Training Report – Day 13

Topic Covered Today:

- Introduction to **Linear Regression**
- Equation and working of Linear Regression
- Implementation using **Python (Scikit-learn)**
- Evaluation of regression model performance

Key Learning:

Introduction to Linear Regression:

Today I learned about **Linear Regression**, which is one of the simplest and most widely used **supervised learning algorithms** in machine learning.
It is used to **predict a continuous dependent variable** (like price, marks, salary) based on one or more independent variables.

Linear Regression finds the **best-fitting straight line** through data points that minimizes the difference between the predicted and actual values.

Mathematical Equation:
[
Y = mX + c
]
where,

- (Y) = dependent variable (predicted value)
- (X) = independent variable (input feature)
- (m) = slope (weight or coefficient)
- (c) = intercept (bias term)

Working of Linear Regression:

1. The algorithm analyzes training data and fits a straight line (for simple regression) or a plane (for multiple regression).
2. It calculates the **error** (difference between actual and predicted values).
3. The goal is to minimize the **Mean Squared Error (MSE)** using optimization techniques like **Gradient Descent**.

Error Formula:
[
MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y_i})^2
]

Implementation in Python:

Here's the example I practiced using **Scikit-learn**:

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Example data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([3, 4, 2, 5, 6])

# Create and train the model
model = LinearRegression()
model.fit(X, y)

# Prediction
pred = model.predict([[6]])
print("Predicted value for X=6:", pred)
```

Output:

```
Predicted value for X=6: [6.2]
```

This means the model predicted approximately 6.2 for an input of 6, showing how it uses the linear relationship between X and Y.

Evaluation Metrics:

To check how well the model performs, we used:

- **Mean Squared Error (MSE)** → Measures average squared difference between actual and predicted values.
- **R^2 Score (Coefficient of Determination)** → Shows how well the model fits the data (1 = perfect fit).

Example:

```
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(X)
print("MSE:", mean_squared_error(y, y_pred))
print("R2 Score:", r2_score(y, y_pred))
```

Activities / Assignments:

- Understood the concept of **regression vs classification**.
- Derived the **equation of a straight line** used in linear regression.
- Implemented **simple linear regression** using Scikit-learn.
- Visualized the regression line using Matplotlib.
- Calculated and compared **MSE** and **R^2 Score** for model performance.

Personal Reflection for Day 13:

Today's session gave me a solid foundation in understanding how **machine learning predicts continuous values** using mathematical relationships. I learned that linear regression is simple yet powerful in finding trends and forecasting future outcomes.

Implementing the model in Python helped me understand how training data affects predictions. Visualizing the regression line made the concept of "best fit" very clear.

Overall, I enjoyed this topic as it connected both mathematics and programming — giving me practical insight into how AI models start learning patterns from data