BE(E&C)	Data Science & Visualization Lab	
Experiment No.:07	**Implementation of Simple and Multiple Linear Regressions with scikit-learn in Python.**	**Page:** /6

- **Aim:** Implementation of Simple and Multiple Linear Regression with scikit-learn in Python.
- **Software Used:** Python 3.12, Jupyter Notebook
- **Learning Objective:**

By the end of this student should be able to:

1. **Understand the Concept**: Grasp the fundamental concepts of simple and multiple linear regression.
2. **Implement Models**: Use scikit-learn to implement simple and multiple linear regression models.
3. **Evaluate Models**: Assess the performance of the regression models using appropriate metrics.
4. **Visualize Results**: Create visualizations to interpret and present the results of the regression analyses.
5. **Apply Techniques**: Apply these regression techniques to real-world datasets.

- **Learning Outcomes:**

After performing the experiment students will be able to-

1. **Explain** the principles and purposes of linear regression, both simple and multiple.
2. **Code** simple and multiple linear regression models using scikit-learn.
3. **Evaluate** model performance with metrics such as Mean Squared Error (MSE) and R^2 Score.
4. **Visualize** data and regression results effectively using matplotlib.
5. **Apply** these techniques to solve practical problems in data analysis and prediction.

- **Theory:**

• **Linear Regression**:

- **Definition**: Linear regression is a statistical method that models the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to observed data.
- **Equation**: For simple linear regression, the model is represented as: y=mx+by = mx + by=mx+b Where:
  - yyy is the predicted value (dependent variable).
  - mmm is the slope of the line (coefficient).
  - xxx is the independent variable.

- bbb is the y-intercept.

- **Simple Linear Regression**:
  - Involves only one independent variable. It aims to find the line of best fit that minimizes the residual sum of squares (the difference between observed and predicted values).

- **Multiple Linear Regression**:
  - Involves two or more independent variables. The equation expands to: y=b0+b1x1+b2x2+...+bnxny = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_ny=b0+b1x1+b2x2+...+bnxn Where b0b_0b0 is the y-intercept, and b1,b2,...,bnb_1, b_2, ..., b_nb1,b2,...,bn are the coefficients for each independent variable.

- **Assumptions of Linear Regression**:
  - **Linearity**: The relationship between the dependent and independent variables should be linear.
  - **Independence**: Observations should be independent of one another.
  - **Homoscedasticity**: The variance of errors should be constant across all levels of the independent variable.
  - **Normality**: The residuals (errors) should be normally distributed.

- **Performance Metrics**:
  - **R^2 Score**: Represents the proportion of variance in the dependent variable that can be explained by the independent variables. Ranges from 0 to 1, with higher values indicating better model fit.
  - **Adjusted R^2**: Adjusts the R^2 value based on the number of predictors in the model, providing a more accurate measure when comparing models with different numbers of independent variables.
  - **Mean Squared Error (MSE)**: Measures the average of the squares of the errors. It provides a measure of how close predictions are to the actual outcomes:

    MSE=n1∑(yi−y^i)2

    - Where yiy_iyi are the actual values and y^i\hat{y}_iy^i are the predicted values.

- **Visualizing Results**:
  - Visualization is crucial for interpreting model performance. Scatter plots with regression lines help in understanding how well the model fits the data.

BE(E&C)	Data Science & Visualization Lab	
Experiment No.:07	**Implementation of Simple and Multiple Linear Regressions with scikit-learn in Python.**	**Page:** /6

Code:

```
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_squared_error, r2_score

# Load dataset (replace 'your_dataset.csv' with the actual dataset)
data = pd.read_csv('your_dataset.csv')

# Example dataset for Simple Linear Regression
# Assuming 'YearsExperience' as independent variable and 'Salary' as dependent variable
simple_data = data[['YearsExperience', 'Salary']]

# Split data into features and target
X_simple = simple_data[['YearsExperience']]
y_simple = simple_data['Salary']

# Split the data
X_train_simple, X_test_simple, y_train_simple, y_test_simple = train_test_split(X_simple, y_simple, test_size=0.2, random_state=42)

# Create and train the Simple Linear Regression model
simple_model = LinearRegression()
simple_model.fit(X_train_simple, y_train_simple)
```

```
# Make predictions
y_pred_simple = simple_model.predict(X_test_simple)

# Evaluate the Simple Linear Regression model
mse_simple = mean_squared_error(y_test_simple, y_pred_simple)
r2_simple = r2_score(y_test_simple, y_pred_simple)

print(f'Simple Linear Regression MSE: {mse_simple:.2f}')
print(f'Simple Linear Regression R²: {r2_simple:.2f}')

# Visualize Simple Linear Regression
plt.scatter(X_test_simple, y_test_simple, color='blue', label='Actual')
plt.plot(X_test_simple, y_pred_simple, color='red', label='Predicted')
plt.title('Simple Linear Regression')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()

# Example dataset for Multiple Linear Regression
# Assuming 'YearsExperience', 'Age', and 'EducationLevel' as independent variables
multiple_data = data[['YearsExperience', 'Age', 'EducationLevel', 'Salary']]

# Split data into features and target
X_multiple = multiple_data[['YearsExperience', 'Age', 'EducationLevel']]
```

```
y_multiple = multiple_data['Salary']


# Split the data
X_train_multiple, X_test_multiple, y_train_multiple, y_test_multiple =
train_test_split(X_multiple, y_multiple, test_size=0.2, random_state=42)


# Create and train the Multiple Linear Regression model
multiple_model = LinearRegression()
multiple_model.fit(X_train_multiple, y_train_multiple)


# Make predictions
y_pred_multiple = multiple_model.predict(X_test_multiple)
# Evaluate the Multiple Linear Regression model
mse_multiple = mean_squared_error(y_test_multiple, y_pred_multiple)
r2_multiple = r2_score(y_test_multiple, y_pred_multiple)
print(f'Multiple Linear Regression MSE: {mse_multiple:.2f}')
print(f'Multiple Linear Regression R²: {r2_multiple:.2f}')
```

❖ Output

The output will include:

- Mean Squared Error (MSE) and R^2 score for both Simple and Multiple Linear Regression models.
- Visual representation of the Simple Linear Regression results.

```
Simple Linear Regression MSE: 2500.00
Simple Linear Regression R²: 0.85


Multiple Linear Regression MSE: 2000.00
Multiple Linear Regression R²: 0.90
```

BE(E&C)	Data Science & Visualization Lab	
Experiment No.:07	**Implementation of Simple and Multiple Linear Regressions with scikit-learn in Python.**	**Page:** /6

❖ **Conclusion:**

_____
_____
_____
_____

❖ **Questions:**

1. What are the key differences between Simple Linear Regression and Multiple Linear Regression in terms of application and interpretation?
2. How does R^2 score help in assessing the fit of a regression model, and why might you prefer Adjusted R^2?
3. What are the implications of violating the assumptions of linear regression, particularly regarding linearity and homoscedasticity?
4. Describe the significance of each coefficient in a Multiple Linear Regression model and how to interpret them.
5. What strategies can be employed to improve a regression model if the initial performance metrics are unsatisfactory?