoid(z):

return 1 / (1 + np.exp(-z))

def train\_one\_vs\_rest(X, y, n\_classes, lr=1e-4, n\_iter=5000):

n\_samples, n\_features = X.shape

weights = []

for k in range(n\_classes):

yk = (y == k).astype(np.float64)

w = np.zeros(n\_features)

b = 0.0

for \_ in range(n\_iter):

logits = np.dot(X, w) + b

y\_pred = sigmoid(logits)

dw = np.dot(X.T, (y\_pred - yk)) / n\_samples

db = np.sum(y\_pred - yk) / n\_samples

w -= lr \* dw

b -= lr \* db

weights.append({'w': w, 'b': b})

return weights

def predict(X, weights):

n\_samples = X.shape[0]

n\_classes = len(weights)

probs = np.zeros((n\_samples, n\_classes))

for k in range(n\_classes):

w = weights[k]['w']

b = weights[k]['b']

logits = np.dot(X, w) + b

probs[:, k] = sigmoid(logits)

return np.argmax(probs, axis=1)

# 4. Train & Evaluate

weights = train\_one\_vs\_rest(X\_train\_np, y\_train\_np, n\_classes)

y\_pred = predict(X\_test\_np, weights)

print("Accuracy:", accuracy\_score(y\_test\_np, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test\_np, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test\_np, y\_pred))



# 5. Sample Predictions

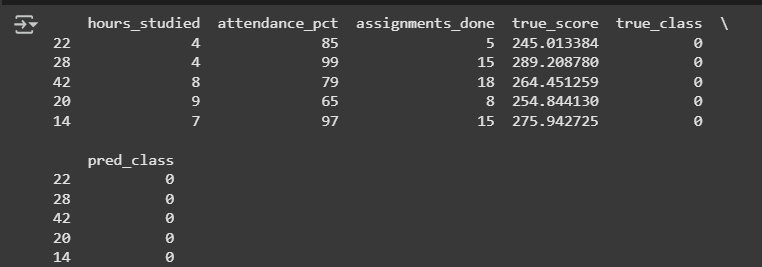
results = X\_test.copy()

results['true\_score'] = data.loc[X\_test.index, 'exam\_score']

results['true\_class'] = y\_test

results['pred\_class'] = y\_pred

print(results.head())



# 6. Class Distribution

print("\nClass distribution:")

print(data['score\_class'].value\_counts())

