# Fourth Industrial Summer School

## Module 4: ML

# Supervised Learning: Regression

Friday, July 5, 2024

# Session Objectives

- ✓Introduction
- ✓Regression
- ✓Evaluation
- ✓Higher order Regression
- ✓Regularization

# Introduction

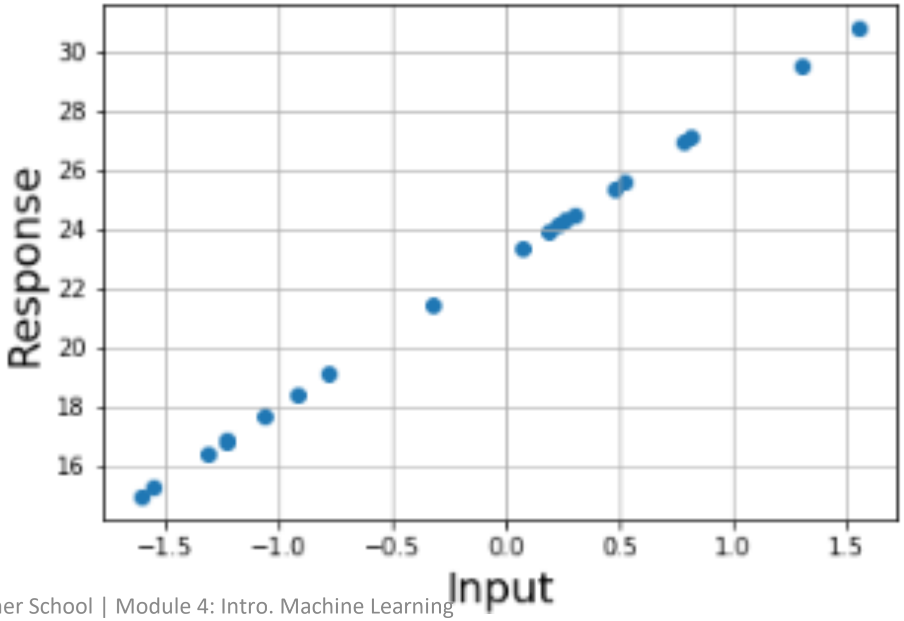
- Supervised techniques are a set of learning techniques where the '**right answer**' for each datapoint exists at the learning stage.

- There are two types of supervised learning
  - **Regression**: the right answers are in the form of continues real values
  - **Classification**: the right answers are in the form of finite integer values.

# Regression

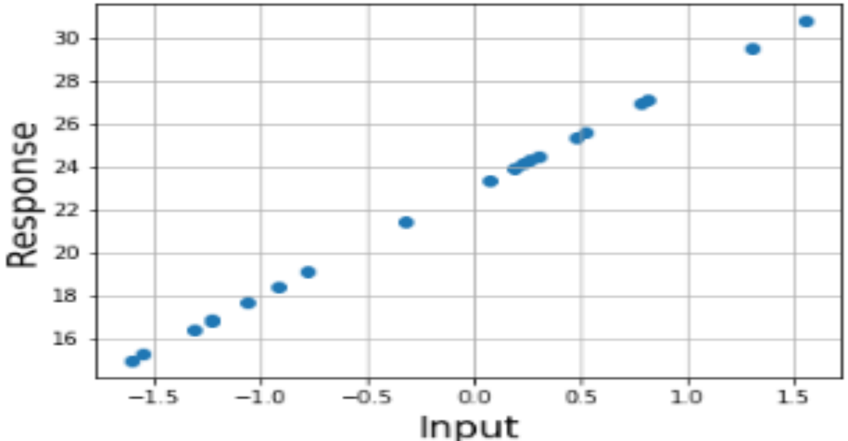
- Regression is a statistical technique that is used widely to predict a continuous future output.
- It models a relationship between two sets of variables

$$x\text{: (Input)}, \quad y\text{: (Response)}.$$



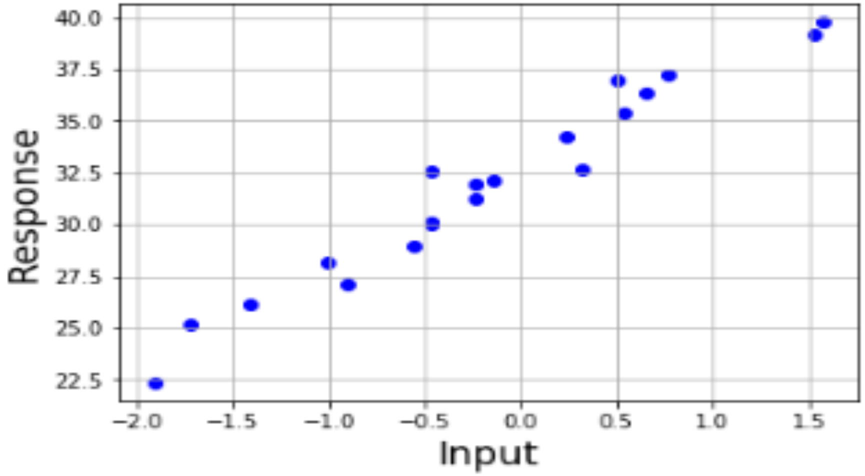
# Regression data

- Data: $(X_i, Y_i)\ for\ i = 1, \ldots, n$




$y = \beta_0 + \beta_1 X$

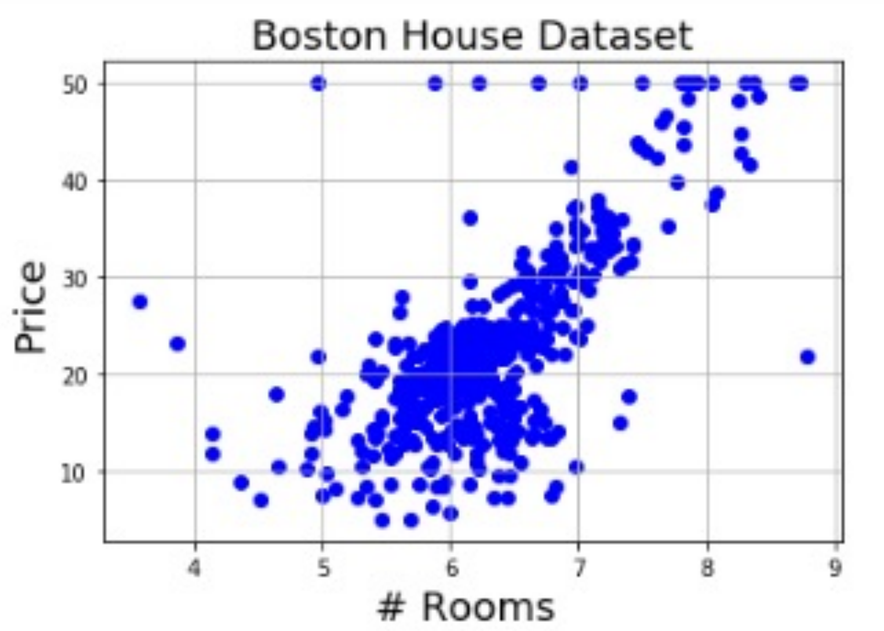
- The data is often not clean




$y = \beta_0 + \beta_1 X + \epsilon$

# Example:

- We have an independent variable ( number of rooms per house),
- and a history of a selling prices of houses




- Suppose a friend asks you to estimate the price of his (6 room) house. What will be your answer ?

# Linear Regression Modeling

- Given

$$x: (\text{Feature/s}),\ \ y: (\text{Outcome}).$$

- Mean of Y is a **straight-line function of *X***, plus an error term (i.e., residual)
- The goal is to find the best fit line that minimizes the sum of the error terms

# Contd.

- **Regression**: the mean of a response variable as a function of one or more explanatory variables:

$$\mu\{Y \mid X\}$$

- A Simple linear regression model:

$$\mu\{Y \mid X\} = \beta_0 + \beta_1 X$$

  - $\mu\{Y \mid X\}$: "mean of Y given X" or "regression of Y on X"
  - $\beta_0$: The intercept
  - $\beta_1$ : The slope
  - $X$: Independent variable

# Regression Procedure

- A fitter value for sample $x_i$ is its estimated mean:
$$\hat{y}_i = \mu\{y_i | x_i\} = \beta_0 + \beta_1 x_i$$

- Residual (Error) for observation $x_i$:
$$E_i = y_i - \hat{y}_i$$

- The objective is to make the residual $E_i$ as small as possible:
$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

# Least Square Procedure

- Expand the Residual : $\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$
- Compute the partial derivative of the objective function LS, and equate the result to zero, we get these two equations

1. $\sum_i \beta_0 + \sum_i \beta_1 x_i = \sum_i y_i$
2. $\sum_i \beta_0\, x_i + \sum_i \beta_1 x_i^2 = \sum_i x_i y_i$

- It can be written as

$n.\beta_0 + \beta_1 \sum_i x_i = \sum_i y_i\,,$

$\beta_0 \sum_i x_i + \beta_1 \sum_i x_i^2 = \sum_i x_i y_i$

It will be clearer if we write them in matrix form:

$$\begin{bmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{bmatrix}$$

# Contd.

$$\begin{bmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{bmatrix}$$

By solving this system of liner equations:

$$\beta_1 = \frac{(n \sum_i x_i y_i - \sum_i x_i \sum_i y_i)}{n \sum_i x_i^2 - \left(\sum_i x_i\right)^2}$$

$$\beta_0 = \frac{1}{n} \sum_i y_i - \frac{\beta_1}{n} \sum_i x_i$$

# Regression Model Parameters: Python Warm-up

- **Practice 1**
  1. Generate random data for X, y (Follow steps in the Notebook)
  2. Develop an estimate_coef () function to return the model coefficients (the structure is given below).
  3. Try the model parameters to predict y_hat for each X value.
  4. You may plot your model as a line over a scatter plot the dataset (X,y)

```python
# estimator
def estimate_coef(X,y):

  #get size of samples
  n = np.size(X) # or any method

  #compute the coeficients
  b1 = # complete this line by coding b1 equation from slides
  b0 = # complete this line by coding b0 equation from slides

  # return learned model paramters
  return (b0, b1)
```

# Regression model using Sklearn: Practice 2

- Sklearn provides a linear regression method under the **linear_model package**
- The steps that can be followed to build a linear regression model:
  1. **Import** the linear regression method
  2. **Initialize** the linear regression method (set all possible parameters at this step)
  3. **Fit (or train)** the linear regression method by passing data, and corresponding responses (**X, y**)

Now you ready to use your trained model! It can be used to predict new responses **y_pred** for each new input **X**

```python
from sklearn.linear_model import LinearRegression

#initialize the model
Lr = LinearRegression()

# train the model
Lr.fit(X, y)

# make prediction
y_pred = Lr.predict (X)
```

# Sklearn Lr model attributes

- Once the model has been trained, we can extract several information from the model:
  - The model estimated parameters $\beta_1$
  - The model estimated intercept $\beta_0$
  - Etc.

```
print('model paramter b1:', Lr.coef_[0])
print('model parameter b0:', Lr.intercept_)
print(f'model train with ({Lr.n_features_in_}) features')
```

```
model paramter b1: 8.154783375959148
model parameter b0: 0.5674843974430133
model train with (1) features
```

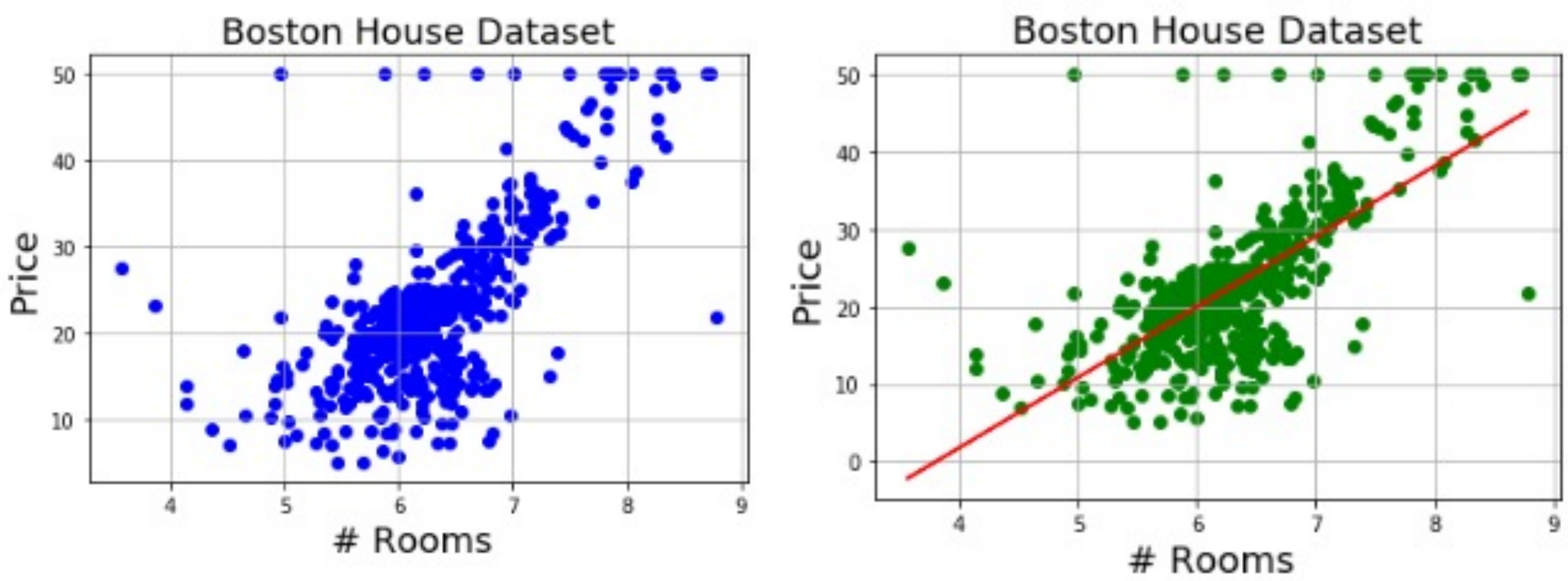
# Exercises (1-3)

- Build simple regression
- Explore and practice Sklearn Regression modules
- Load Boston dataset and answer the slides question

# House-Price Question!

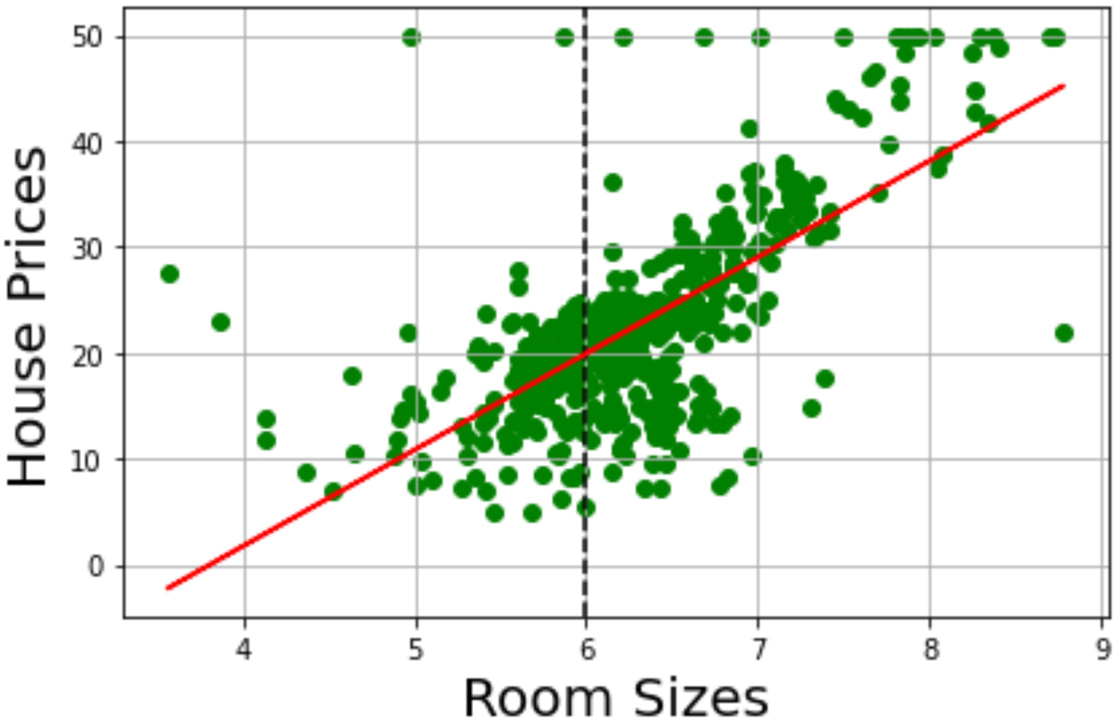
- So, what is your answer to our first question:

  What is your estimate price for your friend's house?.



# Return to our House-Price Question!

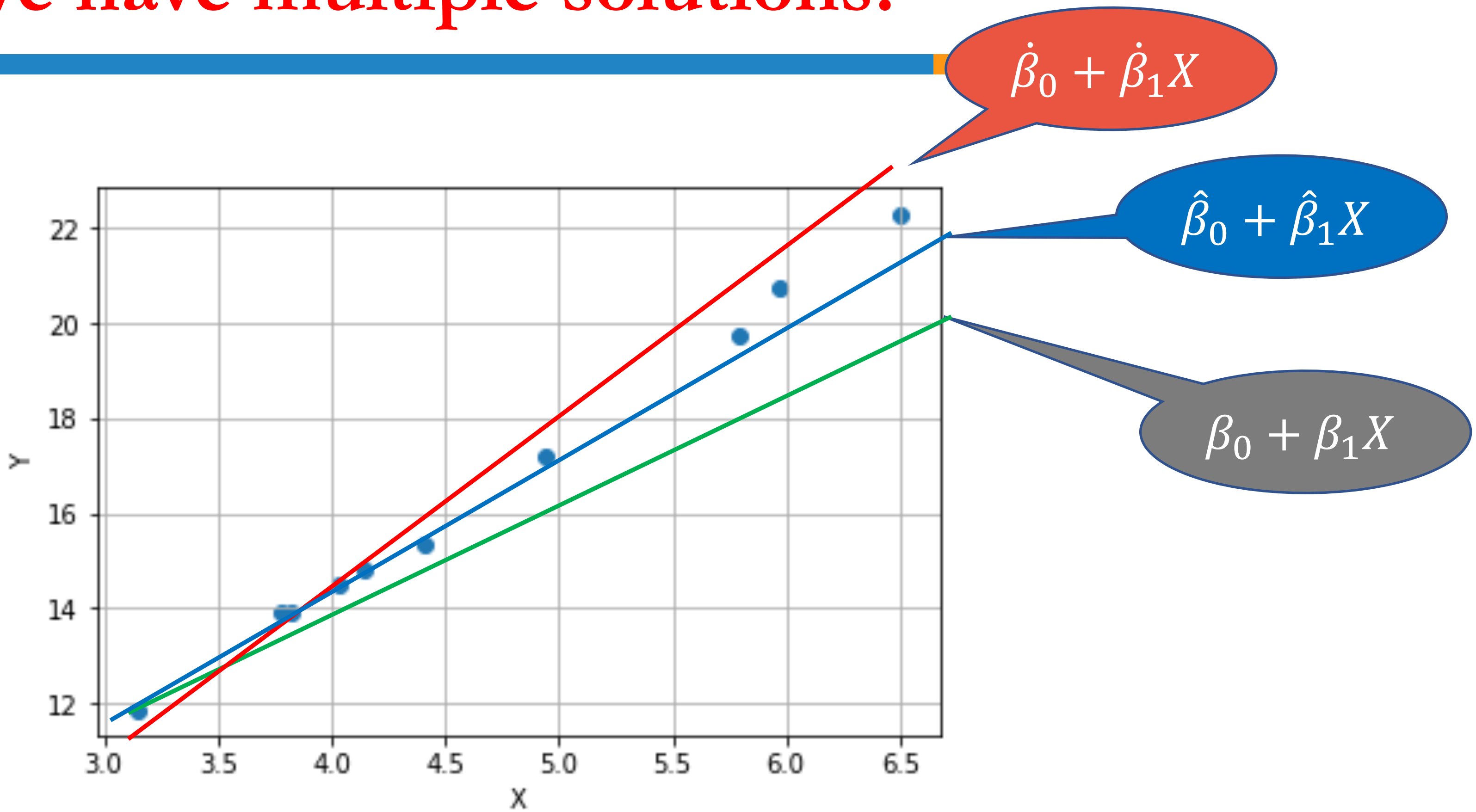
- So, what is the answer to our first question: $20K




```
print("Your House with 6 Rooms Can be Sold with: $%0.2fK"% (b0 + b1* 6) )
```

```
Your House with 6 Rooms Can be Sold with: $19.94K
```

# Can we have multiple solutions?




- We could end up with different regression models based on the fitted sampled datasets.

# Multiple Regression

- A generalization to a single variable case is expanded by considering multiple independent variables and one depended variable.
- The data is in the matrix form: X = [$X_1$, $X_2$ , …, $X_n$] and one response $y$ value

$$y = b_0 x_0 + \boldsymbol{\beta_i x_i}, \qquad i = 1, 2, 3, 4 \dots n$$

- In multiple features case, $\beta$ contains the slopes along each feature $X$.
- Alternatively, we can think of it as a weighted sum of all input features, with weights that can be negative given by the entries of $\beta$.

# Multiple Regression

$$y = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon$$

$$where\ x_{i0} = 1$$

- The multiple regression model is based on the following assumptions:
  - There is a linear relationship between the dependent variable and the independent variables.
  - The independent variables are not highly **correlated** with each other.
  - Observations are selected independently and randomly from the population.
  - The phenomenon error assumed to be normally distributed with a mean of 0 and some variance $\sigma$.

# Multiple Regression Interpretation

- Suppose we learned a regression model for some industrial product (e.g., a price of mobile X). The model then gives the following information:

$$Price_{Mobile} = 85 - 0.8\ IR + 3.5\ OP + 0.5SS + 2SP$$

where:

- IR: Interest rate
- OP: Oil Price
- SS: Screen Size
- SP: Speakers

The question here what information you can get from the above regression model ?

# Summary

- Linear regression models the responses given in the dependent variable over a change in some independent variable.
- A more realistic situation, a dependent variable is usually explained by multiple independent variable.
- Multi-regression assuming that there is a linear relationship between both the dependent and each independent variables

# Sklearn: Regression notes

- Sklearn provides a method named ***make_regression*** to generate regression datasets.
- We can control the number of features which can be important (informative) in relation with the response variable.
- To generate the data, we can follow the steps below

```
from sklearn.datasets import make_regression

X, y = make_regression(n_features = 5, n_informative = 3)

# instantiated previously
Lr = LinearRegression()
# fit (train)
Lr.fit(X, y)

# Predict (test)
y_pred = Lr.predict(X)
```

- When the target dataset is huge, the closed form is not feasible!
- Sklearn provides another version called ***SGDRegressor*** that uses an optimization algorithm to estimate the regression parameters

# Exercises (4-5)

- Multiple regression: Identify which features less important
- Multiple regression: Build simple regression model using Boston dataset