**Lab 4: Regression**

1. **Write Python code to implement this Simple linear regression. Given a dataset with height and weight, find the regression line that best fits to predict the weight of a person having a height of 172 cm. [Simple Linear Regression]**

|  |  |
| --- | --- |
| **X (Height)** | **Y (Weight)** |
| 150 | 50 |
| 160 | 56 |
| 170 | 62 |
| 180 | 68 |

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Sample data (Height in cm, Weight in kg)

# You can replace this with your actual dataset

height = np.array([150, 160, 170, 180]).reshape(-1, 1)

weight = np.array([50, 56, 62, 68])  # Corresponding weights

# Create and train the model

model = LinearRegression()

model.fit(height, weight)

# Coefficients

slope = model.coef\_[0]

intercept = model.intercept\_

print(f"Regression Line: weight = {slope:.2f} \* height + {intercept:.2f}")

# Predict weight for height = 172 cm

height\_to\_predict = np.array([[172]])

predicted\_weight = model.predict(height\_to\_predict)

print(f"Predicted weight for height 172 cm: {predicted\_weight[0]:.2f} kg")

# Plotting

plt.scatter(height, weight, color='blue', label='Data points')

plt.plot(height, model.predict(height), color='red', label='Regression line')

plt.scatter(172, predicted\_weight, color='green', label='Prediction (172 cm)')

plt.xlabel("Height (cm)")

plt.ylabel("Weight (kg)")

plt.title("Simple Linear Regression")

plt.legend()

plt.grid(True)

plt.show()

1. **Write python code to: Given a dataset with study hours and sleep hours for 5 students. Use linear regression to predict the exam score for a new student that studies 6 hours and sleeps 4 hours. [Multiple Linear Regression]**

| **Student** | **Hours Studied (x1)** | **Hours Slept (x2)** | **Exam Score (y)** |
| --- | --- | --- | --- |
| 1 | 2 | 9 | 60 |
| 2 | 1 | 6 | 50 |
| 3 | 3 | 7 | 65 |
| 4 | 4 | 5 | 70 |
| 5 | 5 | 5 | 75 |

import numpy as np

from sklearn.linear\_model import LinearRegression

# Dataset: [Study Hours, Sleep Hours] -> Exam Score

# Data for 5 students

X = np.array([

    [2, 9],   # Student 1

    [1, 6],   # Student 2

    [3, 7],   # Student 3

    [4, 5],   # Student 4

    [5, 5]   # Student 5

])

y = np.array([60, 50, 65, 70, 75])  # Exam scores

# Create and train the model

model = LinearRegression()

model.fit(X, y)

# Predict exam score for a new student who studies 6 hours and sleeps 4 hours

new\_data = np.array([[6, 4]])

predicted\_score = model.predict(new\_data)

# Output

print(f"Predicted exam score: {predicted\_score[0]:.2f}")

**Your Work: Visualize the plane.**

1. **Given the following data points:**

| **Years of Experience (X)** | **Salary ($) (Y)** |
| --- | --- |
| 1 | 50,000 |
| 2 | 55,000 |
| 3 | 65,000 |
| 4 | 80,000 |
| 5 | 110,000 |
| 6 | 150,000 |
| 7 | 200,000 |

**Fit a polynomial regression model of degree 2 to the data and find the equation of the polynomial. Hence, predict the salary for 10 years of experience. [Polynomial Regression]**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

# Data

X = np.array([1, 2, 3, 4, 5, 6, 7]).reshape(-1, 1)

y = np.array([50000, 55000, 65000, 80000, 110000, 150000, 200000])

# Polynomial features of degree 2

poly = PolynomialFeatures(degree=2)

X\_poly = poly.fit\_transform(X)

# Fit the model

model = LinearRegression()

model.fit(X\_poly, y)

# Coefficients

a = model.coef\_[2]

b = model.coef\_[1]

c = model.intercept\_

print(f"Fitted Polynomial: y = {a:.2f}x² + {b:.2f}x + {c:.2f}")

# Predict salary for 10 years of experience

X\_new = np.array([[10]])

X\_new\_poly = poly.transform(X\_new)

predicted\_salary = model.predict(X\_new\_poly)

print(f"Predicted salary for 10 years of experience: ${predicted\_salary[0]:,.2f}")

# Plotting

x\_range = np.linspace(1, 10, 100).reshape(-1, 1)

x\_range\_poly = poly.transform(x\_range)

y\_range = model.predict(x\_range\_poly)

plt.scatter(X, y, color='red', label='Actual Data')

plt.plot(x\_range, y\_range, color='blue', label='Polynomial Regression (Degree 2)')

plt.scatter(10, predicted\_salary, color='green', s=100, label='Predicted (10 yrs)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary ($)')

plt.title('Polynomial Regression: Salary vs Experience')

plt.legend()

plt.grid(True)

plt.show()

1. **Compute the MAE, MSE, RMSE and R2 a model that performed below prediction as in the report.**

|  |  |  |
| --- | --- | --- |
| **Observation** | **Actual Value (y)** | **Predicted Value (ŷ)** |
| **1** | **3** | **2.5** |
| **2** | **5** | **5.1** |
| **3** | **7** | **6.8** |
| **4** | **9** | **9.3** |

import numpy as np

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

# Actual and predicted values

y\_true = np.array([3, 5, 7, 9])

y\_pred = np.array([2.5, 5.1, 6.8, 9.3])

# Metrics

mae = mean\_absolute\_error(y\_true, y\_pred)

mse = mean\_squared\_error(y\_true, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_true, y\_pred)

# Output

print(f"MAE:  {mae:.3f}")

print(f"MSE:  {mse:.3f}")

print(f"RMSE: {rmse:.3f}")

print(f"R²:   {r2:.3f}")