Task #1 Giorgi GHIBRADZE

Linear regression is a statistical method used to model the relationship between a dependent variable (also called the target or response variable) and one or more independent variables (also called predictors or features).

The goal of linear regression is to find the best-fitting straight line that describes the relationship between the variables.

The linear regression model can be expressed mathematically as:

$$y = \beta0 + \beta1{*}x1 + \beta2{*}x2 + \dots + \beta n{*}xn + \varepsilon$$

Where:

y - is the dependent variable

x1, x2, ..., xn - are the independent variables

β0 - is the y-intercept (the value of **y** when all **x** values are **0**)

β1, β2, ..., βn are the regression coefficients, which represent the change in **y** for a one-unit change in the corresponding **x** variable

ε is the error term, which represents the difference between the observed values of **y** and the predicted values of **y**

The goal of the linear regression model is to estimate the values of the regression coefficients (**β0**, **β1**, ..., **βn**) that minimize the sum of the squared differences between the observed and predicted values of **y**.

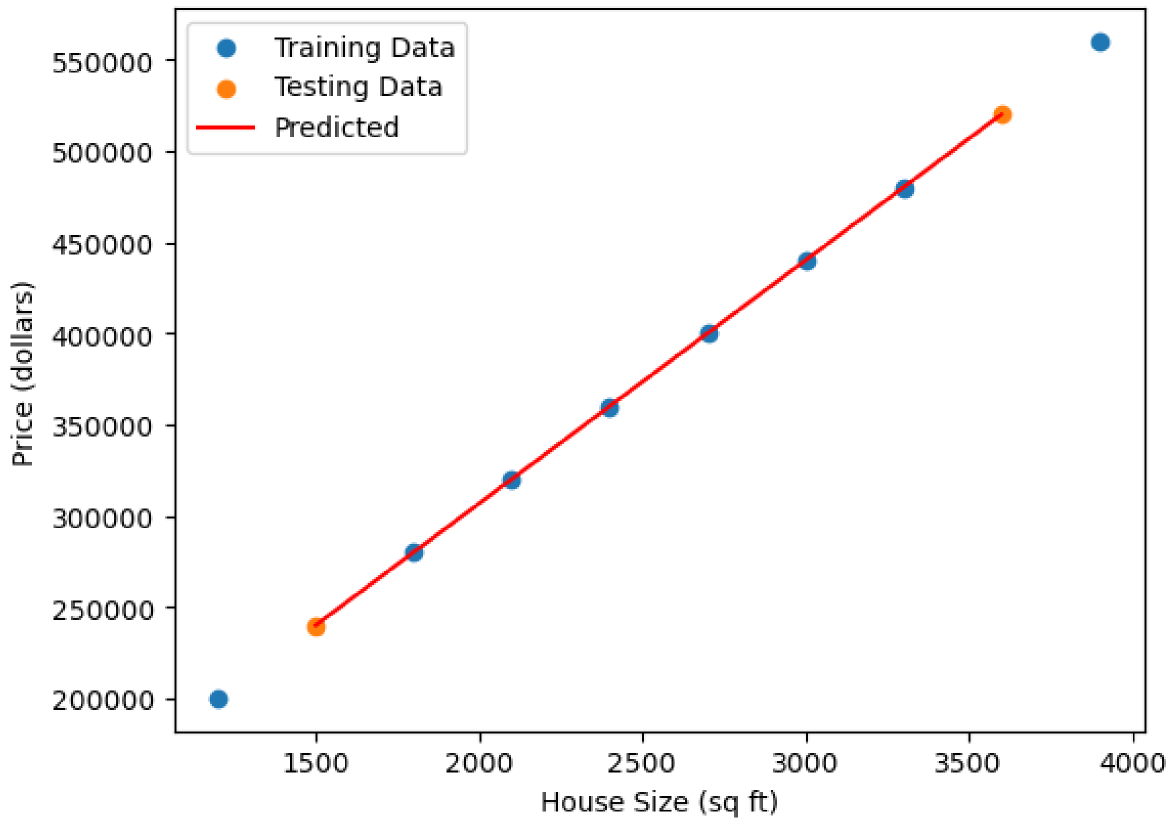
Practical Example

Let's consider a dataset that represents the relationship between the size of a house (in square feet) and its price (in dollars). I'll use the `sklearn` library in Python to build a linear regression model.

```
import numpy as np

import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score
```

```
data = pd.DataFrame({
    'house_size': [1200, 1500, 1800, 2100, 2400, 2700, 3000, 3300, 3600, 3900],
    'price': [200000, 240000, 280000, 320000, 360000, 400000, 440000, 480000, 520000, 560000]
})

X_train, X_test, y_train, y_test = train_test_split(data['house_size'].values.reshape(-1, 1),
                                                    data['price'].values,
                                                    test_size=0.2,
                                                    random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2:.2f}')

import matplotlib.pyplot as plt
plt.scatter(X_train, y_train, label='Training Data')
plt.scatter(X_test, y_test, label='Testing Data')
plt.plot(X_test, y_pred, color='r', label='Predicted')
plt.xlabel('House Size (sq ft)')
plt.ylabel('Price (dollars)')
plt.legend()
plt.show()
```

```
R-squared: 1.00
```



In this example, we have a dataset with two columns: `house_size` and `price`. I split the dataset into training and testing sets, create a linear regression model using `sklearn.linear_model.LinearRegression`, and fit the model to the training data.

I then evaluate the model's performance on the testing data using the mean squared error (MSE) and the coefficient of determination (R-squared). The MSE measures the average squared difference between the predicted and actual values, while the R-squared value indicates the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

Finally, I plot the actual and predicted values to visualize the fit of the linear regression model.

The output of the code will show the MSE and R-squared values, as well as a plot of the actual and predicted values. The linear regression model can then be used to make predictions on new, unseen data.

This is a simple example of a linear regression model, but the concepts and techniques can be extended to more complex scenarios with multiple independent variables. The `sklearn` library provides a wide range of tools and methods for building and evaluating linear regression models, as well as other machine learning algorithms.