



Minia University

Faculty of Computers & information

# Artificial Neural Networks and Deep Learning

Slides By:



T.A. Sarah Osama Talaat



**E-mail:** SarahOsama.fci@gmail.com

Slides were prepared based on set of references mentioned in the last slide



Lectures, FCI, Mina University

# Agenda

- ❑ Revision
- ❑ Data Preprocessing.
  - ❑ Data cleaning
    - Dealing with missing data.
    - Dealing with categorical data.
  - ❑ Feature scaling
    - Rescaling
    - Mean normalization
    - Standardization
    - Scaling to unit length
  - ❑ Dimensionality reduction
    - Feature selection
    - Feature extraction
  - ❑ Partitioning a dataset in training and testing sets



# Let's Start



In machine learning, pattern recognition and learning there is a relationship/nature between the observations (input features) and response (target), we need to understand this relationship .

## ❑ Learning

- The learning process tends to understand and predict new values based on finding a mathematical model and a relationship between the given observations ( i.e. inputs and output).

## ❑ Examples of learning problems:

- ❑ Predict a wind speed and a solar radiation.
- ❑ Detection of cancers; Brest, Leukemia and Spinal cancers.
- ❑ Estimate the amount of glucose in the blood.



# Data Preprocessing

## Introduction

The quality of the data and the amount of useful information that it contains are key factors that determine **how well a machine learning algorithm can learn**. Therefore, it is absolutely critical that we make sure to examine and preprocess a dataset before we feed it to a learning algorithm. In this lecture, we will discuss the essential data preprocessing techniques that will help us to build good machine learning models.

The topics that we will cover in this issues/lecture are as follows:

- Data Cleaning
- Feature Scaling
- Dimensionality Reduction



- ❑ **Data Preprocessing technique** is used to **manipulate** and **transform** the raw dataset into a **clean and scaled** dataset.
- ❑ In other words, whenever the dataset is **gathered from different sources** it is **collected in raw format** which is **not feasible** for the analysis.
- ❑ Therefore, certain steps are executed to convert the dataset into a small clean dataset. The **data preprocessing** technique is performed **before** the uses of the collected dataset.



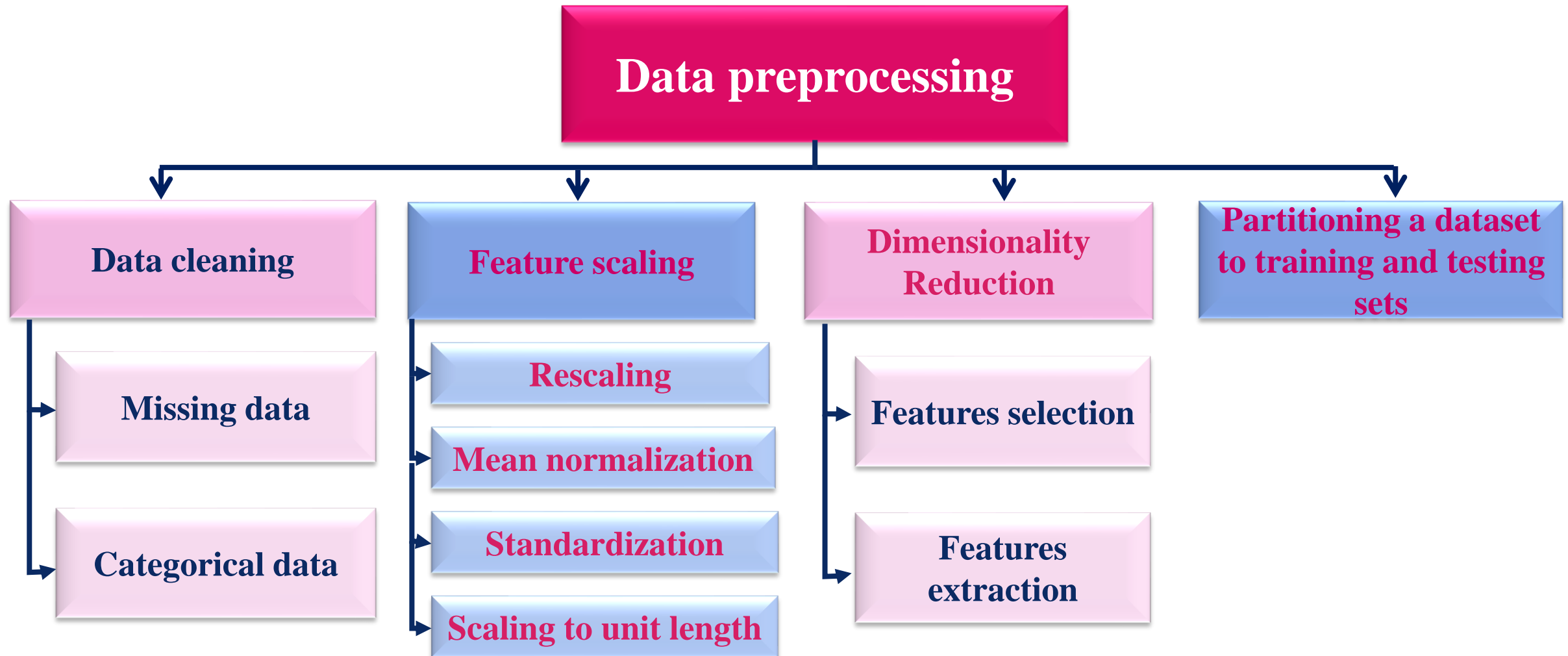
- The main objectives of data preprocessing are to **manipulate** and **transform** raw data into **cleaned** and **scaled** format.
- In addition it is important to **compress** the data onto a **smaller dimensional** subspace while retaining most of the **relevant** information.





# Data Preprocessing

## Types of data preprocessing





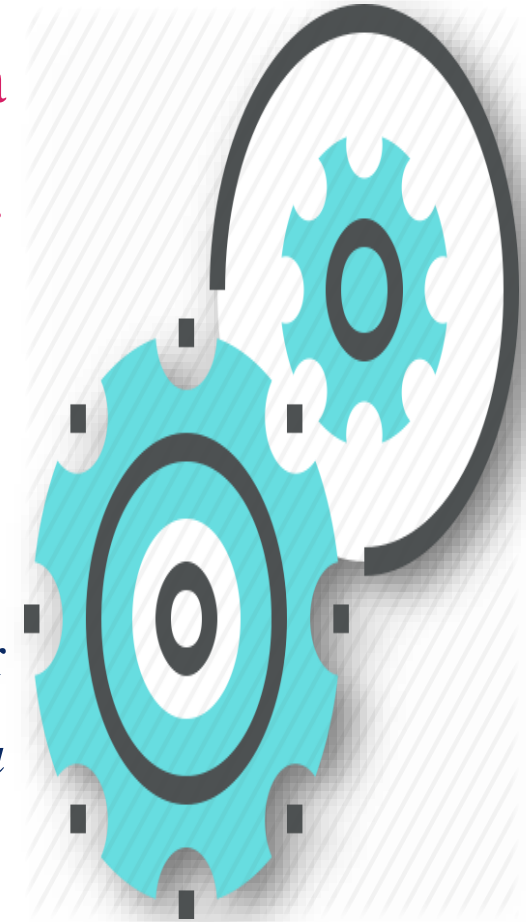
# Data Preprocessing

## Data cleaning: missing data

❑ In real-world applications are **familiar** that the collected data samples contain one or more missing values for various reasons.

These reasons include:

- There could have been an error in the data collection process,
- certain measurements are not applicable and
- particular fields could have been simply left blank in a survey, for instance. We typically see *missing values as the blank spaces in our data table or as placeholder strings such as NaN* (Not A Number).



# Data Preprocessing

## Data cleaning: missing data

Artificial Neural Networks  
&  
Deep Learning

### Solutions to avoid the problem of missing data



```
graph TD; A[Solutions to avoid the problem of missing data] --> B[Eliminate/Remove]; A --> C[Mean]; A --> D[Median]; A --> E[Most frequent value];
```

**Eliminate/  
Remove**

**Mean**

**Median**

**Most frequent  
value**

□ There are several available solutions to avoid the problem of missing data :

- **Eliminate/Remove:** you can simply eliminate/remove the corresponding features (columns) or samples (rows) that contain missing data from the used dataset (BUT it's quite dangerous, because we might lose too much valuable data).
- **Mean/Median/Most frequent value:** you can take the mean/median/most frequent value from the column/row that contains missing data.

# Data Preprocessing

## Data cleaning: missing data

❑ **Mean** is used to replace the missing values using the **mean/average** along the axis (i.e. column or row).

- Note that the mean isn't a value from the original list. This is a common result. You should not assume that your mean will be one of your original numbers.

$$\text{mean} = \frac{\sum x}{n},$$

where **x** represents the existing(i.e. not missing) feature values and **n** represent the number of instance

❑ **Most frequent value** is used to replace missing values using the most frequent value along the axis.

# Data Preprocessing

## Data cleaning: missing data

❑ **Median** is used to replace missing values using the median (middle value) along the axis. The median is the middle value in the sorted list, so when the list is not sorted we rewrite the list in order.

- When the number of data points is odd, the middle data point is returned.

$$\text{median} = \text{List} \left[ \frac{n + 1}{2} \right],$$

where **n** represents the number of existing (i.e. not missing) instance.

- When the number of data points is even, the median is interpolated by taking the average of the two middle values.

$$\text{median} = \frac{\text{List} \left[ \frac{n}{2} \right] + \text{List} \left[ \frac{n + 1}{2} \right]}{2},$$

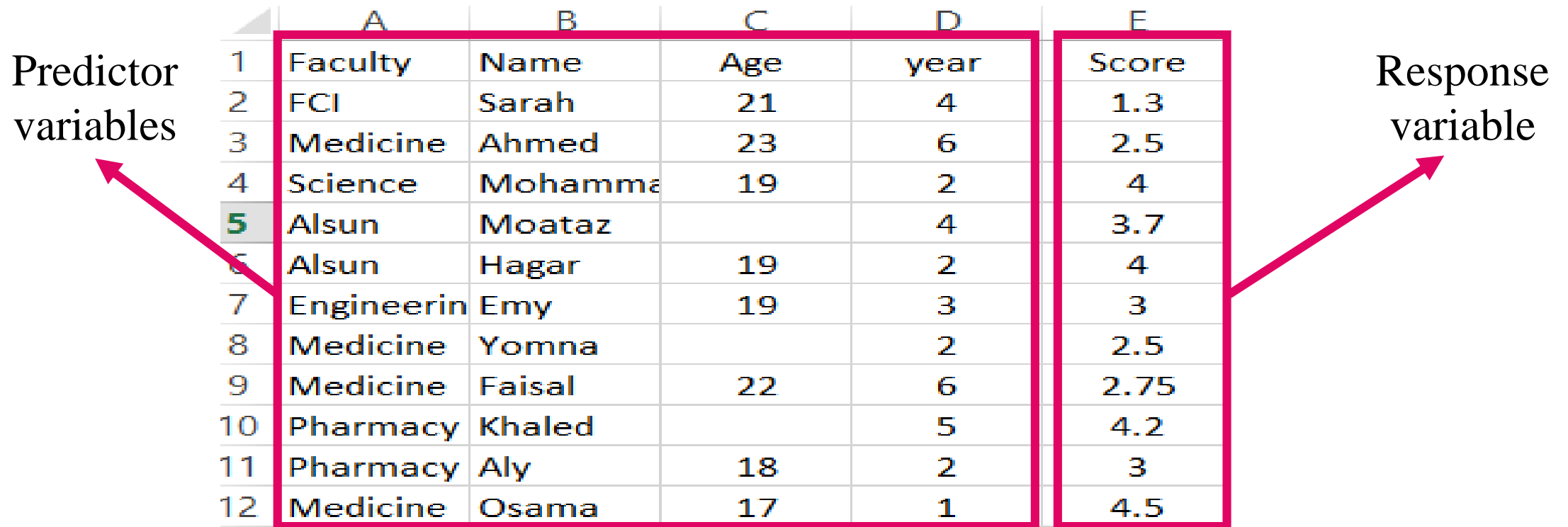
where **n** represents the number of existing (i.e. not missing) instance.

# Data Preprocessing

## Back to data cleaning: missing data

### □ Example(2.1):

- In this dataset, we have 5 columns which includes 4 features (independent variables) and one response (dependent variable).



	A	B	C	D	E
1	Faculty	Name	Age	year	Score
2	FCI	Sarah	21	4	1.3
3	Medicine	Ahmed	23	6	2.5
4	Science	Mohamma	19	2	4
5	Alsun	Moataz		4	3.7
6	Alsun	Hagar	19	2	4
7	Engineerin	Emy	19	3	3
8	Medicine	Yomna		2	2.5
9	Medicine	Faisal	22	6	2.75
10	Pharmacy	Khaled		5	4.2
11	Pharmacy	Aly	18	2	3
12	Medicine	Osama	17	1	4.5

# Data Preprocessing

## Back to data cleaning: missing data

### □ Example(2.1):

- In this dataset, we have 5 columns which includes 4 features (independent variables) and one response (dependent variable).

	A	B	C	D	E
1	Faculty	Name	Age	year	Score
2	FCI	Sarah	21	4	1.3
3	Medicine	Ahmed	23	6	2.5
4	Science	Mohammed	19	2	4
5	Alsun	Moataz		4	3.7
6	Alsun	Hagar	19	2	4
7	Engineering	Emy	19	3	3
8	Medicine	Yomna		2	2.5
9	Medicine	Faisal	22	6	2.75
10	Pharmacy	Khaled		5	4.2
11	Pharmacy	Aly	18	2	3
12	Medicine	Osama	17	1	4.5



# Data Preprocessing

## Back to data cleaning: missing data

### □ Example(2.1):

- In this dataset, we have 5 columns which includes 4 features (independent variables) and one response (dependent variable).

	A	B	C	D	E
1	Faculty	Name	Age	year	Score
2	FCI	Sarah	21	4	1.3
3	Medicine	Ahmed	23	6	2.5
4	Science	Mohamma	19	2	4
5	Alsun	Moataz		4	3.7
6	Alsun	Hagar	19	2	4
7	Engineerin	Emy	19	3	3
8	Medicine	Yomna		2	2.5
9	Medicine	Faisal	22	6	2.75
10	Pharmacy	Khaled		5	4.2
11	Pharmacy	Aly	18	2	3
12	Medicine	Osama	17	1	4.5

### Solution:

- We apply the mean to solve this problem

# Data Preprocessing

## Back to data cleaning: missing data

### □ Example(2.1):

- After calculate the mean

	A	B	C	D	E
1	Faculty	Name	Age	year	Score
2	FCI	Sarah	21	4	1.3
3	Medicine	Ahmed	23	6	2.5
4	Science	Mohammad	19	2	4
5	Alsun	Moataz	19.75	4	3.7
6	Alsun	Hagar	19	2	4
7	Engineering	Emy	19	3	3
8	Medicine	Yomna	19.75	2	2.5
9	Medicine	Faisal	22	6	2.75
10	Pharmacy	Khaled	19.75	5	4.2
11	Pharmacy	Aly	18	2	3
12	Medicine	Osama	17	1	4.5

# Data Preprocessing

## Data cleaning: categorical data

### □ Categorical data

- So far, we have only been working with numerical values. However, it is not uncommon that real-world datasets contain **one or more categorical feature columns**.
- When we are talking about categorical data, we have to further distinguish between **nominal and ordinal features**.



# Data Preprocessing

## Data cleaning: categorical data

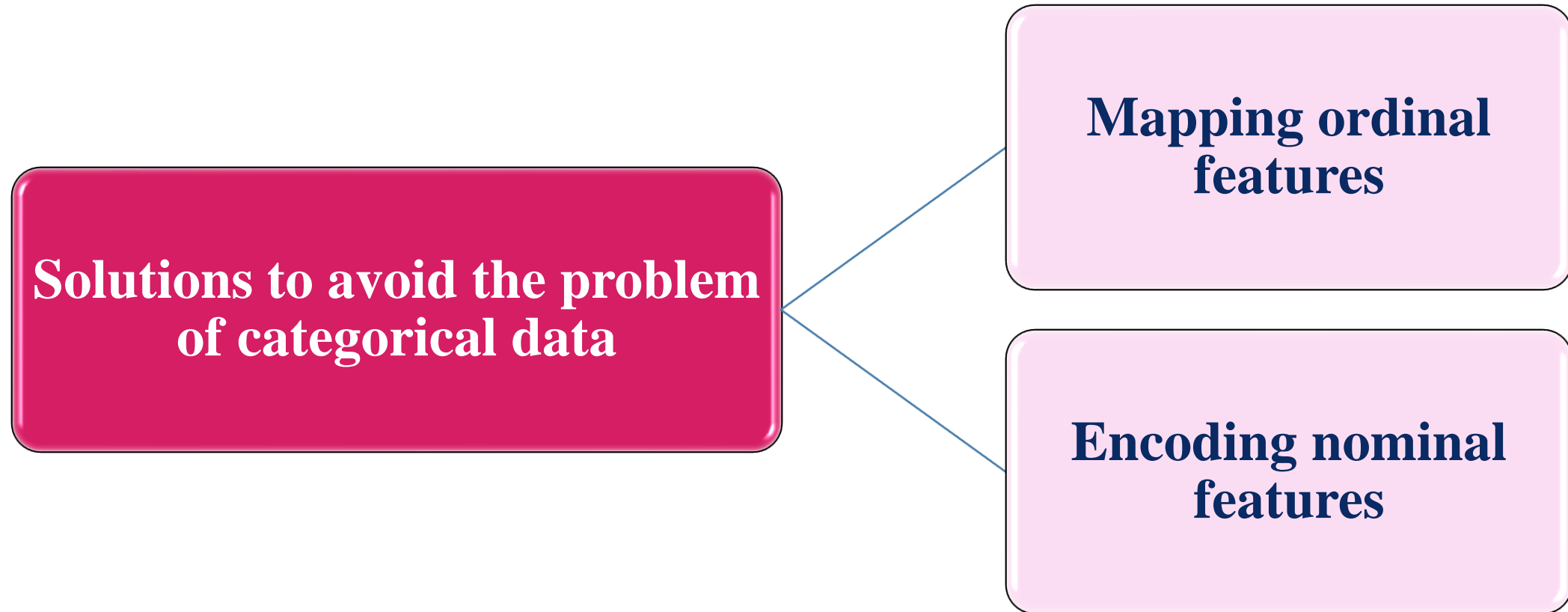
### □Categorical data(cont.)

- As we know the machine learning algorithms are based on mathematical equations so we can intuitively understand that it would cause some problem if we keep the text and the categorical variables in the equations because we would only want numbers in the equations.



# Data Preprocessing

## Data cleaning: categorical data

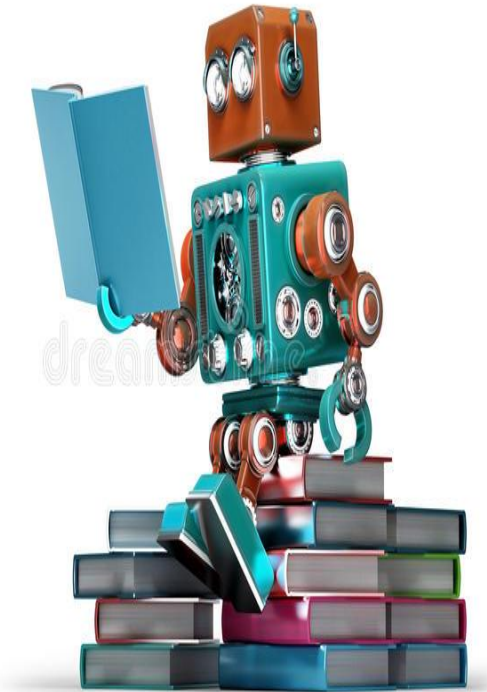


# Data Preprocessing

## Data cleaning: categorical data

### □ Mapping Ordinal Features

- To make sure that the learning algorithm interprets the ordinal features correctly, we need to *convert the categorical string values into integers*.
- Unfortunately, in Python there is no convenient function that can automatically derive the correct order of the labels of the ordinal feature. Thus, we have to define the mapping manually.



# Data Preprocessing

## Data cleaning: categorical data

### □ Example(2.2):

- In this dataset, we have three independent variables and one dependent variable.
- In this dataset all input variables consider as categorical features.



	A	B	C	D
1	Country	Color	Size	Prise
2	Egypt	red	S	150.75
3	USA	black	M	200
4	KSA	blue	L	170
5	Canda	white	S	300
6	Egypt	green	M	120
7	China	green	XI	200.5
8	Germany	blue	2XL	499.99



# Data Preprocessing

## Data cleaning: categorical data

### □ Example(2.2):

- At the first, we deal with the ordinal feature ( feature number 3 at column 2).

	A	B	C	D
1	Country	Color	Size	Prise
2	Egypt	red	S	150.75
3	USA	black	M	200
4	KSA	blue	L	170
5	Canda	white	S	300
6	Egypt	green	M	120
7	China	green	XI	200.5
8	Germany	blue	2XL	499.99

	Country	Color	Size	Prise
0	Egypt	red	1	150.75
1	USA	black	2	200.00
2	KSA	blue	3	170.00
3	Canda	white	1	300.00
4	Egypt	green	2	120.00
5	China	green	4	200.50
6	Germany	blue	5	499.99

# Data Preprocessing

## Data cleaning: categorical data

### □ Encoding Nominal Features

- Encoding categorical string data into numbers to be familiar for mathematical operations.
- This way is useful when we encoding the dependent variable/discrete target



# Data Preprocessing

## Data cleaning: categorical data

- ❑ After replacing the nominal values by the following values, the first column (A) of the previous matrix now holds the new **country** values, which are encoded as follows:

Nominal values	Encoded values
Egypt	2
USA	5
KSA	4
Canada	0
China	1
Germany	3

# Data Preprocessing

## Data cleaning: categorical data

- ❑ After replacing the nominal values by the following values, the second column (B) of the of the previous matrix now holds the new **color** values, which are encoded as follows:

Nominal values	Encoded values
red	3
black	0
blue	1
white	4
green	2

# Data Preprocessing

## Data cleaning: categorical data

Artificial Neural Networks  
&  
Deep Learning

***BUT !!!***

# Data Preprocessing

## Data cleaning: categorical data

- Although the color values don't come in any particular order, a learning algorithm will now assume that *white* is larger than *red* and the *red* is larger than *green*.
- Although this assumption is **incorrect**, the algorithm could still produce useful results. However, those results would not be optimal.

### □ Encoding nominal features (cont.)

- A common workaround for this problem is to use a technique called “**one-hot encoding**”.
- The idea behind this approach is to create a new **dummy** feature for each unique value in the nominal feature column.
- In our example, we would convert the country feature into new six features; **Egypt, USA, KSA, Canda, China and Germany** and also the color feature is converted into four new features: **red, black, blue, white and green**.
- Binary values can then be used to indicate the particular country and color of a sample.



# Data Preprocessing

## Data cleaning: categorical data

### ❑ Dummy encoding of country feature:

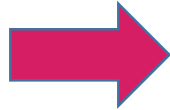
Country	Country_Egypt	Country_USA	Country_KSA	Country_Canda	Country_China	Country_Germany
Egypt	1	0	0	0	0	0
USA	0	1	0	0	0	0
KSA	0	0	1	0	0	0
Canda	0	0	0	1	0	0
China	0	0	0	0	1	0
Germany	0	0	0	0	0	1

# Data Preprocessing

## Data cleaning: categorical data

### ❑ Dummy encoding of color feature:

Color	Color_red	Color_black	Color_blue	Color_white	Color_green
red	1	0	0	0	0
black	0	1	0	0	0
blue	0	0	1	0	0
white	0	0	0	1	0
green	0	0	0	0	1



### □ Introduction to feature scaling:

- Allot of machine learning methods are based on what is called the **Euclidean** distance.
- If you remember that back from high school the **Euclidean** distance between two data points between two is a square root of the sum of the square coordinates.
- The machine learning algorithm calculate the **Euclidean** distance for input variables.
- The input variables have deference ranges, some of these ranges have a much wider range of values.

### □ Introduction to feature scaling(cont.):

- The **Euclidean** is often the “**default**” distance used in e.g., K-nearest neighbors (classification) or K-means (clustering) to find the “k closest points” of a particular sample point.
- Another prominent example is hierarchical clustering, agglomerative clustering (complete and single linkage) where you want to find the distance between clusters.

### □ Euclidean distance:

- In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space.
- The associated norm is called the **Euclidean norm**.
- The Euclidean distance between points ***p*** and ***q*** is the length of the line segment connecting them( $\overline{pq}$ ).

### □ Euclidean distance:

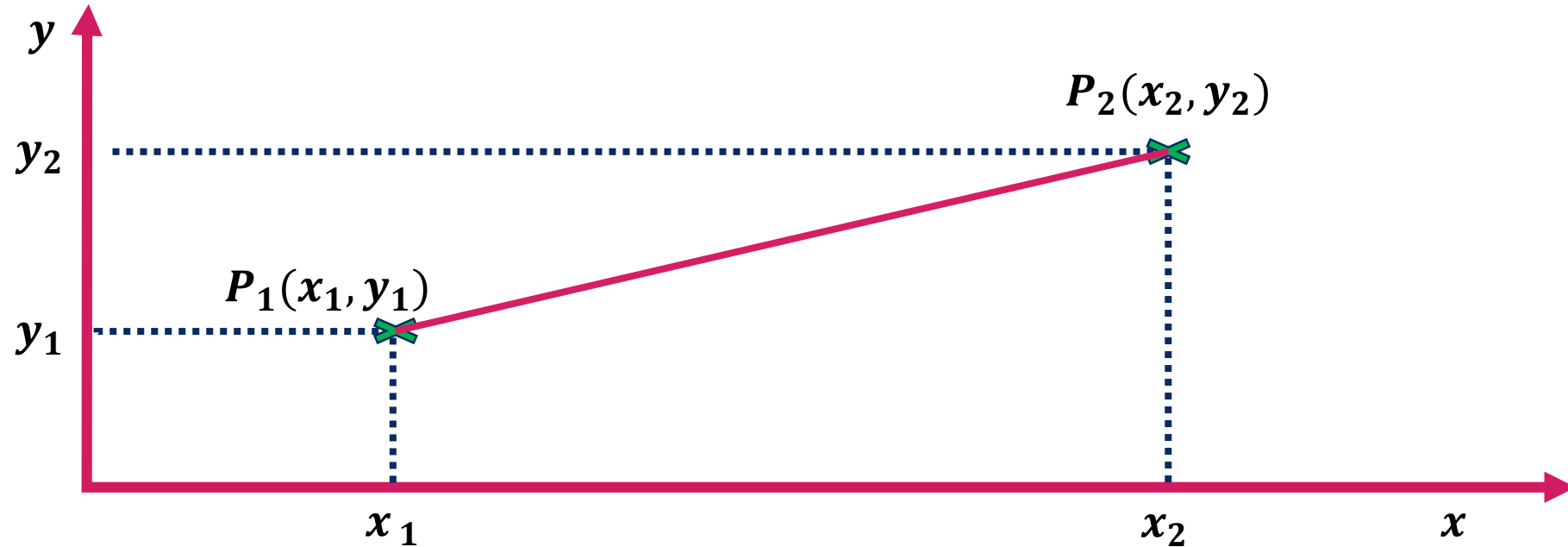
- In Cartesian coordinates, if  $p = (p_1, p_2, \dots, p_n)$  and  $q = (q_1, q_2, \dots, q_n)$  are two points in **Euclidean n-space**, then the distance ( $d$ ) from  $p$  to  $q$ , or from  $q$  to  $p$  is given by the Pythagorean formula:

$$\text{Euclidean distance between } q \text{ and } p \text{ in } n - \text{space} = \sqrt{\sum_{d=1}^n (q_d - p_d)^2}$$

# Data Preprocessing

## Feature scaling

### □ Euclidean distance:



$$\text{Euclidean distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



### □ Euclidean distance:

- For instance, if we have a dataset consists of two variables; age and salary. The age is between 21-40 and the salary is between 2000 – 7000.
- The Euclidean distance will be dominated by the **salary**.
- Assume we take two observations, and then we calculate the Euclidean distance between its values.

Age	Salary
21	2000
25	5000
30	6000
40	7000
38	7000
40	5000
23	6000

### ❑ Euclidean distance:

- The Euclidean distance will compute the difference between the second and fourth salary (25 and 40)
  - $\text{Salary distance} = 7000 - 5000 = 2000$ , after that we put the square to the result to be:
  - $(\text{Salary distance})^2 = (2000)^2 = \mathbf{4,000,000}$ .
- Now we calculate the difference between the age of the same observations
  - $\text{Age distance} = 40 - 25 \Rightarrow (15)^2 = \mathbf{225}$
- As we observed clearly that the salary dominates this square difference.
- The reason behind that, these two variables are on the difference scales (ranges).
- So that's why we absolutely need to put the variables in the same scale.

Age	Salary
21	2000
25	5000
30	6000
40	7000
38	7000
40	5000
23	6000

### □ Feature scaling:

- It is a method used to standardize the range of independent variables or features of data.

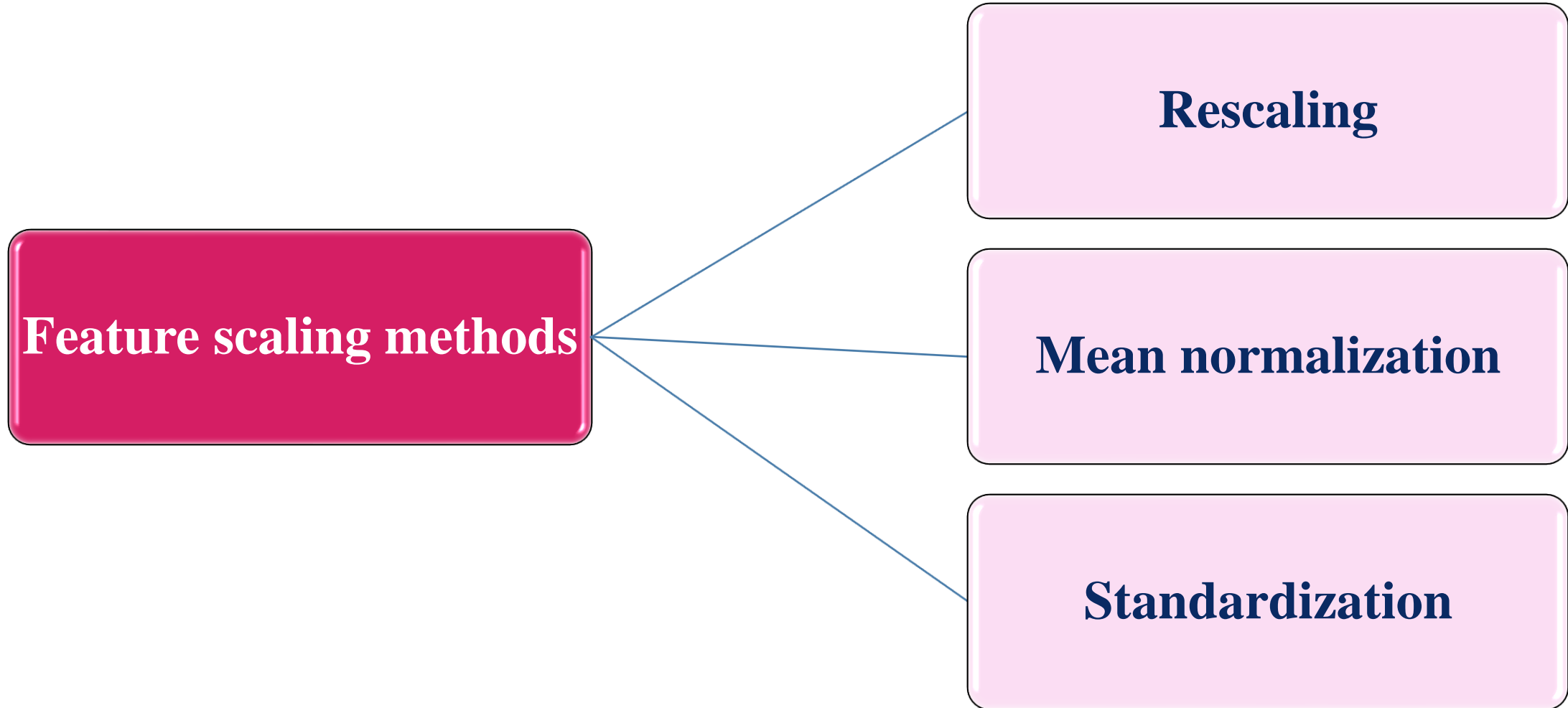
### □ We adopted data scaling for several reason:

- To avoid numerical difficulties during the calculation,
- To avoid features in greater numeric ranges dominating those in smaller numeric ranges (as shown in the previous example) and
- To help getting lower regression error and higher accuracy.

# Data Preprocessing

## Feature scaling

Artificial Neural Networks  
&  
Deep Learning



# Data Preprocessing

## Feature scaling: rescaling

### □ Rescaling:

- The simplest method is rescaling the range of features to scale the range in  $[0, 1]$ .
- Selecting the target range depends on the nature of the data. The general formula is given as:

$$x_{i_{scaled}} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- where  $x_i$  is an original value,  $x_{i_{scaled}}$  is the normalized/scaled value and  $x$  is the feature vector

# Data Preprocessing

## Feature scaling: rescaling

### □ Example(2.3):

- In the following dataset, we have three independent variables and one dependent variable.

	A	B	C	D
1	High temp	Solar radiation	Wind speed	Rain
2	13	192	35	18.00
3	39	178	56	12.00
4	9	199	74	23.00
5	14	100	57	12.00
6	1	187	54	25.00
7	-4	139	57	14.00
8	46	156	58	13.00
9	6	195	51	24.00
10	-4	166	38	10.00

High temp	Solar radiation	Wind speed
0.34	0.92929293	0
0.86	0.78787879	0.53846154
0.26	1.0	1.0
0.36	0.0	0.56410256
1.0	0.87878788	0.48717949
0.2	0.39393939	0.56410256
0.0	0.56565657	0.58974359

# Data Preprocessing

## Feature scaling: mean normalization

### □ Mean normalization:

- The general formula is given as:

$$x_{i_{scaled}} = \frac{x_i - mean(x)}{\max(x) - \min(x)}$$

- where  $x_i$  is an original value,  $x_{i_{scaled}}$  is the normalized/scaled value and  $x$  is the feature vector

# Data Preprocessing

## Feature scaling: standardization

### □ Standardization:

- This method is widely used for normalization in many machine learning algorithms.
- The general formula is given as:

$$x_{i_{scaled}} = \frac{x_i - \bar{x}}{\sigma}$$

- Where  $x_i$  is the original value,  $\bar{x}$  is the mean of that feature vector, and  $\sigma$  is its standard deviation.



### □ Scaling to unit length:

- Another option that is widely used in machine-learning is to scale the components of a feature vector such that the complete vector has length one. This usually means dividing each component by the **Euclidian** length of the vector
- The general formula is given as:

$$x_{i_{scaled}} = \frac{x_i}{\|x\|}$$

- Where  $x_i$  is the original value and  $\|x\|$  is the **Euclidian** length of the feature vector  $x$ .

### □ Dimensionality Reduction:

- In machine learning and statistics, **dimensionality reduction** or dimension reduction is the process of **reducing** the number of **random variables** under consideration.
- There are two main categories of dimensionality reduction techniques: **feature selection** and **feature extraction**.
  - **Feature Selection:** we **select** a subset of the original feature set.
  - **Feature Extraction:** we **derive** information from the feature set to construct a new feature subspace

### □ Definition 2.1 (training set):

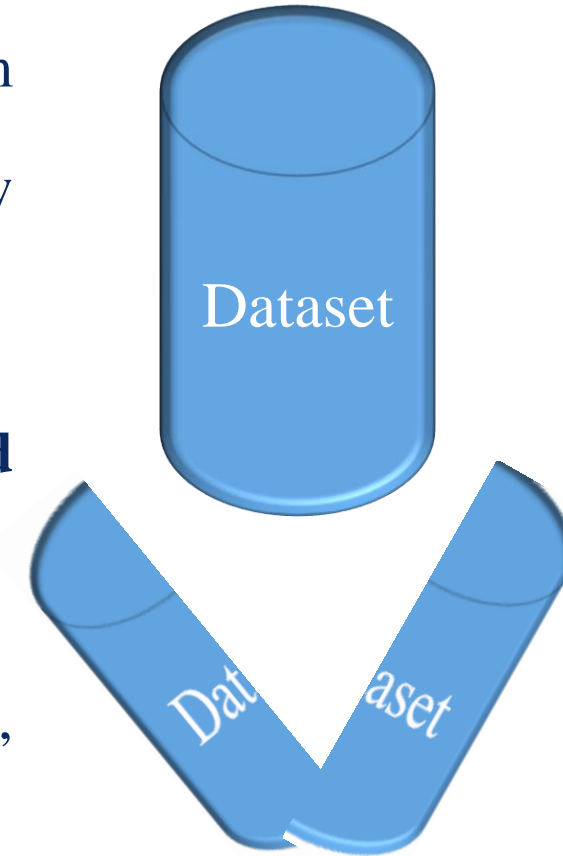
- A training set (*named  $P$* ) is a set of training dataset (e.g. patterns), which we use to train our learning algorithm.

### □ Definition 2.2 (testing set):

- A testing set (*named  $P_t$* ) is a set of testing dataset (e.g. patterns), which we use to test the prediction/classification model that generated by our learning algorithm.

### □ Partitioning a dataset in training and testing sets:

- Partitioning is used to determine whether our machine learning algorithm **not only performs well** on the training set but also **generalizes** well to new data,
- We want to **randomly divide** the dataset into a separate **training and testing sets**.
- We use the **training** set to **train** and **optimize** our machine learning model, while we keep the **testing** set until the very end to **evaluate** the final model.



### □ Partitioning a dataset in training and testing sets(cont.):

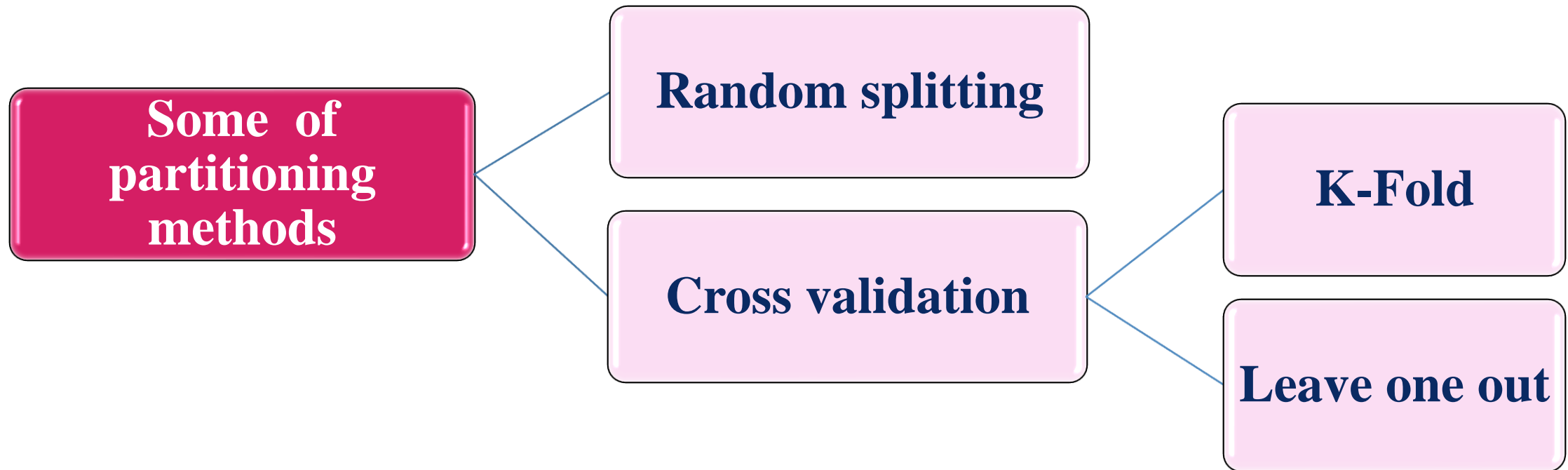
- In other word, learning the parameters of a prediction function and testing it on the **same data** is a methodological **mistake**: a model that would just repeat the labels of the samples that it has just seen would have a **perfect score but would fail to predict anything useful on yet-unseen data**.
- This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to **hold out** part of the available data as a **test set**.



### □ Partitioning a dataset in training and testing sets(cont.):

- When we use a machine learning algorithm, we do not fix the parameters ahead of time, then we need to split our dataset.
- The training set is used to choose the parameters to reduce training set error, then used the test set.
- Under this process, the expected test error is greater than or equal to the expected value of training error. The factors determining how well a machine learning algorithm will perform are its ability to:

### □ Partitioning methods:



### □ Random splitting

- Split arrays or matrices into random train and test subsets





### ❑ Cross validation:

- **Cross-validation** is a technique for **evaluating and comparing** the performance of learning algorithms.
- The performance of the algorithm or the model can be measured using many ways depending on the task ( e.g. for classification task error rate is the usual performance measure).
- The simplest basic for **cross-validation** is the **handout method**.
- The dataset is divided into two subsets; one for training purpose and the other for testing purpose.
- The learning algorithm process is obtained using training dataset, then it gives prediction on the test dataset.

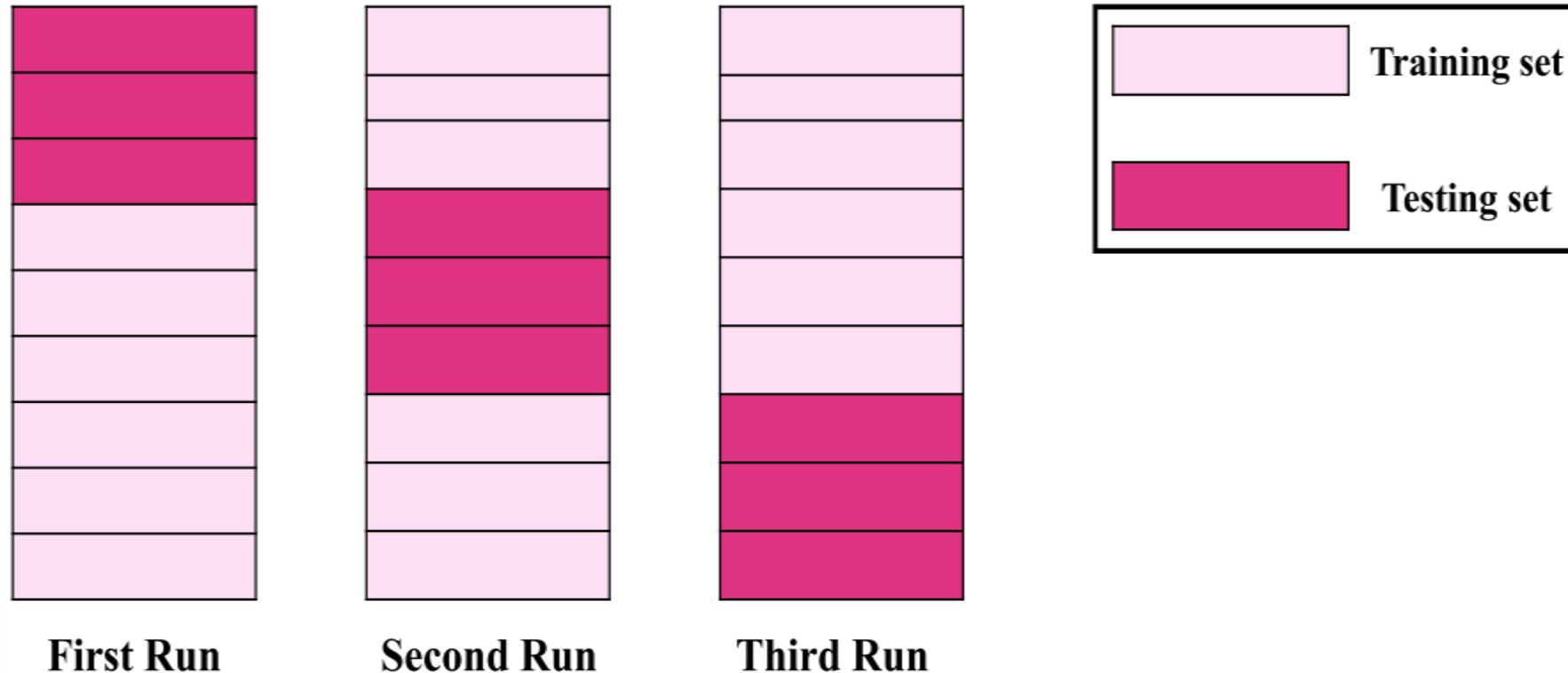


### □ K-Fold cross validation:

- In k-fold cross-validation, the original samples of the dataset were **randomly** partitioned into k subsets of (approximately) equal size and the experiment is run k times.
- For each time, one subset was used as the testing set and the other k-1 subsets were used as the training set.
- The average of the k results from the folds can then be calculated to produce a single estimation of the performances of K times.

### □ K-Fold cross validation:

- The following figure shows the k-fold cross validation schema when  $k = 3$ .



### □ Leave one out(LOO) cross validation:

- LOO is a simple method of cross-validation.
- Each learning set is created by taking **all the** samples **except one**, the test set being the sample **left out**.
- Thus, for samples, we have different training sets and different tests set.
- This cross-validation procedure does not waste much data as only one sample is removed from the training set



### □ Example(2.3):

- Assume that, we have the following dataset

Index	$X_1$	$X_2$	Y
0	1	20	11
1	2	25	1
2	2.2	45	10
3	2.4	5	12
4	2.3	4	20
5	4.55	33	2
6	5.8	8	13
7	8.8	2	34
8	9	1	30
9	10	10	13
10	10	2	24

[ 1	2	3	4	5	6	7	8	9	10	[0]
[ 0	2	3	4	5	6	7	8	9	10	[1]
[ 0	1	3	4	5	6	7	8	9	10	[2]
[ 0	1	2	4	5	6	7	8	9	10	[3]
[ 0	1	2	3	5	6	7	8	9	10	[4]
[ 0	1	2	3	4	6	7	8	9	10	[5]
[ 0	1	2	3	4	5	7	8	9	10	[6]
[ 0	1	2	3	4	5	6	8	9	10	[7]
[ 0	1	2	3	4	5	6	7	9	10	[8]
[ 0	1	2	3	4	5	6	7	8	10	[9]
[0	1	2	3	4	5	6	7	8	9]	[10]

Testing sets

Training sets

# References

- Raschka, Sebastian. “Python machine learning”. Packt Publishing Ltd, 2015.

# Any Questions!?



*Thank you*