scikit-learn

Linear Regression

Linear Regression using scikit-learn

Linear Regression is a statistical method to model the relationship between a dependent variable and one or more independent variables.

It's widely used for prediction and forecasting where a continuous relationship between variables is assumed.

Now, we'll explore how to implement a simple linear regression model using Python's **scikit-learn** library.

Setting Up Your Environment

Before we begin, ensure you have Python and pip installed on your system.

Install scikit-learn by running: `pip install scikit-learn`

This library provides simple and efficient tools for data mining and data analysis.

Generating Sample Data with Random Function

```
import numpy as np
import matplotlib.pyplot as plt
# Generating random dataset
np.random.seed(42) # For reproducibility
X = np.random.rand(100, 1) * 10 # Generate 100 random points
y = 2.5 * X + np.random.randn(100, 1) * 2 + 5 # y = mx + c + noise
# Visualizing the dataset
plt.scatter(X, y)
plt.title("Generated Linear Dataset")
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```

Training the Model

Split the dataset into training and testing sets to evaluate the model's performance.

Train the model using the `LinearRegression` class from scikit-learn.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
```

Making Predictions and Evaluating the Model

Use the trained model to make predictions on the test set.

Evaluate the model's accuracy by calculating the Mean Squared Error (MSE) between the predicted and actual values.

```
from sklearn.metrics import mean_squared_error


y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

Understanding Model Coefficients

The model's coefficients indicate the importance of each feature in predicting the target variable.

`model.coef_` provides the slope(s), and `model.intercept_` provides the intercept.

These values help understand the linear relationship modeled by our regression.

```
slope = model.coef_

intercept = model.intercept_

print(f"Slope (model.coef_): {slope}")

print(f"Intercept (model.intercept_): {intercept}")
```

Visualizing the Trained Linear Model

```
# Predicting y values using the trained model for plotting
line_x = np.linspace(X.min(), X.max(), 100)  # Generating points to plot the regression line
line_y = model.predict(line_x.reshape(-1, 1))
plt.scatter(X, y, color='blue', label='Original data')
# Plotting the linear regression line
plt.plot(line_x, line_y, color='red', label='Fitted line')
plt.title('Linear Regression Model Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```