Lab 3: Linear Regression with a Single Variable

Objective:

To understand and implement Linear Regression with a single independent variable. The lab focuses on the core concepts, assumptions, and practical implementation using Python. Students will learn to visualize the relationship, interpret results, and evaluate model performance.

Prerequisites:

- Knowledge of basic Python programming.
- Familiarity with libraries: Pandas, NumPy, Matplotlib, and Scikit-learn.
- Understanding of basic statistical concepts like correlation and variance.

1. Introduction to Regression Analysis

What is Regression?
Regression is a statistical method for understanding the relationship between a dependent variable (target) and one or more independent variables (predictors).

In **Simple Linear Regression**, the relationship is modeled as a straight line:

$$y=\beta 0+\beta 1x+\epsilon$$

- y: Dependent variable (e.g., Price).
- x: Independent variable (e.g., Avg. Area House Age).
- $\beta 0$: Intercept (value of y when x is 0).
- $\beta 1$: Slope (rate of change in y with a unit increase in x).
- ϵ: Error term (difference between predicted and actual values).

2. Assumptions of Linear Regression

1. **Linearity:** The relationship between xxx and yyy is linear.
2. **Independence:** Observations are independent of each other.
3. **Homoscedasticity:** Residuals (errors) have constant variance.
4. **Normality of Residuals:** Residuals follow a normal distribution.

3. Theory: Evaluating Linear Regression Models

Key Metrics:

1. **Mean Squared Error (MSE):** Measures average squared error between predicted and actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

2. **R-squared (R^2):** Explains the proportion of variance in y explained by x

$$R^2 = 1 - \frac{\text{Sum of Squared Errors}}{\text{Total Sum of Squares}}$$

3. **Visualization:**

- Regression line: Shows how well the model fits the data.
- Residual plot: Helps assess assumptions of linearity and homoscedasticity.

4. Implementation of Simple Linear Regression

Step 1: Import Libraries

```
import import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model  import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

- **Pandas:** For loading and manipulating tabular data.
- **NumPy:** For mathematical operations (used for arrays and calculations).
- **Matplotlib:** For creating visualizations like scatter plots and regression lines.
- **Scikit-learn**: For building and evaluating the regression model.

Step 2: Load Dataset

```
data = pd.read_csv("housing.csv")
print("First 5 rows of the dataset:")
print(data.head())
```

- pd.read_csv: Reads a CSV file into a DataFrame.
- data.head(): Displays the first five rows to check data loading and structure.

Step 3: Data Preprocessing

1. Select the independent (x) and dependent (y) variables.
2. Handle missing values (if any).

```
# Selecting independent (X) and dependent (y) variables
X = data[['Avg. Area House Age']] # Independent variable
y = data['Price'] # Dependent variable

# Check for missing values print("\nMissing values in the
dataset:")
print(data.isnull().sum())
```

- **X** = data[['SquareFeet']]: Extracts the independent variable column (predictor).
- **y** = data['Price']: Extracts the dependent variable column (target).
- **isnull().sum():** Identifies missing values in each column.

Step 4: Train-Test Split

Divide the data into training and testing sets.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

train_test_split: Divides the data into training and testing sets.

- test_size=0.2: Reserves 20% of the data for testing.
- random_state=42: Ensures reproducibility of the split.

Step 5: Train the Model

```
# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Print model parameters
print("Slope (β1):", model.coef_)
print("Intercept (β0):", model.intercept_)
```

- LinearRegression(): Creates a linear regression model.
- model.fit: Fits the model to the training data.
- model.coef_: Extracts the slope (β1).
- model.intercept_: Extracts the intercept (β0).

# Step 6: Make Predictions

```
 y_pred = model.predict(X_test)

# Compare actual and predicted values
comparison = pd.DataFrame({'Actual': y_test.values, 'Predicted': y_pred})
print(comparison.head())
```

- **model.predict:** Generates predictions for the test set.
- **DataFrame:** Combines actual and predicted values for easy comparison.

Step 7: Evaluate the Model

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation Metrics:")
 print("Mean Squared Error (MSE):", mse)
print("R-squared (R²):", r2)
```

Step 8: Visualize the Results

```
# Scatter plot with regression line
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Regression Line')
plt.xlabel("Avg. Area House Age")
plt.ylabel("Price")
plt.title("Linear Regression: Avg. Area House Age vs Price")
plt.legend() plt.show()
```

- **plt.scatter:** Creates a scatter plot of the actual data points.
- **plt.plot:** Adds the regression line to the plot.
- **xlabel, ylabel, title**: Adds labels and a title for better understanding.
- **legend:** Displays the legend.

5. Summary and Insights

- **Slope (β1):** Indicates how much the dependent variable changes for a unit change in the independent variable.
- **Intercept (β0):** The predicted value of yyy when xxx is zero.
- **R-squared (R^2):** Evaluates how well the model fits the data.
- **Visualization:** Confirms if the regression line captures the trend in the data.
-

6. Deliverables

1. A Jupyter Notebook with implementation.
2. Visualizations of the regression line and residuals.
3. Answers to lab questions.

Lab Questions

1. What does the slope (β1) indicate for this dataset?
2. How well does the model predict y based on the R^2value?
3. Are there any patterns in the residuals that violate linear regression assumptions?