Python and Data science
Zaouche-Dahmani Djaouida

Data science

- Data science is the process of deriving knowledge and insights from a huge and diverse set of data through organizing, processing and analyzing the data.

- It involves many different disciplines like mathematical and statistical modelling, extracting data from it source and applying data visualization techniques.

- Often it also involves handling big data technologies to gather both structured and unstructured data.

Data science

- Financial Risk management
- Improvement in Health Care services
- Computer Vision
- Efficient Management of Energy

Data science

- Empirical approach based on data to provide answers to problems
- A data is observation's results of a population or a sample
- A data is a number, or a characteristic that brings information about an individual, an object or an observation

```
Example:
Jean Michel: I am 10 years old
             I have blue eyes
```

Notion of variables

Relation between a variable and data:
The number / characteristic varies with the individuals / objects

Notations:

- Variable X_j
- for individuals/objects/ observation i: X_{ij}.

Examples:

Variable X_{age} for individual 1,2, 3, …. : X_{1age}, X_{2age}, X_{3age},……

Variable X_{salary} for individual 1,2, 3, …. : $X_{1salary}$, $X_{2salary}$, $X_{3salary}$,……

Kind of variables

- Quantitative data
  - number
  - How many?
  - Computation possible (mean, …)
  - Comparison possible (=,>,<,….)
    * Continue in R
    * Discrete: limited number of possible values

Example:
Jean Michel: I am 10 years old

Kind of variables

- Qualitative data
  - Quality or characteristics
  - corresponds to "category"
    * Nominal (categorial)

eye color comparison
(equality / difference)
* Ordinal has order

(degree) a test of opinion etc.)

Example:
Jean Michel:I have blue eyes 5
I got ***A*** for my french test and ***E*** for English test

Linear Regression Model

What does Linear Regression mean?

- ✓ Linear regression is a kind of statistical analysis that attempts to show a relationship between two or more variables.

- ✓ Linear regression can create a predictive model on apparently random data, showing trends in data, such as in cancer diagnoses or in stock prices.

Simple Linear Regression Model

Monthly income	Average spending for food
1000 euros	200 euros
1500 euros	300 euros
…..	…
…..	….

x_i, i=1,….,m

y_i, i=1…,m

Objective: predict the average spending for food according to the incomes of the households

The set $\{(x_i, y_i) | 1 \leq i \leq m\}$ is called *learning data*

Simple Linear Regression Model

Solution:

- Find a linear function $h(x)=\theta_0+\theta_1 x$, with x for monthly income (IC) and $h(x)$ for average spending (AS). This function predicts the value of AS for any IC.

- Function h is called hypothesis function.

Simple Regression Model

- Problematic:
  - Find the best hypothesis function, called h, to approximate the output values Y according to a learning data:

Hypothesis h

input X ---------- > output Y

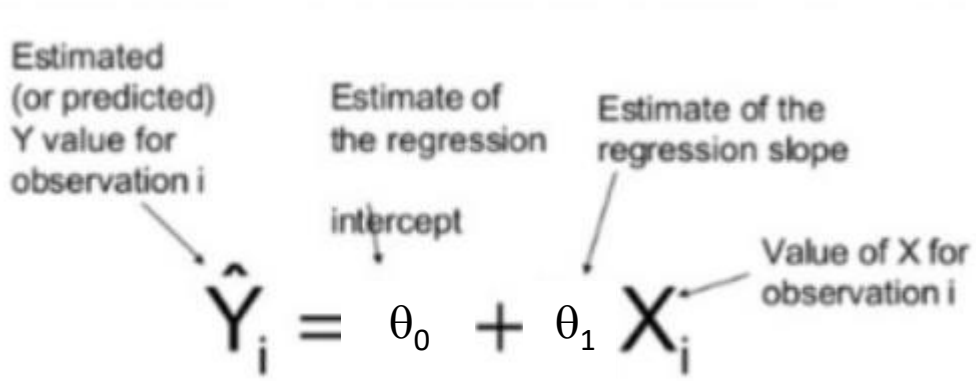
- Simple regression:

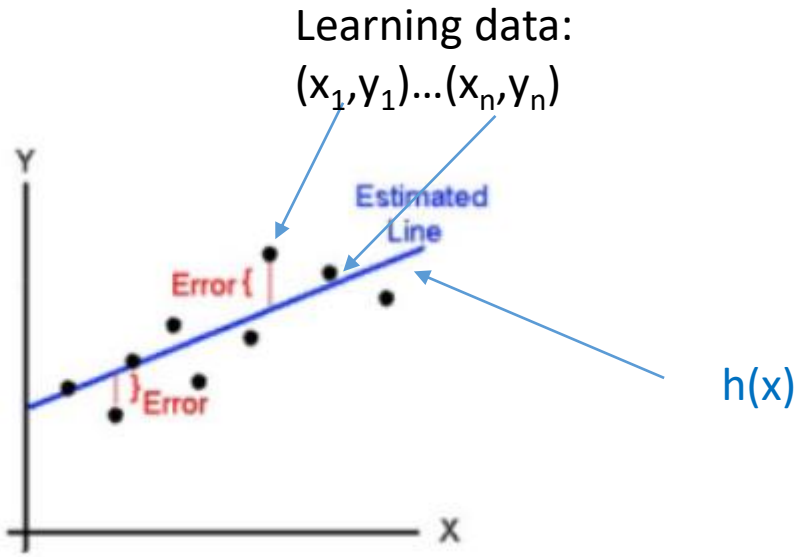
$$h(X) = \theta_0 + \theta_1 X$$

Simple Regression Model

- Consider m individuals $(x_1, y_1) \ldots (x_m, y_m)$
- Find the best couple (θ_0, θ_1) such that the set of $h(x_i)$ is as close as possible to the set of y_i, i in $[1..m]$
- y_i is the real value associated with x_i and $h(x_i)$ its predict value
- Observe that when x is not in the learning data, it has only its predict value, that means $h(x)$

Simple Linear Regression





Based on the learning data, find the best blue line to find their best approximation

Simple Linear Regression

Définition de la fonction hypothèse

$$\text{valeur d'entrée } x \xrightarrow{\text{hypothèse } h} \text{valeur de sortie } y$$

Dans le cas de la régression linéaire univariée :

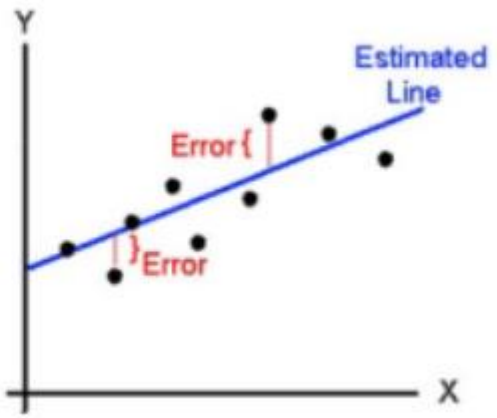
$$h(X) = \theta_0 + \theta_1 X$$

x is an input of *h*

y is an output of *h*

- *h* is a linear function
- *h* is a hypothesis function

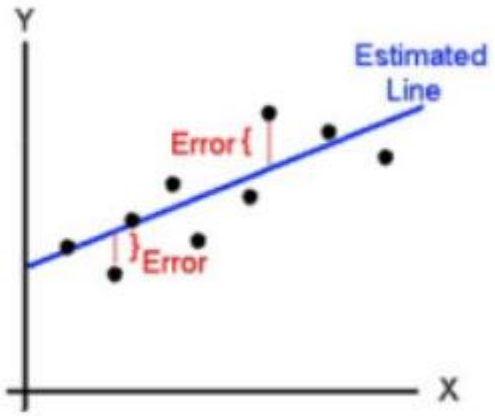
Simple Linear Regression



- The blue line represents the best approximation to the cloud of black dots. This approximation is made possible by calculating the predictive parameters θ_0 and θ_1 that define the blue line.

- The question is: how do we calculate the values of θ_0 and θ_1?

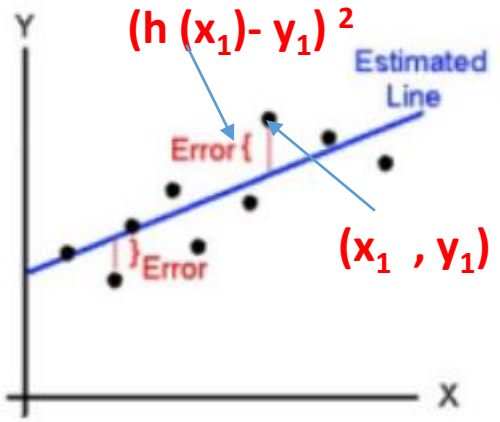
The cost function



- The figure shows that the blue line tries to approach as many points as possible (by reducing the gap with them).
- It minimizes the overall error as much as possible.
- The goal of the game is to find an optimal pair (θ_0, θ_1) such that h (x) is as close as possible to y (the value we are trying to predict). And this, for all the couples (x, y) that make up our set of **learning data.**

The cost function

Error for point (x_i , y_i)



- Sum of all dots's errors
- Consider all the dots of the learning data

We define "the unit error" between an observed value y_i and a predicted value $h(x_i)$, as follows:

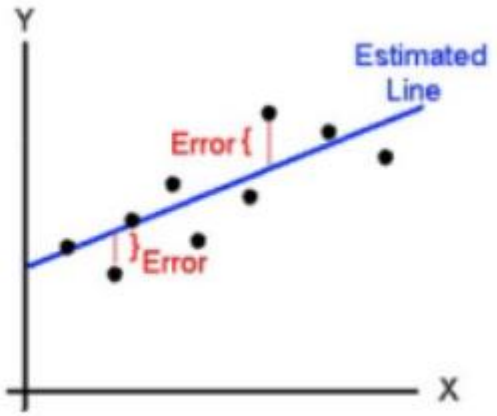
$$\mathbf{(h(x_i)- y_i)^2}$$

Find the best pair (θ_0, θ_1) to minimize the overall cost of unit errors, which is defined as follows:

$$\sum_{k=1}^{m}(h(x_i) - y_i)^2$$

where m is the size of the training set

The cost function



The cost function is defined as follows:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\theta_0+\theta_1 x_i - y_i)^2$$

Finding the best parameters θ_0, θ_1 consists to minimize (find the minimum) the cost function.

Gradient descent algorithm

Begin

Assign random values for θ_0 and θ_1

Repeat

$$\theta_j \leftarrow \theta_j - \alpha * \frac{d}{d\theta_j} J(\theta_0, \theta_1) \quad \text{for j} \quad \text{in } \{0,1\}$$

Until convergence towards the miminum of the cost function

Return (θ_0, θ_1)

End

Multiple Linear Regression

- Like for the simple linear regression, we aim to find the best hypothesis h to approximate learning data
- Instead of using a single variable X as input, we offer more variables as input to make better approximation of the leaning data
- For instance, to approximate the prices of houses, we consider the number of rooms, the size, the town, and the street

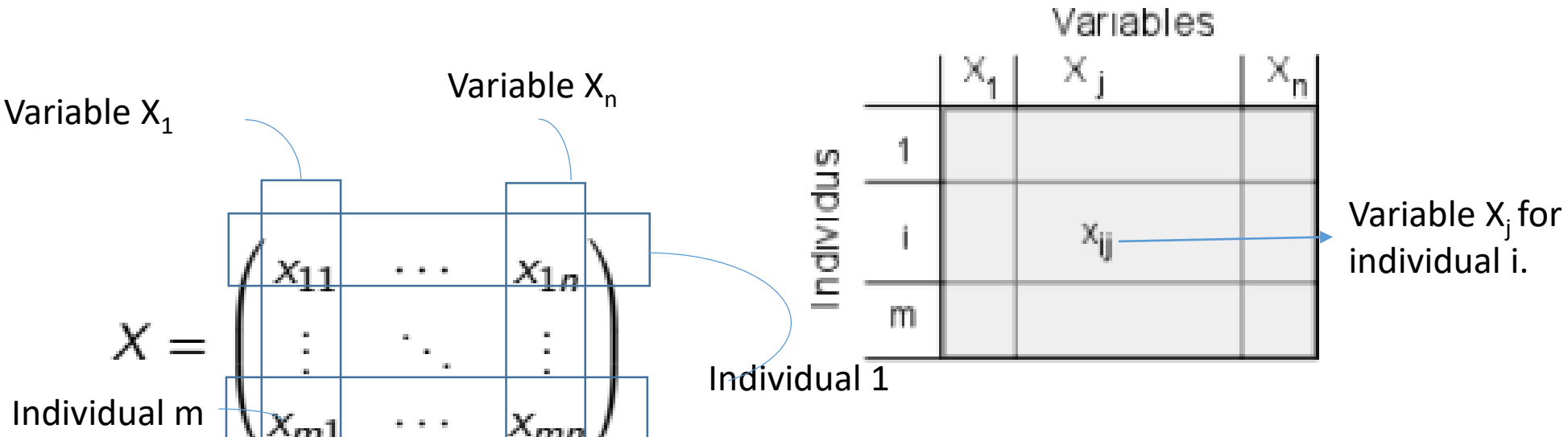
- Hypothesis function h

$$h(X)=\theta_0 + \theta_1 X_1+ \theta_2 X_2+ \ldots+ \theta_n X_n$$

Multiple variables $X_1, X_2, \ldots, X_j$ from 1 to n

can describe a same individual/ object/observation

Big number of individuals from 1 to m



Multiple Linear Regression

Definition of the hypothesis function h

hypothesis function h

value of input X $\longrightarrow$ output value Y

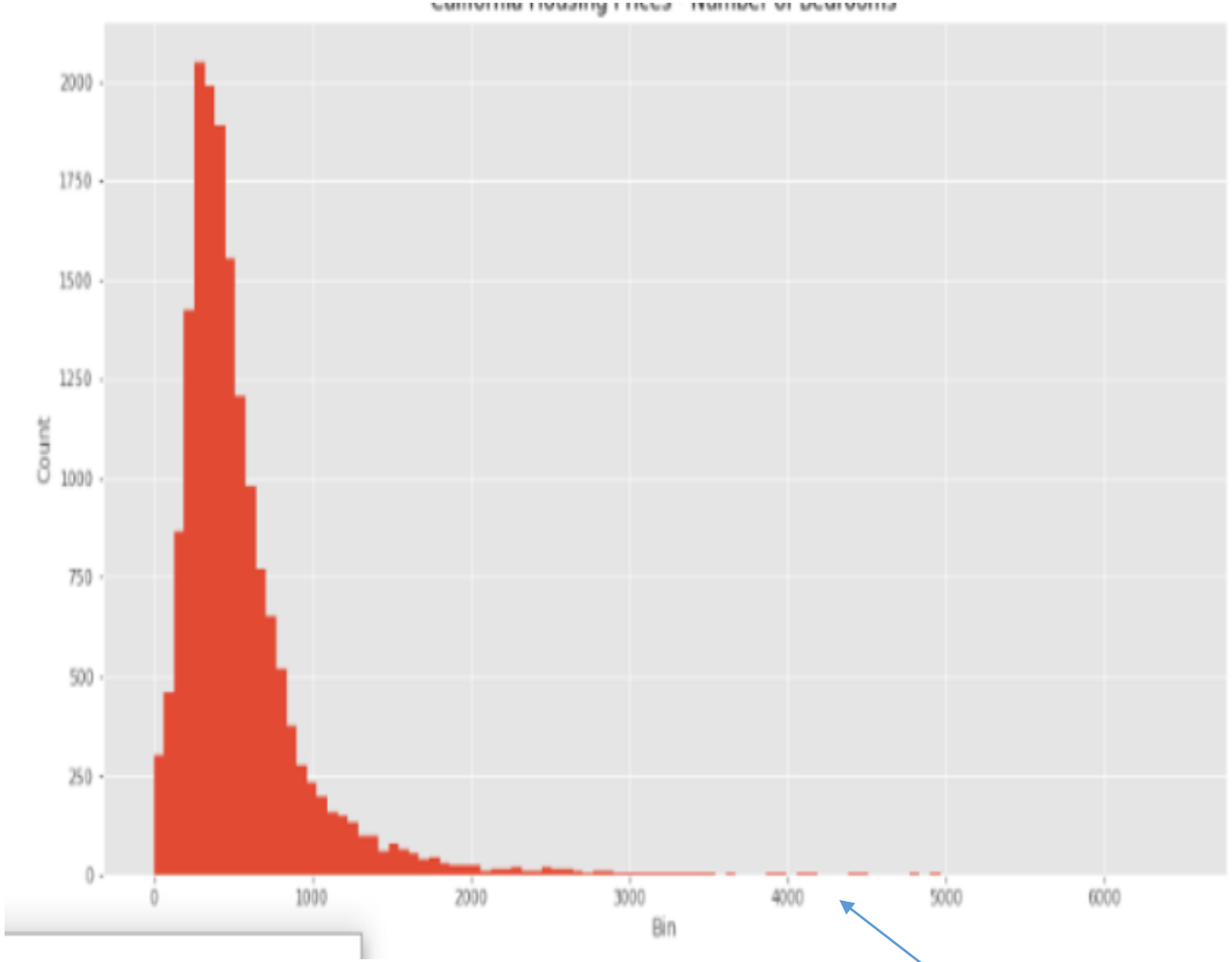
Multiple linear regression

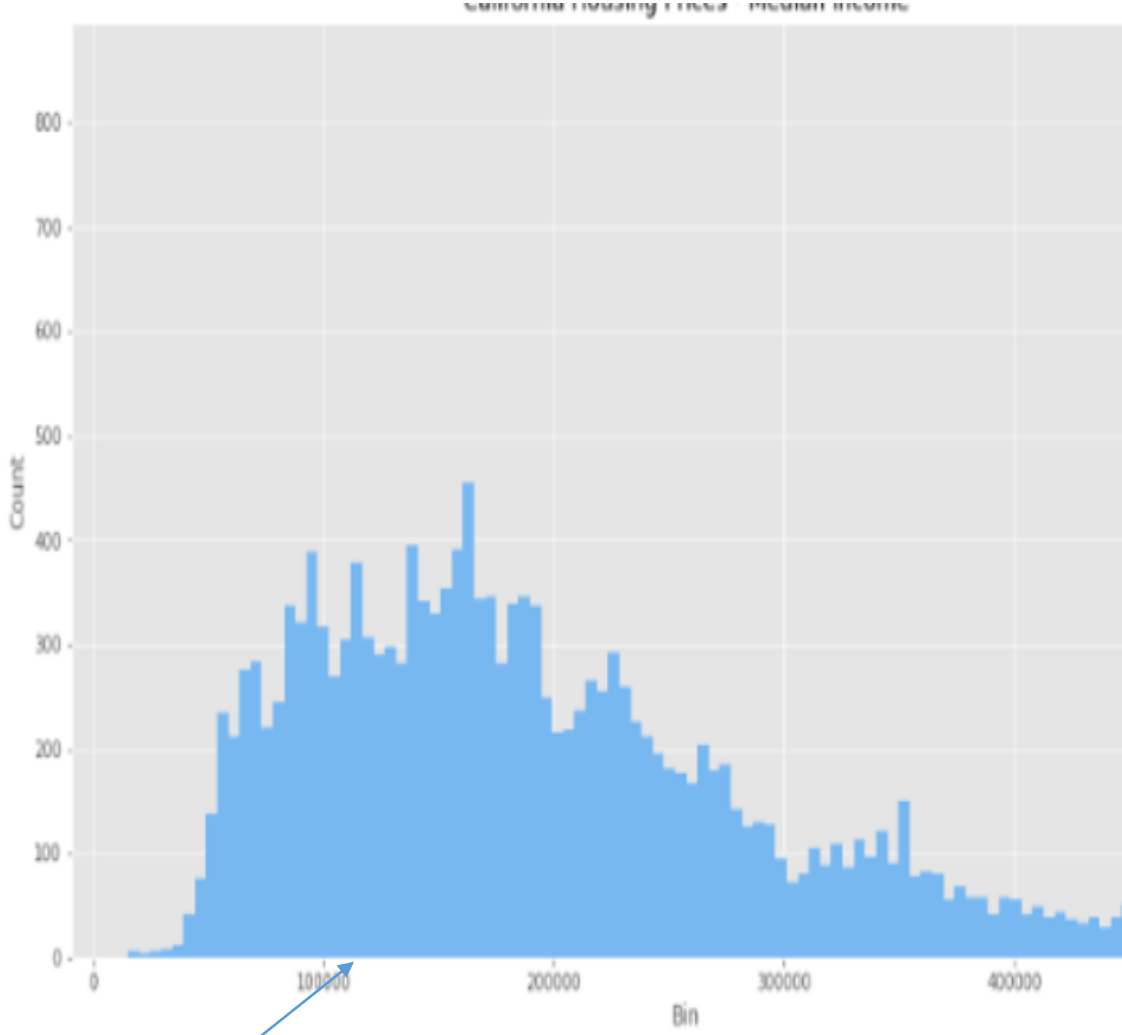
$$h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_n X_n$$

Cost function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Multiple Linear Regression





Different scales for data

Normalization/ standardization of data

- Some data analysis algorithms (linear regression, linear discriminant analysis (LDA) and Gaussian naive Bayes) will achieve better performance if your data has a consistent scale or distribution.

- There are two main techniques to consistently rescales data:
  - normalization
  - standardization.

Normalization of data

- Scaling just changes the range of data. Normalization is a more radical transformation. The point of normalization is to change your observations

- **Normalization of ratings** means adjusting values measured on different scales to a common scale

Normalization of data

A Min-Max scaling is typically done via the following equation:

$$X_{norm} = \frac{X - Xmin}{X_{max} - Xmin}$$

Normalization of data

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
data_1=np.array([1,567,890,45,456,678,333,444,789,1000])
data_2=np.array([1,5,8,4,6,8,3,4,7,1])
values_1 = pd.DataFrame(data_1)
values_2 = pd.DataFrame(data_2)
scaler = MinMaxScaler(feature_range=(2, 20))
scaler = scaler.fit(values_1)
print('_____Min and Max _____')
print(scaler.data_min_, scaler.data_max_)
print('_____Normalized values _____')
normalized = scaler.transform(values_1)
for i in range(9):
    print(normalized[i])
print('_____Initial values _____')

inversed = scaler.inverse_transform(normalized)
for i in range(9):
    print(inversed[i])
```

```
print('_____ ¨Print for
value_2_____')
scaler = scaler.fit(values_2)
print('_____Min and Max _____')
print(scaler.data_min_, scaler.data_max_)
print('_____Normalized values
_____')
normalized = scaler.transform(values_2)
for i in range(9):
    print(normalized[i])
print('_____Initial values _____')

inversed = scaler.inverse_transform(normalized)
for i in range(9):
    print(inversed[i])
```

Normalization of data

```
Give a python implementation (without using any normalization package ) to normalize
data_1 and data_2.
```

Standardisation of data

- Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

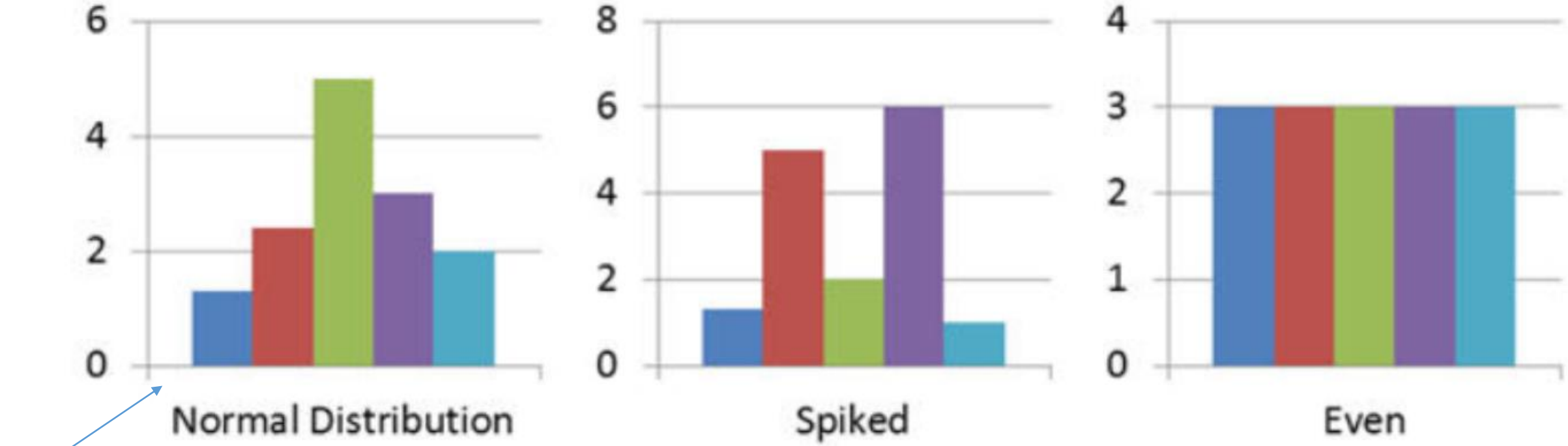
- Like normalization, standardization can be useful, and even required in some machine learning algorithms when data has input values with differing scales

- Standardization assumes that the observations fit a Gaussian distribution with a well behaved mean and standard deviation. If data are standardized when this expectation is not met, the results may not be reliable

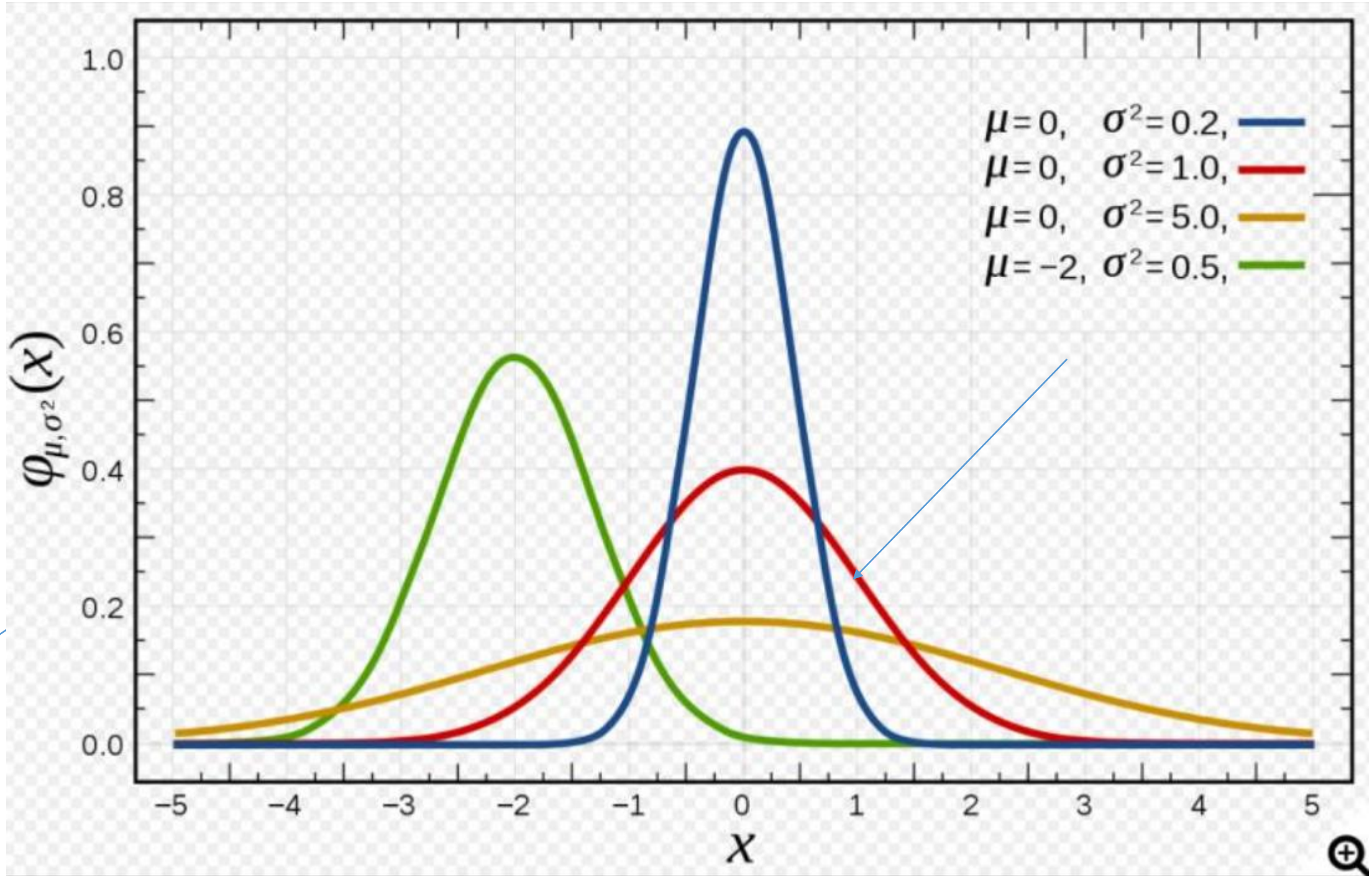
Standardisation of data

- Statisticians refer to the normal curve as the Gaussian Probability distribution, named after Gauss. The Normal distribution is used to analyze data when there is an equally likely chance of being above or below the mean for continuous data whose histogram fits a bell curve

Standardisation of data



Histogram shapes


1.0
0.8
0.6
0.4
0.2
0.0
$\varphi_{\mu,\sigma^2}(x)$
$\mu = 0, \quad \sigma^2 = 0.2,$
$\mu = 0, \quad \sigma^2 = 1.0,$
$\mu = 0, \quad \sigma^2 = 5.0,$
$\mu = -2, \quad \sigma^2 = 0.5,$
−5
−4
−3
−2
−1
0
1
2
3
4
5
x

Standardisation of data

- A value is standardized as follows:

$$x' = (x - mean) \,/\, standard\_deviation$$

Where:

- the mean is calculated as:

$$mean = sum(x) \,/\, count(x)$$

- and the *standard_deviation* is calculated as:

$$standard\_deviation = sqrt(\, sum(\, (x - mean)2 \,) \,/\, count(x))$$

```
from sklearn.preprocessing import StandardScaler
from math import sqrt
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data = np.random.randn(10)
series = pd.Series(data)
print('_____
__')
values = series.values
values = values.reshape((len(values), 1))
plt.hist(values)
plt.show()
print('_____
__')
# train the standardization
scaler = StandardScaler()
scaler = scaler.fit(values)
print('Mean: %f, StandardDeviation: %f' % (scaler.mean_,
sqrt(scaler.var_)))
```

```
# standardization the dataset and print the first 5 rows
normalized = scaler.transform(values)
plt.hist(normalized)
plt.show()
scaler = scaler.fit(normalized)
print('Mean: %f, StandardDeviation: %f' %
(scaler.mean_, sqrt(scaler.var_)))

print('_____PRINT
VALUES_____',)

for i in range(len(data)):
    print(normalized[i])
#inverse transform and print the first 5 rows

scaler = scaler.fit(values)
inversed = scaler.inverse_transform(normalized)
for i in range(len(data)):
   print(inversed[i],'***',values[i])
```

Standardization or normalization?

- There is no obvious answer to this question: it really depends on the application.
- For example, in clustering analyses, standardization may be especially crucial in order to compare similarities between features based on certain distance measures
- A popular application of normalization is image processing, where pixel intensities have to be normalized to fit within a certain range (i.e., 0 to 255 for the RGB color range). Also, typical neural network algorithm require data that on a 0-1 scale.