Kingdom of Saudi Arabia **Ministry of Education** **University of Jeddah** **College of Computer Science and Engineering** **Department of Computer Science and Artificial Intelligence**		**المملكة العربية السعودية** **وزارة التعليم** **جامعة جدّة** **كلية علوم وهندسة الحاسب** **قسم علوم الحاسب والذكاء الاصطناعي**

Lab 7

CCAI 312 Pattern Recognition

Third Trimester 2023

Student Name: Ruba Khalid Alsulami

Student ID: 2110618

Part 1

Lab Assessment

Step1: Create a new notebook and name it "CCAI312_YOURSTUDENTID_Lab4"

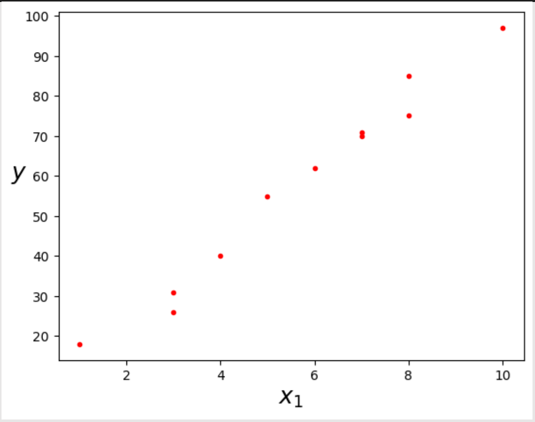
Step2: Generate the following data
num_hours_studied = np.array([1, 3, 3, 4, 5, 6, 7, 7, 8, 8, 10])
exam_score = np.array([18, 26, 31, 40, 55, 62, 71, 70, 75, 85, 97])

```
import numpy as np
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

```
import matplotlib.pyplot as plt
plt.scatter(num_hours_studied, exam_score)
plt.xlabel('Number of Hours Studied')
plt.ylabel('Exam Score')
plt.title('Exam Score VS Number of Hours Studied')
plt.show()
```



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

```
X = np.column_stack((np.ones_like(num_hours_studied),
num_hours_studied))
theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(exam_score)
output: array([4.27819549, 9.40225564])
```

Step5: What is the predicted value (y_predicted) when x is 9?

```
x_new = np.array([1, 9])
y_predicted = x_new.dot(theta)
print(y_predicted)
output: 88.89849624060155
```

Part 2

Lab Assessment

Step1: Create a new notebook and name it "CCAI312_YOURSTUDENTID_Lab5"

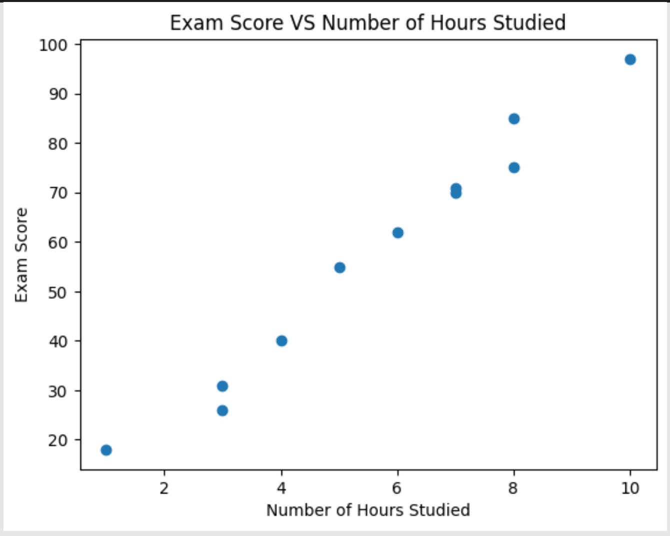
Step2: Generate the following data
num_hours_studied = np.array([1, 3, 3, 4, 5, 6, 7, 7, 8, 8, 10])
exam_score = np.array([18, 26, 31, 40, 55, 62, 71, 70, 75, 85, 97])

```
import numpy as np
import matplotlib.pyplot as plt
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

```
plt.scatter(num_hours_studied, exam_score)
plt.xlabel('Number of Hours Studied')
plt.ylabel('Exam Score')
plt.title('Exam Score VS Number of Hours Studied')
plt.show()
```



Step4: Fit a linear regression using Batch Gradient Descent and plot the results. Report the learned theta.

```
# Add bias term to X
X = np.c_[np.ones(num_hours_studied.shape[0]), num_hours_studied]
y = exam_score.reshape(-1, 1)

# Set hyperparameters
```

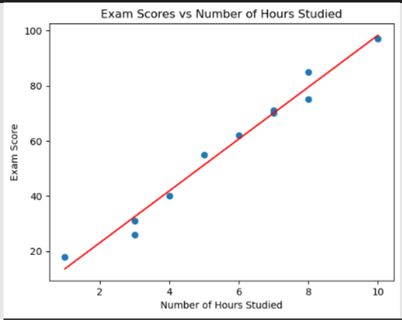
```
learning_rate = 0.01
n_iterations = 1000

# Initialize theta randomly
np.random.seed(0)
theta = np.random.randn(2,1)

# Batch Gradient Descent
for iteration in range(n_iterations):
    gradients = 2/X.shape[0] * X.T.dot(X.dot(theta) - y)
    theta = theta - learning_rate * gradients

# Plot the results
plt.scatter(num_hours_studied, exam_score)
plt.plot(num_hours_studied, X.dot(theta), 'r')
plt.xlabel('Number of Hours Studied')
plt.ylabel('Exam Score')
plt.title('Exam Scores vs Number of Hours Studied')
plt.show()

# Report the learned theta
print('Learned theta: ', theta)
```



```
Learned theta:  [[4.23832304]
 [9.40813948]]
```

Step5: Fit a linear regression using Batch Gradient Descent with a different learning rate and plot the results. Report the learned theta. Repeat this step with two different learning rates (0.02 and 0.5). What did you observe?

```
learning_rates = [0.02, 0.5]

for rate in learning_rates:
    theta = np.zeros((2, 1))
    theta, _ = gradient_descent(X_b, exam_score, theta, rate,
num_iterations)
    plt.scatter(num_hours_studied, exam_score)
print('Learned theta (Learning Rate = {}):'.format(rate), theta)
```

- When the learning rate is set to 0.02 the model still converges to a reasonable solution, but the learned theta values are slightly different from the previous step.
- When the learning rate is set to 0.5 the model diverges and the learned theta values become extremely large and negative, indicating that the learning rate is too high.

Step5: Fix the learning rate to 0.1. Then, fit a linear regression using Batch Gradient Descent with number of iterations = 10. Report the learned theta. Repeat this step with two different number of iterations of your choice. What do you observe?

```
learning_rate = 0.1
iterations = [10, 100, 1000]
for num_iter in iterations:
    theta, _ = gradient_descent(X_b, exam_score, theta, learning_rate,
num_iter)
    print('Learned theta (Iterations = {}):'.format(num_iter), theta)
```

- When the number of iterations is set to a very low value (10 in this case), the model does not have enough time to converge to a reasonable solution. The learned theta values are far from the true values,
- When the number of iterations is increased to 100, the model converges to a reasonable solution, and the learned theta values are very close to the values obtained in Step 4