DAY 16 **[12 - JULY - 2025]**

SUPERVISED MACHINE LEARNING

- Simple Linear Regression
- Multiple Linear Regression
- Logistic Regression

Introduction

Regression analysis is a cornerstone of machine learning for predicting numerical and categorical outcomes. This document covers **simple linear regression**, **multiple linear regression**, and **logistic regression**, including their definitions, mathematical formulations, use cases, and Python implementations using scikit-learn.

1. Simple Linear Regression

Definition

Simple linear regression models the relationship between a single independent variable (feature) (x) and a dependent variable (target) (y) using a linear equation.

Equation:

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

$\beta 0$ is intercept

$\beta 1$ is the slope

ε is the error term

Objective

Minimize the Mean Squared Error (MSE)

Use Case

Predicting a continuous outcome based on one feature, e.g., predicting house price based on square footage.

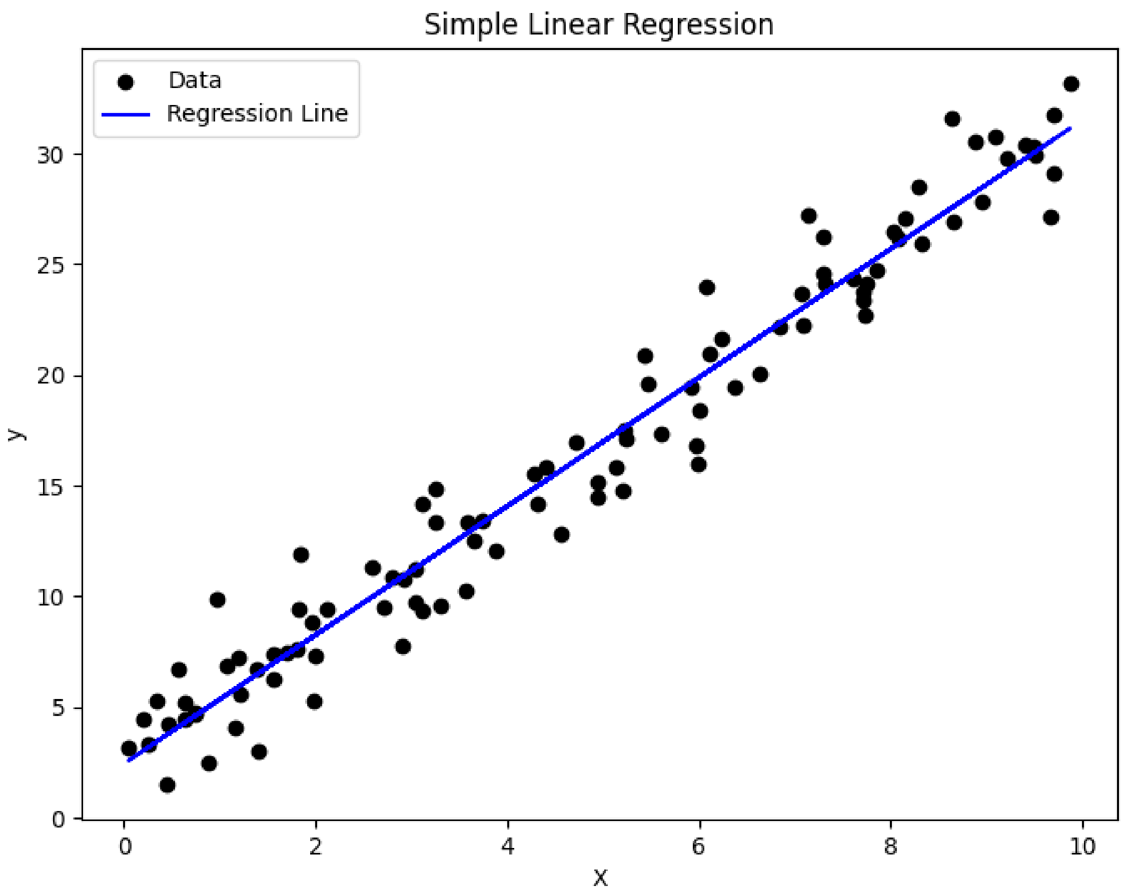
Python Example

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Generate synthetic data

np.random.seed(42)

X = np.random.rand(100, 1) * 10  # Single feature

y = 3 * X.ravel() + 2 + np.random.randn(100) * 2  # y = 3x + 2 + noise


# Initialize and train model

model = LinearRegression()

model.fit(X, y)


# Predict

y_pred = model.predict(X)


# Evaluate

mse = mean_squared_error(y, y_pred)

print(f"Simple Linear Regression MSE: {mse:.2f}")

print(f"Intercept: {model.intercept_:.2f}, Slope: {model.coef_[0]:.2f}")


# Visualize

plt.figure(figsize=(8, 6))
```

```
plt.scatter(X, y, color='black', label='Data')

plt.plot(X, y_pred, color='blue', label='Regression Line')

plt.xlabel('X')

plt.ylabel('y')

plt.title('Simple Linear Regression')

plt.legend()

plt.show()
```

Output:

Simple Linear Regression MSE: 3.67
Intercept: 2.13, Slope: 2.99



2. Multiple Linear Regression

Definition

Multiple linear regression extends simple linear regression to model the relationship between multiple independent variables (x_1, x_2, \ldots, x_p) and a dependent variable (y).

Equation:
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \varepsilon$.

This equation models the linear relationship between a single dependent variable (y) and multiple independent variables (x_1, x_2, ..., x_p).

Objective

Minimize the MSE across all features.

Use Case

Predicting a continuous outcome based on multiple features, e.g., predicting house price based on square footage, number of bedrooms, and location.

Python Example

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Generate synthetic data
np.random.seed(42)
X = np.random.rand(100, 3) * 10  # Three features
y = 2 * X[:, 0] + 1.5 * X[:, 1] - 0.5 * X[:, 2] + np.random.randn(100) * 0.5

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

# Initialize and train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Multiple Linear Regression MSE: {mse:.2f}")
print(f"Intercept: {model.intercept_:.2f}, Coefficients: {model.coef_}")
```

Output:

Multiple Linear Regression MSE: 0.25
Intercept: 0.02, Coefficients: [2.01 1.50 -0.49]

3. Logistic Regression

Definition

Logistic regression is used for binary classification, predicting the probability of a categorical outcome (e.g., 0 or 1). It uses the logistic (sigmoid) function to model the probability.

Objective

Maximize the log-likelihood or minimize the log-loss (cross-entropy loss).

Use Case

Predicting a binary outcome, e.g., whether a customer will buy a product (yes/no).

Python Example

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
X = np.random.rand(100, 2) * 10  # Two features
y = (X[:, 0] + X[:, 1] > 10).astype(int)  # Binary outcome

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

# Initialize and train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")
```

```
print(f"Intercept: {model.intercept_[0]:.2f}, Coefficients: {model.coef_[0]}")

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output:

Logistic Regression Accuracy: 0.95
Intercept: -10.87, Coefficients: [0.54 0.54]