Cairo University
Faculty of Computers and Artificial Intelligence
Machine Learning Course

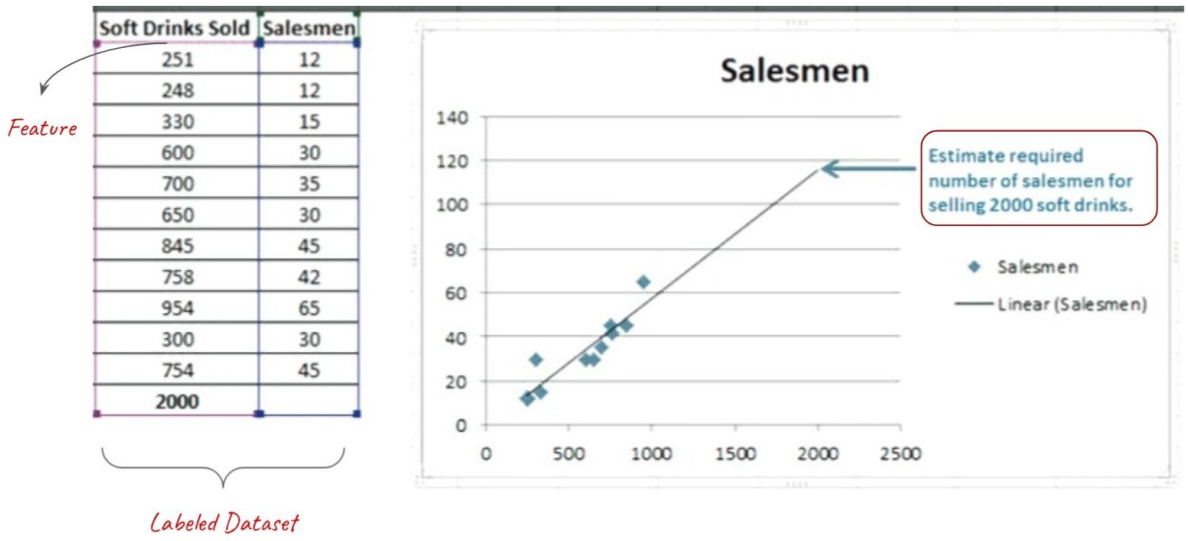
Lab 3: Linear Regression

Lab Outline:

- ➢ Summary of linear regression
- ➢ Steps of applying linear regression to a dataset
- ➢ Pseudocode for gradient descent in linear regression
- ➢ Practice

Summary of Linear Regression:

Linear regression is a ***supervised*** machine learning algorithm. It builds a linear model that can predict the value of a real/continuous target variable based on independent variables (features).



This linear model is in the form of a linear equation ***"hypothesis function"*** as follows:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Parameters: $\theta = \{\theta_0, \theta_1, \theta_2, \ldots . \theta_n\}$

Features: $x = \{x_0, x_1, x_2, \ldots . x_n\}$

Steps of Applying Linear Regression to a Dataset:

1. **Analyze** the dataset
2. Perform data **preprocessing**
3. **Split** the data into train and test sets
4. **Fit** the linear regression model to the training data
5. **Assess** the fitted model on the test data
6. Generate **predictions** for new data

*Note: **Some data preprocessing techniques should be performed after the splitting of data.** So, we can swap the orders of steps 2 and 3, or perform step 3 at a certain point within step 2.*

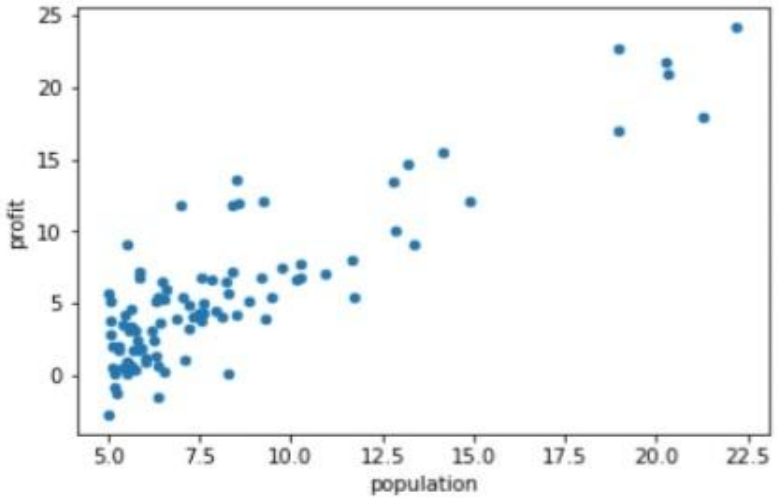
Example:

Suppose you are the CEO of a big company franchise and the company is considering different cities for opening a new store. The chain already has stores in various cities and you have the profits and population data of these cities. You would like to use this data to help you select which new city to expand to next. *How can you do that?*

You can fit a linear regression model to this data to predict the profit based on the population and then select the city which is expected to produce the highest profit.

```
data = pandas.read_csv('data.csv')

# Step 1 (simple analysis through visualizaton)
data.plot(kind='scatter', x='population', y='profit')
plot.show()
```



```
# Step 2
data = shuffle(data)
x = data['population'].to_numpy().reshape((-1,1))
y = data['profit'].to_numpy().reshape((-1,1))

# Step 3 (manual splitting, or we can use train_test_split instead)
x_train = x[0:80]
y_train = y[0:80]
x_test = x[80:]
y_test = y[80:]
```

```
# Step 4
model = linear_model.LinearRegression()
model.fit(x_train, y_train)
```

```
# Step 5
y_pred = model.predict(x_test)
print('Coefficients: \n', model.coef_, " ", model.intercept_)
print('Mean squared error: %.2f'% mean_squared_error(y_test, y_pred))
```

```
Coefficients:

 [[1.21294587]]   [-3.99642098]

Mean squared error: 6.30
```

```
# Step 6
predictions = model.predict(new_data)
```

Pseudocode for Gradient Descent in Linear Regression:

In order to implement the "model fitting" step, we will use the optimization algorithm ***"Gradient Descent"*** to learn the model parameters as follows:

Function fit_GD

Input: *X, y*
Output: *theta*

1: **Initialize** *theta*
2: **Set** *alpha* **and** *max_iterations*
3: **for** *i=0* **to** *max_iterations*
4: **Compute** the hypothesis function values according to the equation: $h=theta^T*X$
5: **for** *j=0* **to** *theta.length*
6: **Compute** the partial derivative of the error: $part\_der=(1/m)*sum((h-y)*X_j)$
7: **Update** $theta_j$ according to the equation: $theta_j=theta_j-alpha*part\_der$
8: **end**
9: **end**

Practice:

Given "data.csv", "new_data.csv" and the theta values [-3.63, 1.16]:

- ➢ Implement the **"predict"** function.
- ➢ Implement the **"mean_squared_error"** function.
- ➢ Test your functions.