**LAB ASSIGNMENT 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Semester:** | VII | **Division:** | 6 |
| **Roll No.:** | 21BCP359 | **Date:** | 05-08-24 | **Batch:** | G11 |
| **Aim:** | Implement simple and multi-linear regression to predict profits for a food truck. Compare the performance of the model on linear and multi-linear regression. | | | | |

**Objective**

The objective of this lab assignment is to implement simple and multi-linear regression models to predict profits for a food truck business. By comparing the performance of these two regression models, you will gain insights into when and how to use simple and multi-linear regression techniques.

**Steps**

1. **Data Preparation**:

* Construct a numpy array containing both features (Population, Years in Business) and the target variable (Profit).
* Separate the data into feature set XXX and target variable yyy.
* Address any missing values in yyy by substituting NaN with the average of yyy.

1. **Simple Linear Regression**:

* Develop a function called simple\_linear\_regression to compute the coefficients (β0\beta\_0β0​, β1\beta\_1β1​), make predictions, and calculate the mean squared error (MSE) for a single feature (Years in Business).
* Extract the Years in Business as X\_simple and apply the simple linear regression model.

1. **Multiple Linear Regression**:

* Create a function named multi\_linear\_regression to determine coefficients, predictions, and MSE using all features.
* Prepare the feature set X\_multi by adding a column of ones for the intercept term and then perform the multiple linear regression analysis.

1. **Plotting**:

* Generate two subplots:
  + The first subplot will display the results of the simple linear regression (Years in Business vs. Profit).
  + The second subplot will illustrate the results of the multiple linear regression (Population vs. Profit).
* Show both actual and predicted values in these plots.

1. **Print Results**:

* Print out the coefficients, intercepts, and MSE for both the simple and multiple linear regression models.

**Code**

import numpy as np

import matplotlib.pyplot as plt

data = np.array(

    [

        [10000, 5, 10000],

        [15000, 6, 12000],

        [20000, 6, 13000],

        [9000, 5, 12000],

        [12000, 4, np.nan],

    ]

)

X = data[:, :-1]

y = data[:, -1]

mean\_y = np.nanmean(y)

y[np.isnan(y)] = mean\_y

*def* simple\_linear\_regression(*X\_simple*, *y*):

    X\_simple\_mean = np.mean(X\_simple)

    y\_mean = np.mean(y)

    beta1 = np.sum((X\_simple - X\_simple\_mean) \* (y - y\_mean)) / np.sum(

        (X\_simple - X\_simple\_mean) \*\* 2

    )

    beta0 = y\_mean - beta1 \* X\_simple\_mean

    y\_pred = beta0 + beta1 \* X\_simple

    mse = np.mean((y - y\_pred) \*\* 2)

    return beta0, beta1, y\_pred, mse

*def* multi\_linear\_regression(*X\_multi*, *y*):

    XTX = np.dot(X\_multi.T, X\_multi)

    XTX\_inv = np.linalg.inv(XTX)

    XTy = np.dot(X\_multi.T, y)

    beta = np.dot(XTX\_inv, XTy)

    y\_pred = np.dot(X\_multi, beta)

    mse = np.mean((y - y\_pred) \*\* 2)

    return beta, y\_pred, mse

X\_simple = X[:, 1]

beta0, beta1, y\_pred\_simple, mse\_simple = simple\_linear\_regression(X\_simple, y)

X\_multi = np.hstack([np.ones((X.shape[0], 1)), X])

beta\_multi, y\_pred\_multi, mse\_multi = multi\_linear\_regression(X\_multi, y)

plt.figure(*figsize*=(14, 6))

plt.subplot(1, 2, 1)

plt.scatter(X\_simple, y, *color*="blue", *label*="Actual Profit")

plt.plot(

    X\_simple, y\_pred\_simple, *color*="red", *linewidth*=2, *label*="Simple Linear Regression"

)

plt.xlabel("Years in Business")

plt.ylabel("Profit")

plt.title(*f*"Simple Linear Regression\nMSE = {mse\_simple*:.2f*}")

plt.legend()

plt.subplot(1, 2, 2)

X\_population = X[:, 0]

plt.scatter(X\_population, y, *color*="blue", *label*="Actual Profit")

plt.scatter(

    X\_population,

    y\_pred\_multi,

*color*="green",

*marker*="x",

*label*="Multi-Linear Regression Predictions",

)

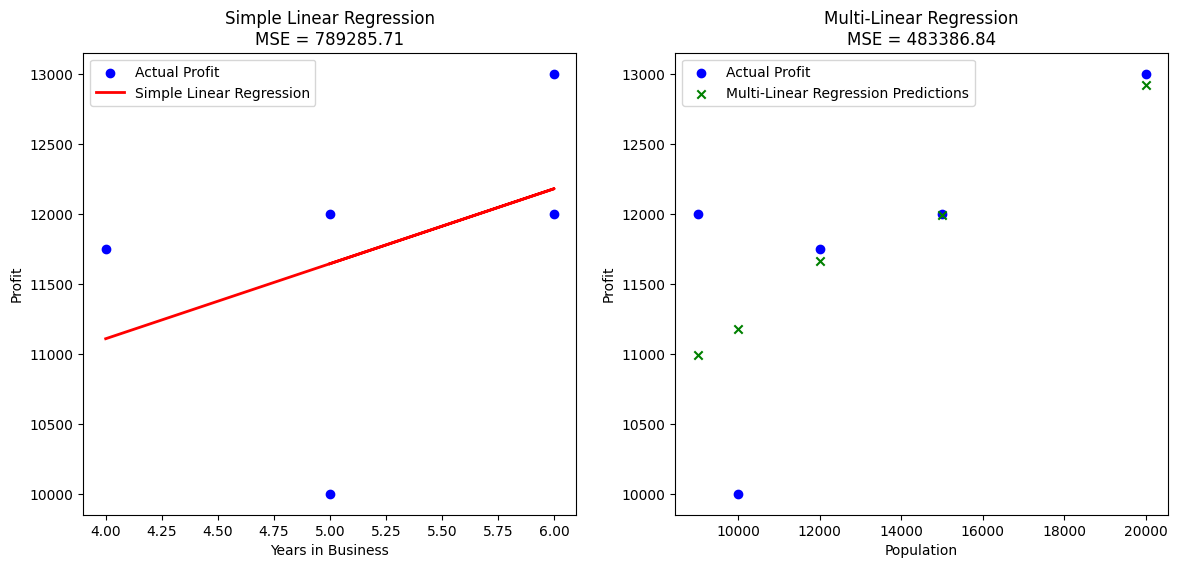
plt.xlabel("Population")

plt.ylabel("Profit")

plt.title(*f*"Multi-Linear Regression\nMSE = {mse\_multi*:.2f*}")

plt.legend()

plt.show()



X = np.array([[10000], [15000], [20000], [9000]])

y = np.array([10000, 12000, 13000, 12000])

Population\_mean = X.mean()

Population\_std = X.std()

Population\_normalized = (X - Population\_mean) / Population\_std

X = np.hstack((np.ones((X.shape[0], 1)), Population\_normalized))

y = y

w = np.zeros(X.shape[1])

learning\_rate = 0.01

n\_iterations = 1000

*def* predict(*X*, *w*):

    return X.dot(w)

*def* compute\_cost(*X*, *y*, *w*):

    m = len(y)

    predictions = predict(X, w)

    cost = (1 / (2 \* m)) \* np.sum((predictions - y) \*\* 2)

    return cost

*def* gradient\_descent(*X*, *y*, *w*, *learning\_rate*, *n\_iterations*):

    m = len(y)

    cost\_history = []

    for i in range(n\_iterations):

        predictions = predict(X, w)

        errors = predictions - y

        gradients = (1 / m) \* X.T.dot(errors)

        w -= learning\_rate \* gradients

        cost = compute\_cost(X, y, w)

        cost\_history.append(cost)

    return w, cost\_history

w, cost\_history = gradient\_descent(X, y, w, learning\_rate, n\_iterations)

print(*f*"Final weights: {w}")

plt.figure(*figsize*=(12, 6))

plt.plot(range(len(cost\_history)), cost\_history)

plt.xlabel("Iterations")

plt.ylabel("Cost (Loss)")

plt.title("Cost Function over Iterations")

plt.grid(True)

plt.show()

**Final weights: [11749.49273784 769.20068232]**

