Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

División de ingenierías para la integración ciber-humana

Profesora: Gerardo Garcia Gil
Juan Antonio Pérez Juárez
215660996

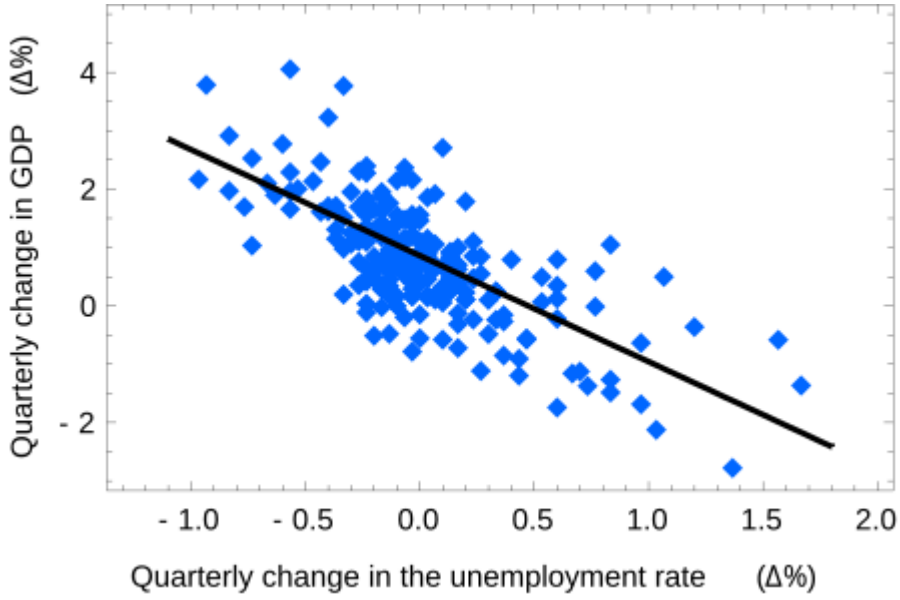
Introduction

Linear regression is one of the most fundamental and widely used techniques in statistical modeling and machine learning. Its primary objective is to establish a linear relationship between a dependent variable and one or more independent variables by fitting a line that best describes the observed data. This method has found extensive applications across various domains, including engineering, economics, biology, and social sciences, due to its simplicity and interpretability.

In the context of predictive analytics, linear regression serves as a foundational tool for forecasting and trend analysis. By minimizing the sum of squared errors between predicted and actual values, it provides an optimal solution for estimating unknown parameters. The mathematical tractability of linear regression enables efficient computation and straightforward extension to more complex models, such as multiple regression and polynomial regression.

Simple linear regression

In statistics, simple linear regression (SLR) is a linear regression model with a single explanatory variable. That is, it concerns two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variable. The adjective simply refers to the fact that the outcome variable is related to a single predictor.



Okun's law in macroeconomics is an example of simple linear regression. Here the dependent variable (GDP growth) is presumed to be in a linear relationship with the changes in the unemployment rate.

It is common to make the additional stipulation that the ordinary least squares (OLS) method should be used: the accuracy of each predicted value is measured by its squared residual (vertical distance between the point of the data set and the fitted line), and the goal is to make the sum of these squared deviations as small as possible. In this case, the slope of the fitted line is equal to the correlation between y and x corrected by the ratio of standard deviations of these variables. The intercept of the fitted line is such that the line passes through the center of mass (x, y) of the data points.

Consider the model function:

$$y = \alpha + \beta x,$$

which describes a line with slope β and y-intercept α. In general, such a relationship may not hold exactly for the largely unobserved population of values of the independent and dependent variables; we call the unobserved deviations from the above equation the errors. Suppose we observe n data pairs and call them {(xi, yi), i = 1, ..., n}. We can describe the underlying relationship between yi and xi involving this error term εi by

$$y_i = \alpha + \beta x_i + \varepsilon_i.$$

This relationship between the true (but unobserved) underlying parameters α and β and the data points is called a linear regression model.

The goal is to find estimated values $\widehat{\alpha}$ and $\widehat{\beta}$ for the parameters α and β which would provide the "best" fit in some sense for the data points. As mentioned in the introduction, in this article the "best" fit will be understood as in the least-squares approach: a line that minimizes the sum of squared residuals (see also Errors and residuals) $\hat{\varepsilon}_i$ (differences between actual and predicted values of the dependent variable y), each of which is given by, for any candidate parameter values α and β.

$$\hat{\varepsilon}_i = y_i - \alpha - \beta x_i.$$

In other words, $\widehat{\alpha}$ and $\widehat{\beta}$ solve the following minimization problem:

$$(\hat{\alpha},\, \hat{\beta}) = \operatorname{argmin}(Q(\alpha, \beta)),$$

where the objective function Q is:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 \ .$$

Example

Suppose we want to analyze the relationship between the number of hours a student studies (x) and their exam score (y). We collect the following data:

Hours Studied (x)	**Exam Score (y)**
2	**81**
4	**93**
6	**91**
8	**97**
10	**99**

We aim to find a linear relationship of the form:
$y = \beta 0 + \beta 1 x$

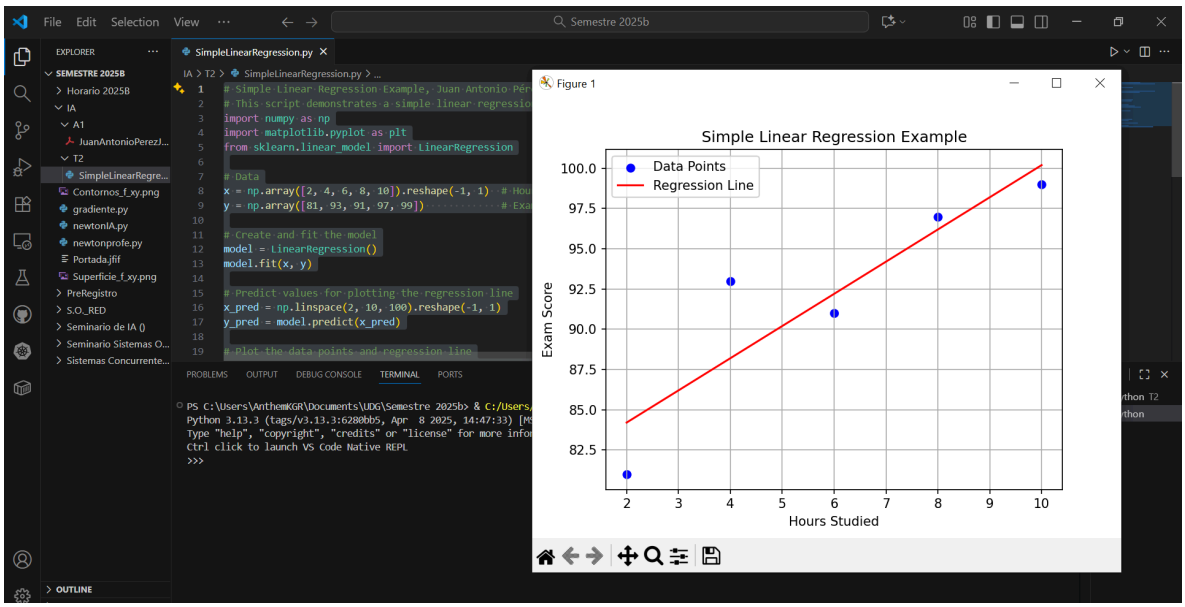
By applying simple linear regression to this dataset, we can estimate the coefficients ($\beta 0$ and $\beta 1$), allowing us to predict the exam score for any given number of study hours.

Let's Program it

Python

```python
# Simple Linear Regression Example, Juan Antonio Pérez Juárez
# This script demonstrates a simple linear regression using Python's sklearn library.
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Data
x = np.array([2, 4, 6, 8, 10]).reshape(-1, 1)  # Hours studied (reshaped for sklearn)
y = np.array([81, 93, 91, 97, 99])             # Exam scores

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

# Predict values for plotting the regression line
x_pred = np.linspace(2, 10, 100).reshape(-1, 1)
y_pred = model.predict(x_pred)

# Plot the data points and regression line
plt.scatter(x, y, color='blue', label='Data Points')
plt.plot(x_pred, y_pred, color='red', label='Regression Line')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.title('Simple Linear Regression Example')
plt.legend()
plt.grid(True)
plt.show()


# Print the coefficients
print(f"Intercept (β₀): {model.intercept_:.2f}")
print(f"Slope (β₁): {model.coef_[0]:.2f}")

# Predict an exam score for a student who studied 7 hours
hours = 7
predicted_score = model.predict([[hours]])
print(f"Predicted exam score for {hours} hours studied: {predicted_score[0]:.2f}")
```

Screenshots:


File Edit Selection View
Semestre 2025b
EXPLORER
SEMESTRE 2025B
Horario 2025B
IA
A1
JuanAntonioPerezJ...
T2
SimpleLinearRegre...
Contornos_f_xy.png
gradiente.py
newtonIA.py
newtonprofe.py
Portada.jfif
Superficie_f_xy.png
PreRegistro
S.O._RED
Seminario de IA ()
Seminario Sistemas O...
Sistemas Concurrente...
OUTLINE
SimpleLinearRegression.py
IA > T2 > SimpleLinearRegression.py > ...
# Simple Linear Regression Example, Juan Antonio Pér
# This script demonstrates a simple linear regressio
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Data
x = np.array([2, 4, 6, 8, 10]).reshape(-1, 1)
y = np.array([81, 93, 91, 97, 99])
# Create and fit the model
model = LinearRegression()
model.fit(x, y)
# Predict values for plotting the regression line
x_pred = np.linspace(2, 10, 100).reshape(-1, 1)
y_pred = model.predict(x_pred)
# Plot the data points and regression line
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\AnthemKGR\Documents\UDG\Semestre 2025b> & C:/Users
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [M
Type "help", "copyright", "credits" or "license" for more info
Ctrl click to launch VS Code Native REPL
>>>
Figure 1
Simple Linear Regression Example
Data Points
Regression Line
Exam Score
Hours Studied
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
2
3
4
5
6
7
8
9
10

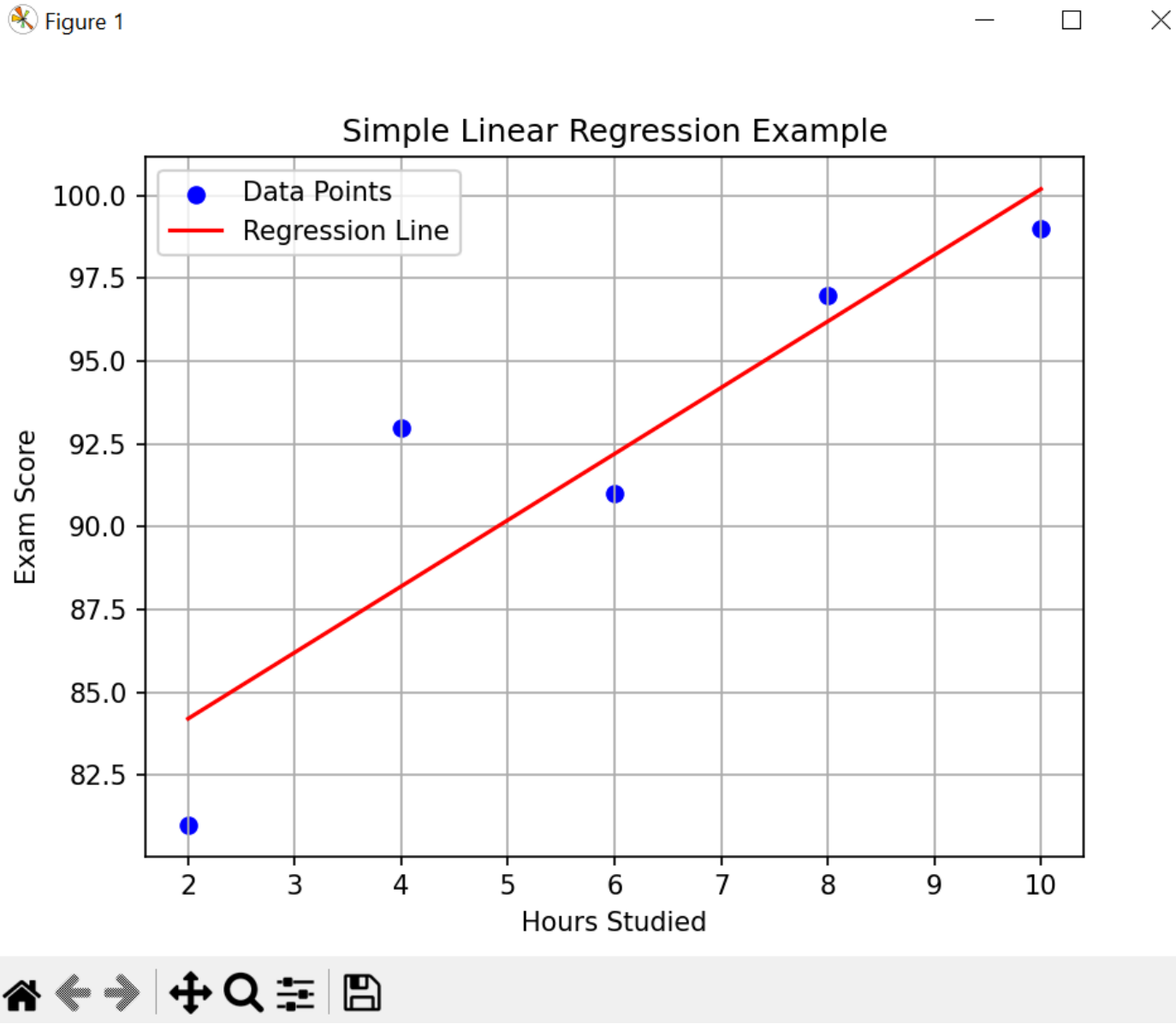


Figure 1
Simple Linear Regression Example
Data Points
Regression Line
Exam Score
Hours Studied
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
2
3
4
5
6
7
8
9
10

Data: Five pairs of hours studied and exam scores.
Model: Uses LinearRegression from scikit-learn.
Graph: Blue points show the data, red line shows the fitted regression.
Coefficients: Prints intercept and slope.
Prediction: Shows how to predict a score for a new value (e.g., 7 hours).

Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of MLR is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

$y_i = \text{dependent variable}$

$x_i = \text{explanatory variables}$

$\beta_0 = \text{y-intercept (constant term)}$

$\beta_p = \text{slope coefficients for each explanatory variable}$

$\epsilon = \text{the model's error term (also known as the residuals)}$

The MLR model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables
- The independent variables are not too highly correlated with each other
- yi observations are selected independently and randomly from the population
- Residuals should be normally distributed with a mean of 0 and variance σ

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R2 always increases as more predictors are added to the MLR model, even though the predictors may not be related to the outcome variable.

Thus, R2 by itself can't be used to identify which predictors should be included in a model and which should be excluded. R2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables, and 1 indicates that the outcome can be predicted without error from the independent variables.

When interpreting the results of multiple regression, beta coefficients are valid while holding all other variables constant ("all else equal"). The output from a multiple regression can be displayed horizontally as an equation or vertically in table form.

Ordinary linear squares (OLS) regression compares the response of a dependent variable given a change in some explanatory variables. However, a dependent variable is rarely explained by only one variable. In this case, an analyst uses multiple regression, which attempts to explain a dependent variable using more than one independent variable.

Multiple regressions can be linear and nonlinear. MLRs are based on the assumption that there is a linear relationship between both the dependent and independent variables. It also assumes no major correlation between the independent variables.

Example:
Suppose we want to predict a student's exam score (y) based on both:
Hours studied (x1)
Number of hours slept the night before (x2)

Hours Studied (x1)	**Hours Slept (x2)**	**Exam Score (y)**
2	7	81
4	6	89
6	8	92
8	8	95
10	9	99

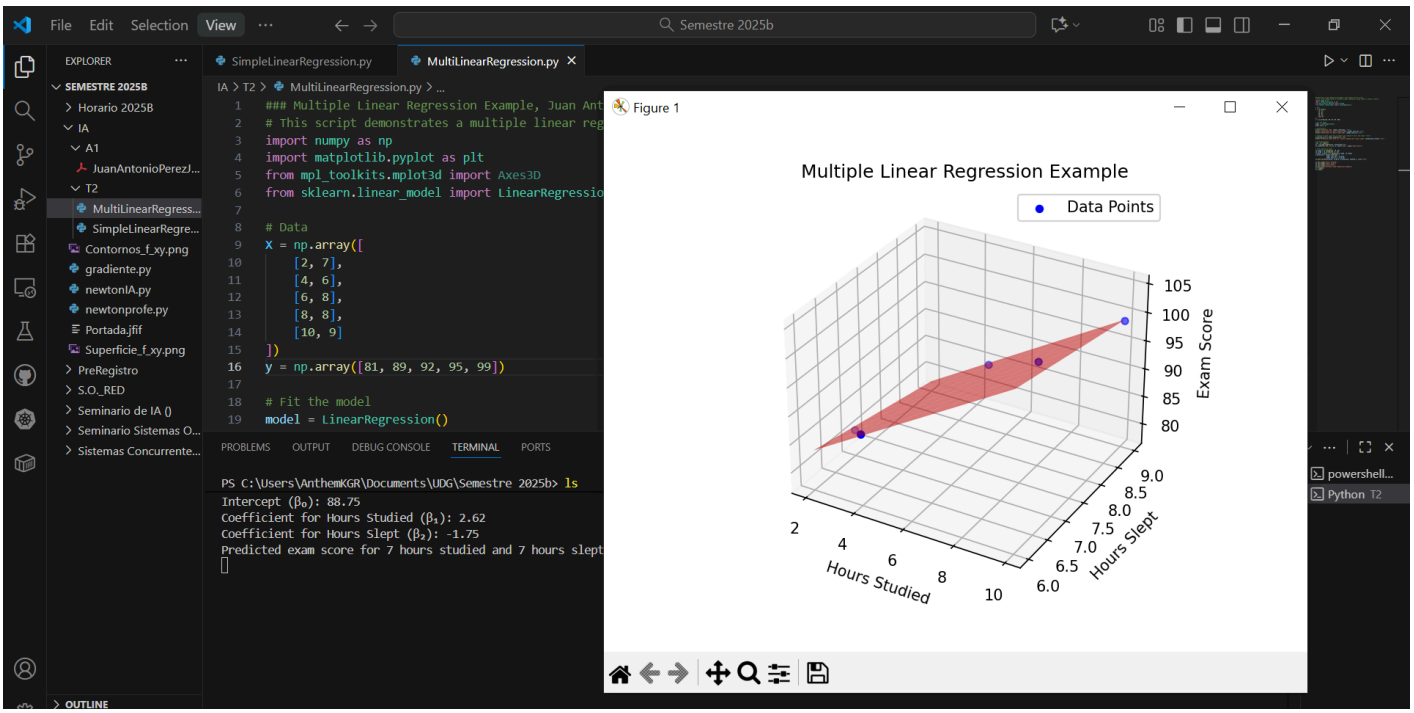
The multiple linear regression model is:
$y = \beta 0 + \beta 1 x1 + \beta 2 x2$

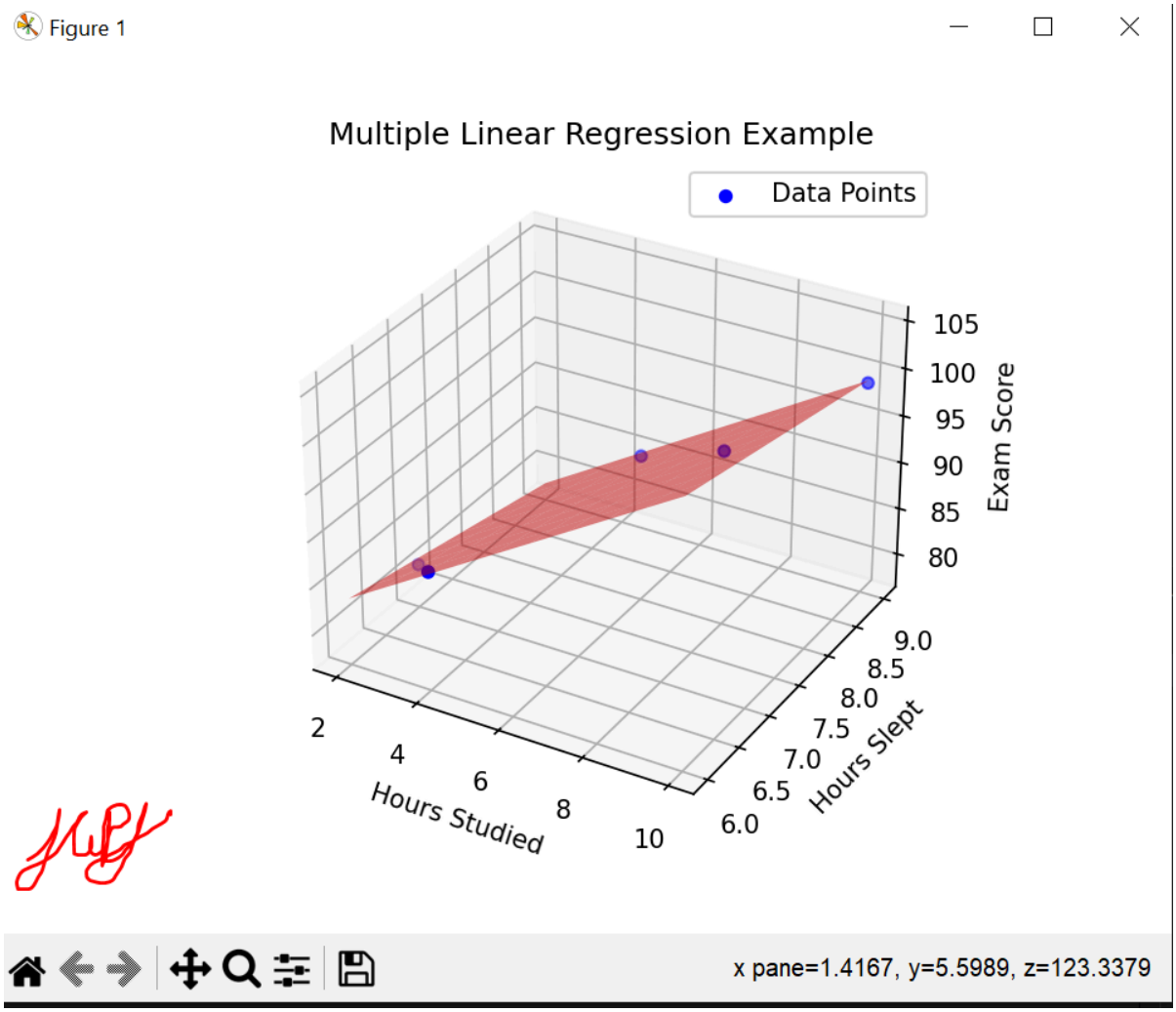
Lets program it

```python
### Multiple Linear Regression Example, Juan Antonio Pérez Juárez
# This script demonstrates a multiple linear regression using Python's sklearn library
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.linear_model import LinearRegression

# Data
X = np.array([
    [2, 7],
    [4, 6],
    [6, 8],
    [8, 8],
    [10, 9]
])
y = np.array([81, 89, 92, 95, 99])
```

```
# Fit the model
model = LinearRegression()
model.fit(X, y)

# Coefficients
print(f"Intercept (β₀): {model.intercept_:.2f}")
print(f"Coefficient for Hours Studied (β₁): {model.coef_[0]:.2f}")
print(f"Coefficient for Hours Slept (β₂): {model.coef_[1]:.2f}")

# Predict an exam score for a student who studied 7 hours and slept 7 hours
predicted_score = model.predict([[7, 7]])
print(f"Predicted exam score for 7 hours studied and 7 hours slept:
{predicted_score[0]:.2f}")

# 3D Visualization
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0], X[:,1], y, color='blue', label='Data Points')

# Plot regression plane
x1_range = np.linspace(2, 10, 10)
x2_range = np.linspace(6, 9, 10)
x1_grid, x2_grid = np.meshgrid(x1_range, x2_range)
y_pred_grid = (model.intercept_ +
               model.coef_[0] * x1_grid +
               model.coef_[1] * x2_grid)
ax.plot_surface(x1_grid, x2_grid, y_pred_grid, alpha=0.5, color='red')

ax.set_xlabel('Hours Studied')
ax.set_ylabel('Hours Slept')
ax.set_zlabel('Exam Score')
ax.set_title('Multiple Linear Regression Example')
plt.legend()
plt.show()
```

Screenshots





Data: Exam scores predicted by both hours studied and hours slept.
Model: Uses LinearRegression from scikit-learn.

Coefficients: Prints intercept and both slopes.
Prediction: Shows how to predict a score for new values.
Graph: 3D scatter plot and regression plane for visual understanding.

Conclusion

As I explored the world of linear regression, I realized it's much more than just a mathematical technique. It's a bridge between raw data and meaningful understanding.

Personally, learning about these models has taught me the value of curiosity and perseverance. Every dataset tells a story, and with the right tools, we can listen, interpret, and even predict the next chapter. The journey from confusion to clarity, from numbers to knowledge, is what makes data science so rewarding.

I'm not in mathematics really, because I'm really bad at it, but when I was reading to do this activity, it felt like an oppenheimer in the moment when Bohr asked if he could hear the music.

In the end, linear regression is not just about lines and equations; it's about connecting with the data, finding meaning, and using that insight to make a difference.

*In the past I heard a phrase that could be used at this moment for this topic. "**Creativity is the art of connecting two variables that, at first glance, have nothing in common.".***

The image in the first page is made on Leonardo.ai by the way.

Referencias

Wikipedia contributors, "Simple linear regression," Wikipedia, Aug. 04, 2025. https://en.wikipedia.org/wiki/Simple_linear_regression

A. Hayes, "Multiple Linear Regression (MLR): definition, formula, and example," Investopedia, Apr. 14, 2025. https://www.investopedia.com/terms/m/mlr.asp

Scikit-Learn, "GitHub - scikit-learn/scikit-learn: scikit-learn: machine learning in Python," GitHub. https://github.com/scikit-learn/scikit-learn

Wikipedia contributors, "Scikit-learn," Aug. 06, 2025. https://en.wikipedia.org/wiki/Scikit-learn