

# Lab-4: Linear Regression

## 1. Introduction

Linear Regression is a supervised machine learning algorithm used to model the relationship between dependent and independent variables.

There are two types:

- **Simple Linear Regression** → One independent variable
- **Multiple Linear Regression** → More than one independent variable

Linear Regression helps to:

- Predict continuous values
- Understand relationships between variables
- Perform forecasting and trend analysis

Mathematical Form:

**Simple Linear Regression:**

$$Y = \beta_0 + \beta_1 X$$

**Multiple Linear Regression:**

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

## 2. Experiments and Observations

---

### 2.1 Simple Linear Regression

**Procedure:**

- Created a dataset (Experience vs Salary).
- Split the dataset into training and testing sets.
- Trained a Linear Regression model using sklearn.
- Predicted salary values.
- Visualized regression line.

## **Code :**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
# Sample dataset
```

```
data = {'Experience': [1,2,3,4,5,6,7,8,9,10],
        'Salary': [30000,35000,40000,45000,50000,
                    55000,60000,65000,70000,75000]}
```

```
df = pd.DataFrame(data)
```

```
X = df[['Experience']]
```

```
y = df['Salary']
```

```
# Splitting dataset
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

```
# Model training
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
# Prediction
```

```
y_pred = model.predict(X_test)
```

```
# Visualization
```

```
plt.scatter(X, y)
```

```
plt.plot(X, model.predict(X), color='red')
```

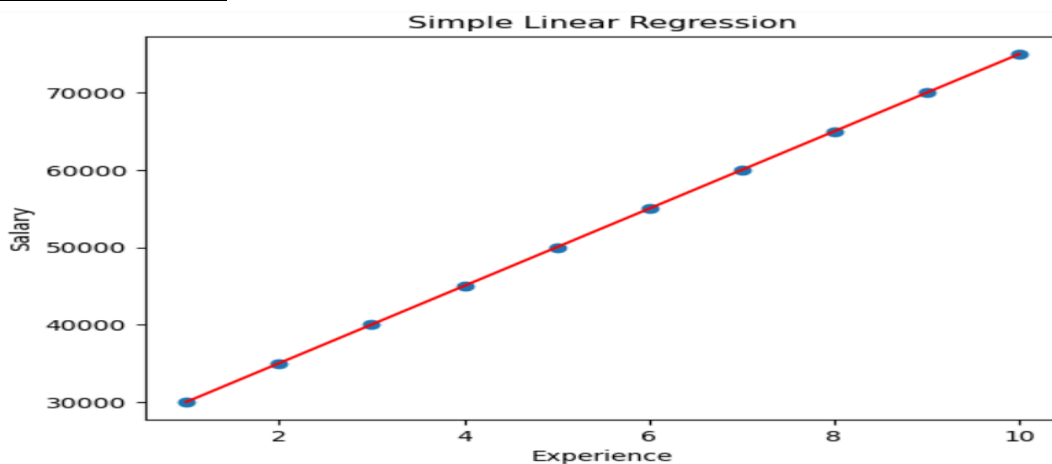
```
plt.xlabel("Experience")
```

```
plt.ylabel("Salary")
```

```
plt.title("Simple Linear Regression")
```

```
plt.show()
```

## **Observation :**



## 2.2 Multiple Linear Regression

### Procedure:

- Loaded California Housing dataset.
- Used multiple features to predict house prices.
- Split dataset into training and testing sets.
- Trained Linear Regression model.
- Predicted house prices.

### Code:

```
from sklearn.datasets import fetch_california_housing
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

data = fetch_california_housing()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

### Observation :

- Multiple features influence house price.
- Coefficients represent feature importance.
- Model accuracy depends on data quality.
- Prediction error varies depending on feature distribution.

## 2.3 Evaluating Model Performance

### Procedure:

Calculated performance metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- $R^2$  Score

### **Code:**

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

```
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r2)
```

### **Observation :**

```
MAE: 0.5332001304957
MSE: 0.5558915986952425
RMSE: 0.7455813830127751
R2 Score: 0.5757877060324521
```

## **2.4 Visualizing Residuals**

### **Procedure:**

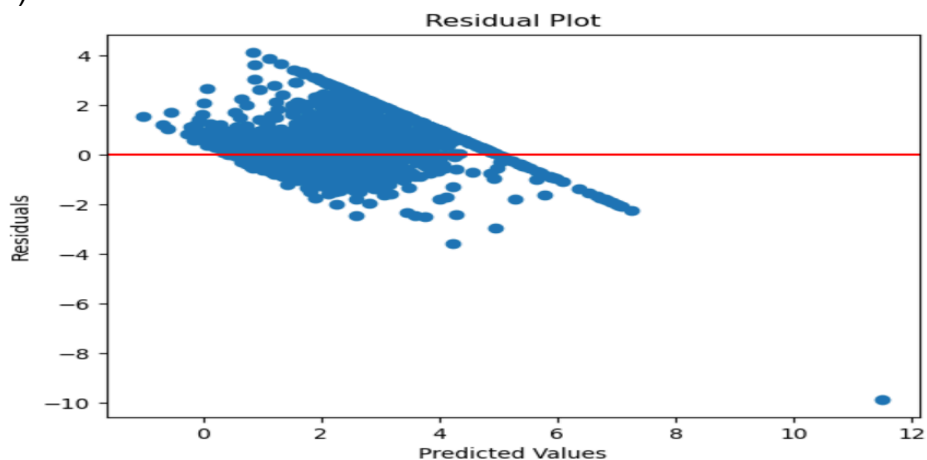
- Calculated residuals (Actual – Predicted).
- Plotted residual graph.
- Checked for patterns.

### **Code:**

```
residuals = y_test - y_pred
```

```
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='red')
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```

### **Observation :**



## 2.5 Predicting New Data

### Procedure:

- Used trained model.
- Provided new input values.
- Generated prediction.

### Code:

```
import numpy as np

# New house data (must contain 8 features)
# Order: [MedInc, HouseAge, AveRooms, AveBedrms,
#         Population, AveOccup, Latitude, Longitude]

new_house = np.array([[8.5, 35, 6, 1, 300, 3, 37.5, -122]])

# Predict house price
predicted_price = model.predict(new_house)

print("Predicted House Price:", predicted_price)
```

### Observation :

---

**Predicted House Price: [4.33299073]**

## 3. Conclusion

- Simple Linear Regression models relationship between two variables.
- Multiple Linear Regression handles multiple predictors.
- Performance metrics evaluate model accuracy.
- Residual analysis validates model assumptions.
- Linear regression is widely used for prediction and forecasting tasks.