**Name:- Medhansh K**

**PRN:-122A8035**

**Batch:-B2**

**EXPERIMENT 2**

**Aim:-** Implementation of simple and multiple linear Regression.

**Theory:-**

Linear regression is also a type of [supervised machine-learning algorithm](https://www.geeksforgeeks.org/supervised-machine-learning/) that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation with observed data. It predicts the continuous output variables based on the independent input variable.

For example if we want to predict house price we consider various factor such as house age, distance from the main road, location, area and number of room, linear regression uses all these parameter to predict house price as it consider a linear relation between all these features and price of house.

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.

The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).



Multiple linear regression (MLR) is a statistical technique that uses multiple independent variables to predict a dependent variable. It's used to model quantitative outcomes.

How it works

* MLR assumes that the dependent variable is a linear combination of the independent variables
* It fits a regression surface in a space with as many dimensions as the number of independent variables plus one
* It can include categorical independent variables
* It can also include functions of the independent variables, such as polynomial terms and cross-products

When to use

* MLR can be used to study the relationship between many factors and a single outcome, such as the occurrence of a disease
* It can also be used to understand how different factors contribute to a variation in an outcome, such as exam performance

Assumptions

* The effects of each variable are the same regardless of the values of other variables
* The errors are independent and identically distributed



**Code:**

Simple linear regression:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from google.colab import files

uploaded = files.upload()

file\_name = "experience\_salary.xlsx"

df = pd.read\_excel(file\_name)

X = df[['Years of Experience']].values

y = df['Salary'].values

# Splitting the dataset into training (80%) and testing (20%) sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Creating and training the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Making predictions

y\_pred = model.predict(X\_test)

# Visualizing the results

plt.scatter(X\_train, y\_train, color="blue", label="Training Data")

plt.scatter(X\_test, y\_test, color="red", label="Test Data")

plt.plot(X\_test, y\_pred, color="black", linewidth=2, label="Prediction Line")

plt.xlabel("Years of Experience")

plt.ylabel("Salary")

plt.title("Simple Linear Regression: Experience vs Salary")

plt.legend()

plt.show()

# Print model parameters

print(f"Intercept: {model.intercept\_}")

print(f"Coefficient: {model.coef\_[0]}")



Multiple Linear Regression:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

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

# Load dataset

df = pd.read\_excel("multiple\_linear\_regression.xlsx")

# Features (Independent variables) and Target (Dependent variable)

X = df[['Years of Experience', 'Age', 'Education Level']]

y = df['Salary']

# Splitting the dataset into training (80%) and testing (20%) sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Creating and training the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Making predictions

y\_pred = model.predict(X\_test)

# Model evaluation

print(f"Mean Absolute Error: {mean\_absolute\_error(y\_test, y\_pred):.2f}")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test, y\_pred):.2f}")

print(f"R² Score: {r2\_score(y\_test, y\_pred):.2f}")

# Print model coefficients

print(f"Intercept: {model.intercept\_}")

print(f"Coefficients: {model.coef\_}")

# Predict Salary for a specific example (e.g., 10 years experience, 35 years old, education level 3)

sample\_input = np.array([[10, 35, 3]])

predicted\_salary = model.predict(sample\_input)

print(f"Predicted Salary for (10 years experience, 35 years old, Education Level 3): {predicted\_salary[0]:.2f}")

# Plotting Actual vs Predicted Salaries

plt.figure(figsize=(10, 6))

# Scatter plot of actual vs predicted values

plt.scatter(y\_test, y\_pred, color='blue', label='Predicted vs Actual')

# Plot the line y = x (ideal predictions)

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--', label='Ideal Prediction')

# Adding labels and title

plt.xlabel('Actual Salary')

plt.ylabel('Predicted Salary')

plt.title('Actual vs Predicted Salary')

# Show the legend

plt.legend()

# Display the plot

plt.show()





**Conclusion:-** The multiple linear regression model successfully predicts salary based on factors such as years of experience, age, and education level. The evaluation metrics indicate a reasonable fit, with the model demonstrating a strong ability to estimate salaries within the given dataset.