# Objectives

* Implement Linear regression using the normal equation
* Introducing linear regression in sickit-learn

# Lab Tool(s)

<https://www.kaggle.com/>

# Lab Deliverables

Submit a pdf document to Blackboard containing your solution to the lab assessment at the end of this document.

# Linear Regression using the Normal Equation

**Step0:** Recall,

A linear model makes a prediction by simply computing a weighted sum of the input features, plus a constant called the bias term (also called the intercept term), as shown in the Equation;

Equation 4-1. Linear Regression model prediction

y = θ0 + θ1x1 + θ2x2 + ⋯ + θnxn

• ŷ is the predicted value.

• n is the number of features.

• xi is the ith feature value.

• θj is the jth model parameter (including the bias term θ0 and the feature weights θ1, θ2, ⋯, θn).

**Step1:** We will start by randomly generating linear-looking data, and plotting it

**import** **numpy** **as** **np**

X = 2 \* np.random.rand(100, 1)

y = 4 + 3 \* X + np.random.randn(100, 1)

\* NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/[3][4] (NUM-pee)) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

X is a vector of random numbers

Y (the predicted output) is a linear function of X. *y* = 4 + 3*x*1 + Gaussian noise

import matplotlib.pyplot as plt

plt.plot(X, y, "b.")

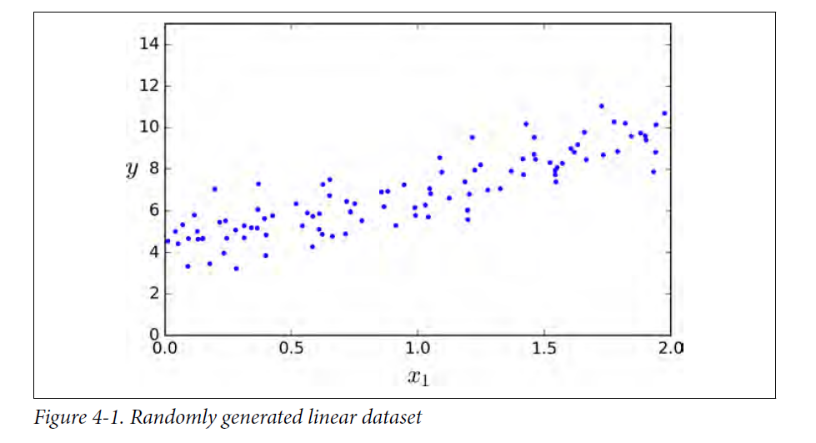
plt.xlabel("$x\_1$", fontsize=18)

plt.ylabel("$y$", rotation=0, fontsize=18)

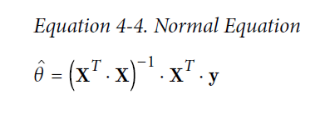
plt.axis([0, 2, 0, 15])

save\_fig("generated\_data\_plot")

plt.show()



**Step2:** We will compute the value of the parameter using the normal equation



We will use the inv() function from NumPy’s Linear Algebra module (np.linalg) to compute the inverse of a matrix, and the dot() method for matrix multiplication:

X\_b = np.c\_[np.ones((100, 1)), X] *# add x0 = 1 to each instance*

theta\_best = np.linalg.inv(X\_b.T.dot(X\_b)).dot(X\_b.T).dot(y)

**Step3:** print theta\_best and compare the learned model with the true one (*y* = 4 + 3*x*1 + Gaussian noise)

theta\_best

The first value represents θ0 and the second value is for θ1.

**Step4:** making predictions using the learned parameters.

X\_new = np.array([[0], [2]])

X\_new\_b = np.c\_[np.ones((2, 1)), X\_new] *# add x0 = 1 to each instance*

y\_predict = X\_new\_b.dot(theta\_best)

Plot the predictions (as a red line) and the actual data points (as blue dots)

plt.plot(X\_new, y\_predict, "r-")

plt.plot(X, y, "b.")

plt.axis([0, 2, 0, 15])

plt.show()

Note: The normal equations becomes very slow when the number of features grows large (e.g., 100,000). This is because it computes the inverse in O(n3) which is an n x n matrix, where n is the number of features.

# Linear Regression in Sickit-learn

**Scikit-learn** (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**Step1:** fitting a model

**from** **sklearn.linear\_model** **import** LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X, y)

lin\_reg.intercept\_, lin\_reg.coef\_

Note that Scikit-Learn separates the bias term (intercept\_) from the feature weights (coef\_).

**Step2:** Making predictions

lin\_reg.predict(X\_new)

Note: the linearregression function of sickit-learn is not based on the normal equation.

# Lab Assessment

**Step1:** Create a new notebook and name it “CCAI312\_YOURSTUDENTID\_Lab4”

**Step2:** Generate the following data

num\_hours\_studied = np.array([1, 3, 3, 4, 5, 6, 7, 7, 8, 8, 10])

exam\_score = np.array([18, 26, 31, 40, 55, 62, 71, 70, 75, 85, 97])

where num\_hours\_studied represents X and exam\_score is y.

**Step3:** visualize the data using scatter plots

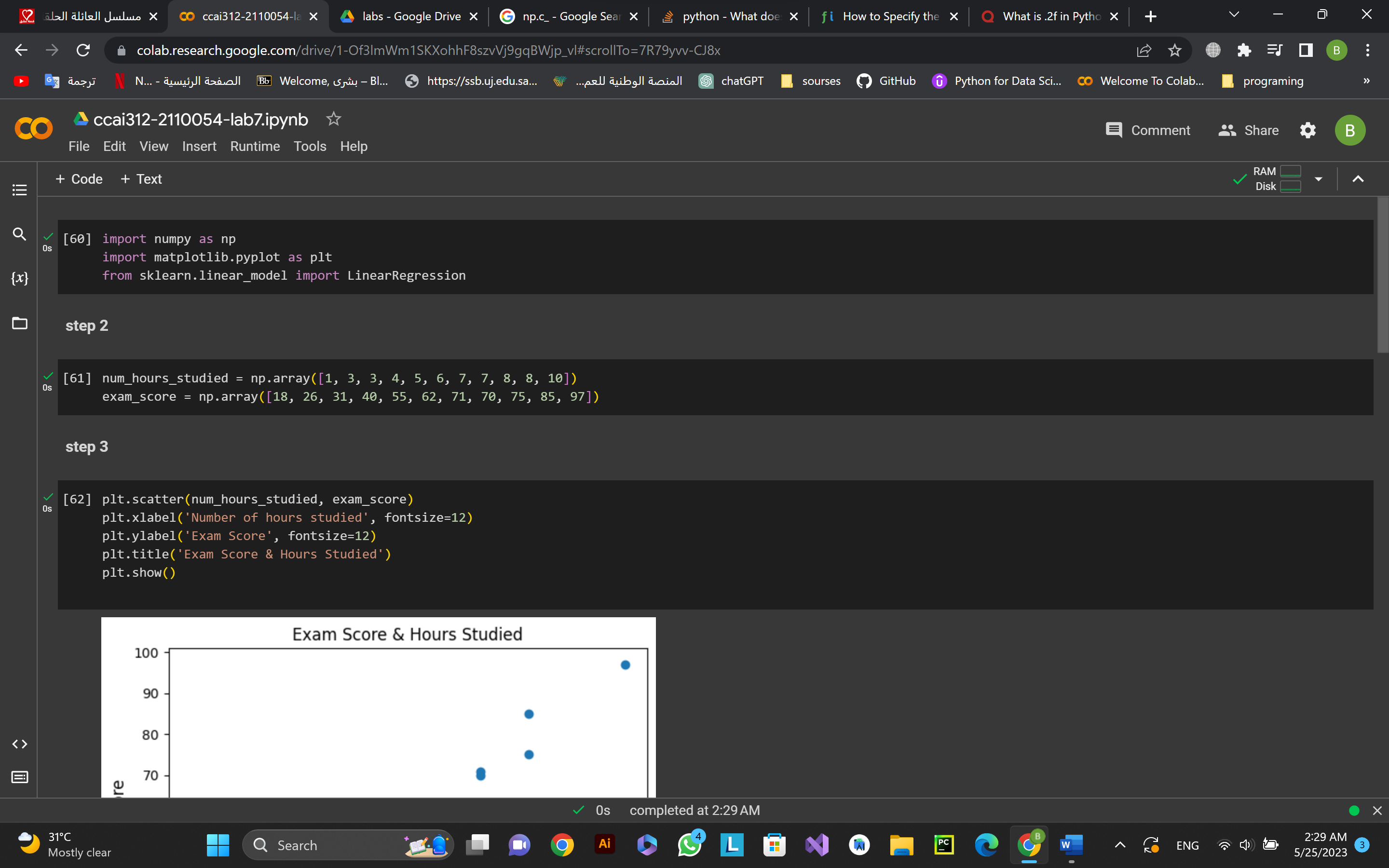
**Step4:** Fit a linear regression using the normal equation and plot the results

**Step5:** What is the predicted value (y\_predicted) when x is 9?

**Step6:** Submit a pdf document containing **your code and answers** to Blackboard, name the file as: CCAI312\_YOURSTUDENTID\_Lab4.pdf.

References:

Hands-on machine learning with sickit-learn and tensor flow, Chapter 4



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated