LAB - 3 (24/3/25)

Simple Linear Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def estimate_coef(x,y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    SS_xy = np.sum((x-m_x)*(y-m_y))
    SS_xx = np.sum((x-m_x)**2)
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

def plot(x,y,b):
    plt.scatter(x,y, color='m')
    y_pred = b[0] + b[1]*x
    plt.plot(x, y_pred, color='g')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()

file_path = input("Enter path:")
df = pd.read_csv(file_path)

x = df.iloc[:,0].values
y = df.iloc[:,1].values

b = estimate(x,y)
plot(x,y,b)
```

Output:

Enter path: /content/tvmarketing.csv

Estimated coefficients:

b_0 = 7.032

b_1 = 0.047

Multiple Linear Regression

```
data = {
"feature1": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
"feature2": [2, 3, 5, 7, 11, 13, 17, 19, 23, 29],
"features": [5, 6, 9, 12, 15, 18, 21, 24, 27, 30],
"Target": [5, 9, 15, 22, 51, 41, 53, 66, 80, 96]
}
df = pd.DataFrame(data)
X = df.drop(columns=["Target"]).values
y = df["Target"].values.reshape(-1, 1)
X = np.hstack((np.ones((X.shape[0], 1)), X))
beta = np.linalg.solve(X.T @ X + 0.01 * np.identity(X.shape[1]),
y_pred = X @ beta
mse = np.mean((y - y_pred) ** 2)
tot_var = np.sum((y - np.mean(y)) ** 2)
exp_var = np.sum((y_pred - np.mean(y)) ** 2)
r2 = exp_var / tot_var

print("output:")
```

output:

Model Coefficients: [0.040, 3.313, 0.120]

Intercept: -3.1599

Mean Squared Error: 5.362

R-Squared Score: 0.993

Logistic Regression

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def compute(X, y, theta):
    m = len(y)
    h = sigmoid(X @ theta)
    cost = (-1/m) * np.sum(y * np.log(h) + (1-y) * np.log(1-h))
    return cost

def grad_descent(X, y, theta, alpha, iterations):
    m = len(y)
    cost_history = []
    for _ in range(iterations):
        gradient = (1/m) * X.T @ (sigmoid(X @ theta) - y)
        theta -= alpha * gradient
    return theta, cost_history

def predict(X, theta):
    return (sigmoid(X @ theta) >= 0.5).astype(int)

accuracy = np.mean(y_pred == y)
print("Accuracy:", accuracy * 100)
```

Output:

Accuracy: 1