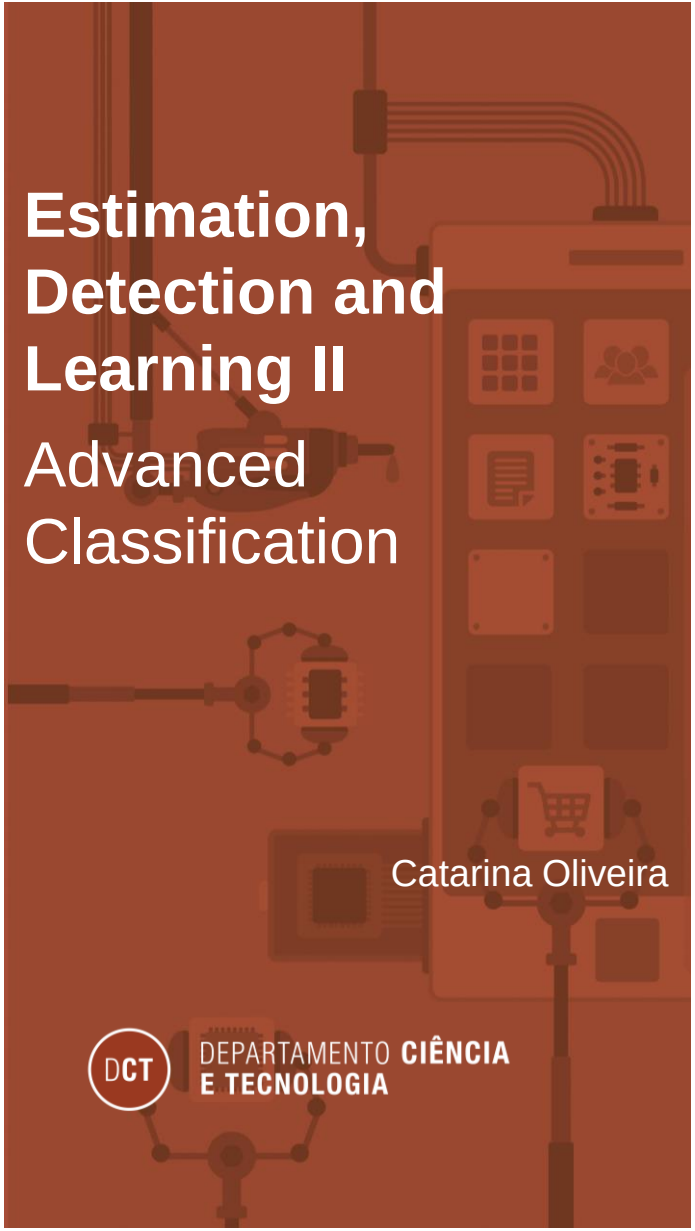


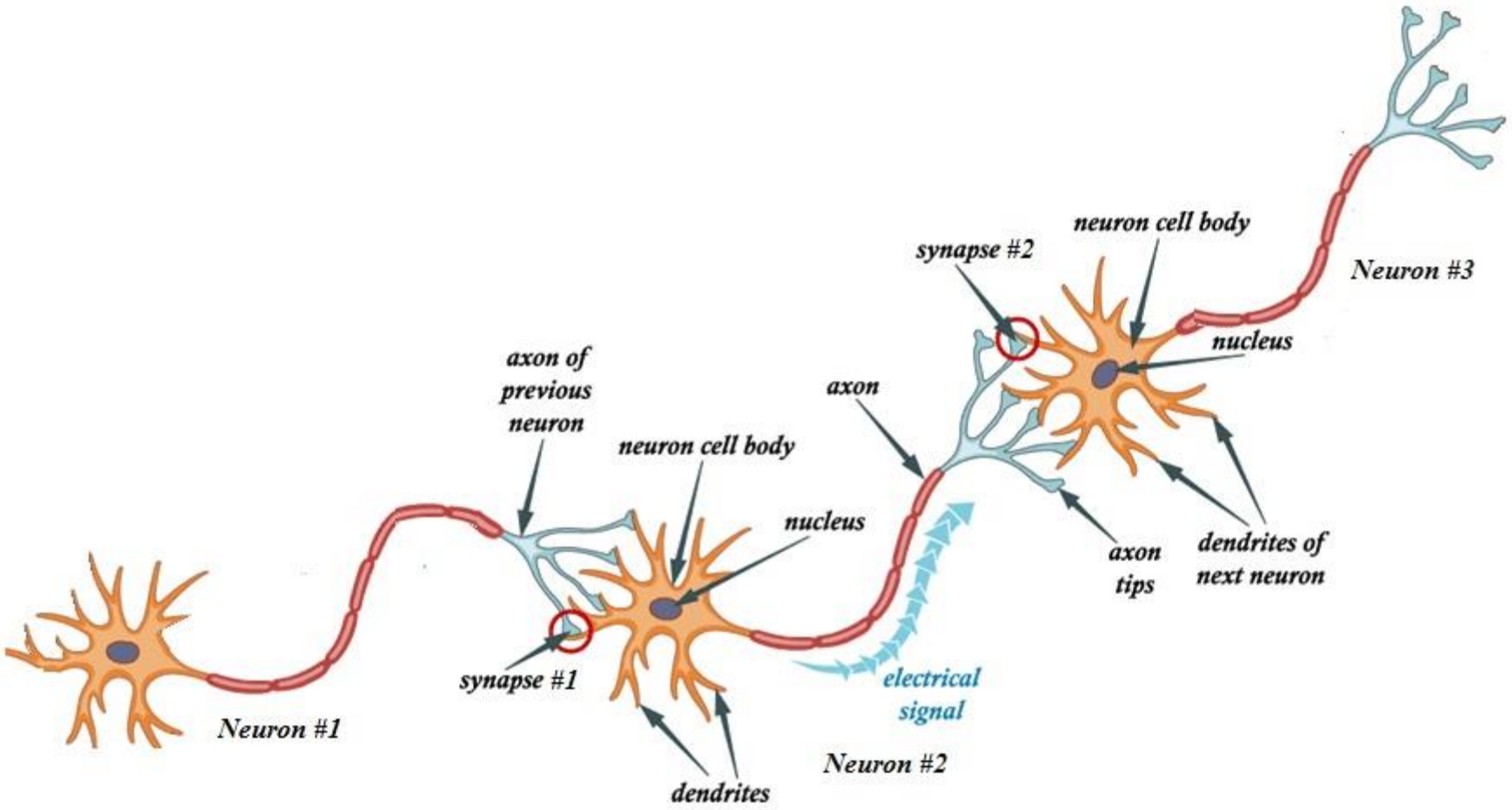
UNIVERSIDADE PORTUCALENSE

CONTENT

1. neural networks
2. Support vector Machines
3. nearest neighbor
4. Semi-supervised
5. transfer learning

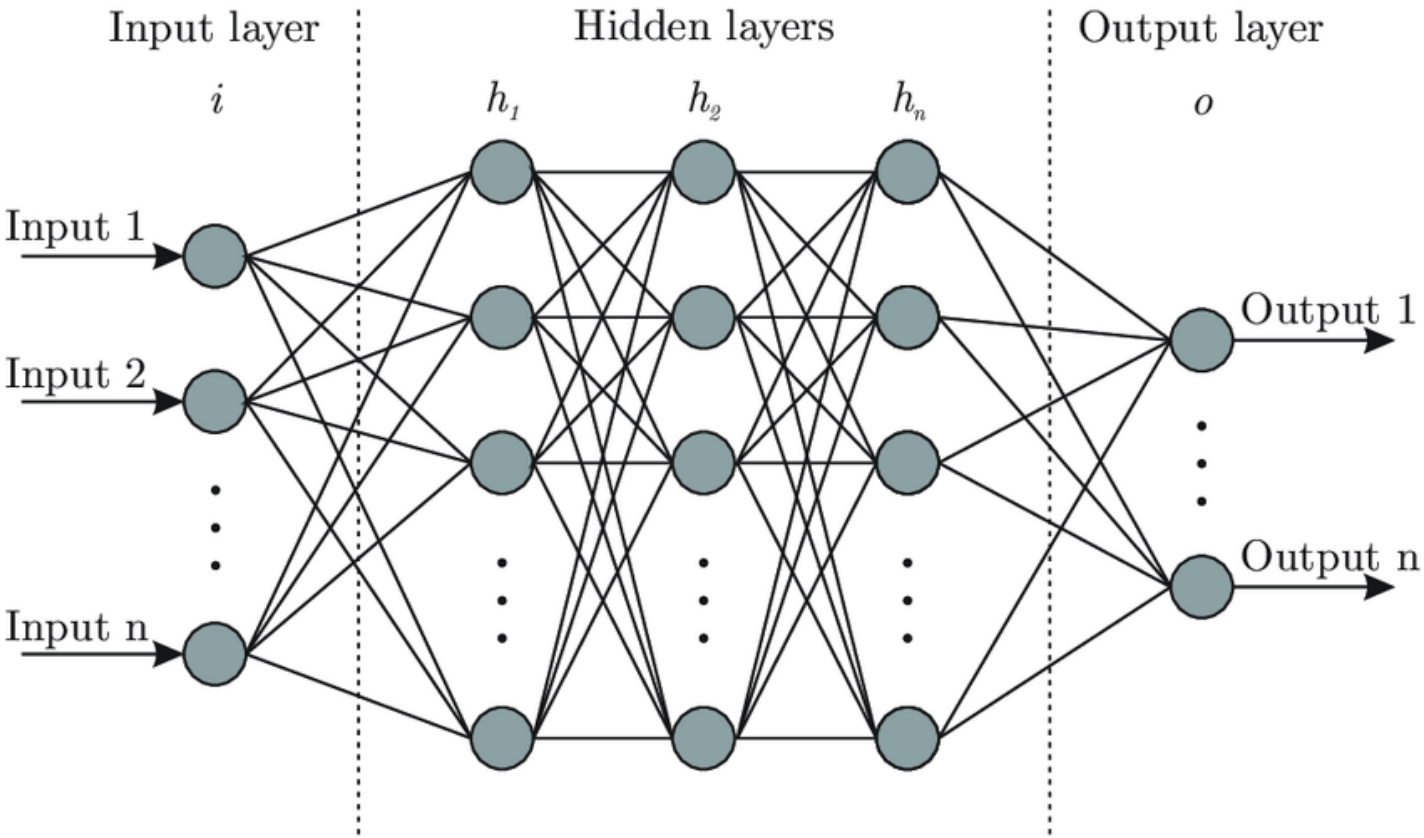
Neural Networks
Neural Networks

Neural networks: biological context



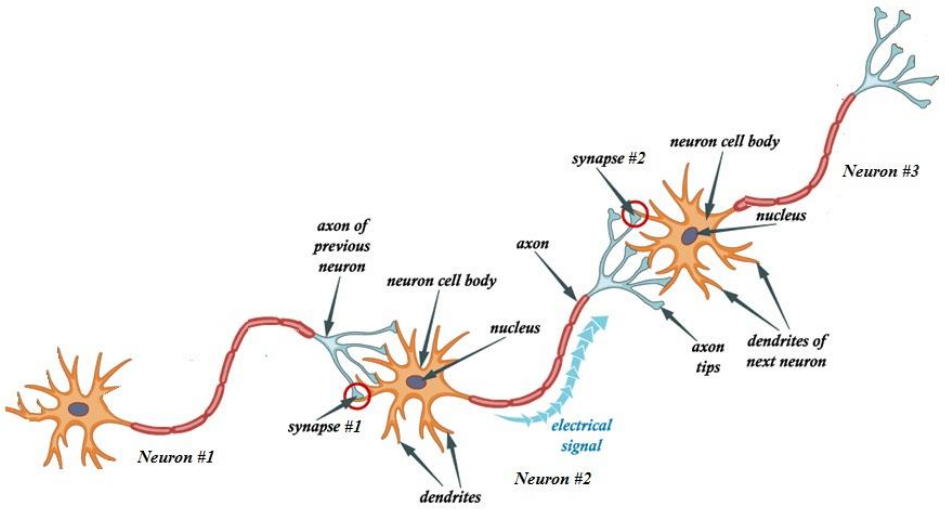
https://www.chegg.com/homework-help/questions-and-answers/let-s-put-together-review-steps-action-potential-events-synapses-fill-following-blanks-tra-q41393959

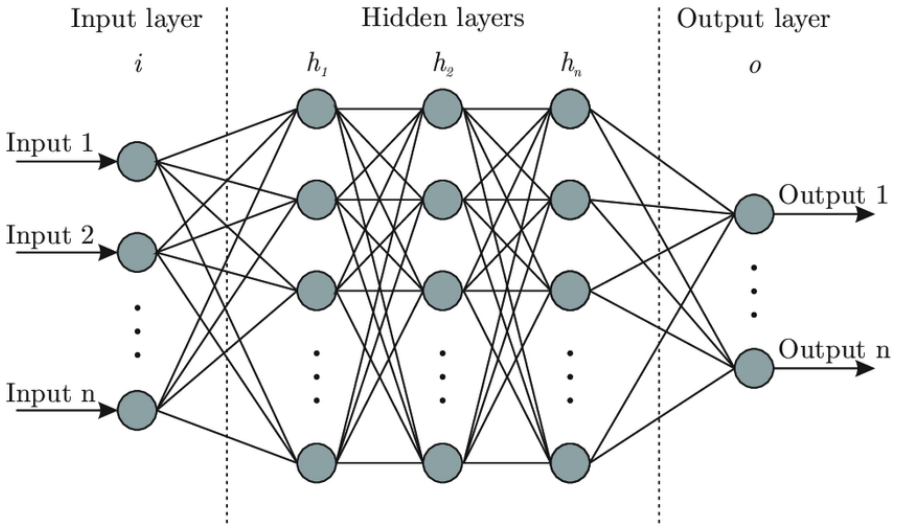
Neural Networks: Artificial Neural Networks (ANNs)



Bre , Facundo & Gimenez, Juan & Fachinotti , Victor. (2017). Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. Energy and Buildings . 158. 10.1016/j.enbuild.2017.11.045.

Neural Networks: biological vs. ANNs



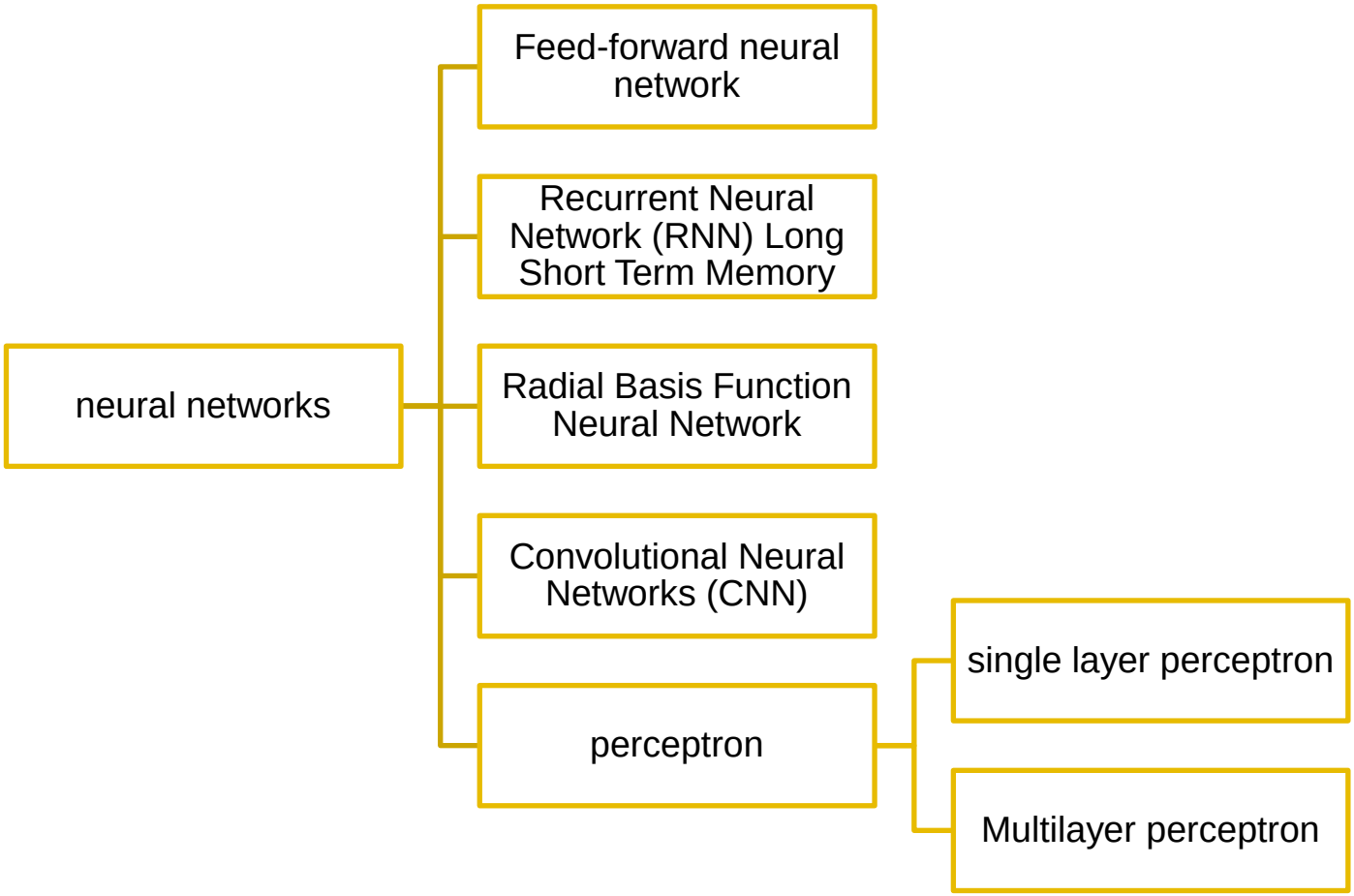


The human brain contains neurons that are composed by several *dendrites* (providing inputs to the neuron) and an *axon* (working as the neuron's output). The neurons are connected and the connections, called *synapses*, allow the communication between neurons. When neurons "fire", they send an electrical impulse that propagates through the cell body, to the *axon* and then the *synapse*. From here, the electrical impulse acts as an input for the subsequent neuron. Each *synapse* has an associated strength and the combination of all the inputs received, when compared to a certain threshold, will define if the neuron will "fire" an electrical impulse to the subsequent neuron.

We focus on the multi-layer perceptron which contains units (*neurons*) organised in layers: the input layer, at least one hidden layer, and the output layer. The units on one layer are connected to the units in the subsequent layer. The output of each unit passes through the connections (*synapses*) to the units in the next layer. This simulates the input and output through *dendrites* and *axon*, respectively. The connections between units have associated weights (*synapse* strength) that influence the impact of the information passed to the next unit.

Each layer has an associated activation function. The input values from the previous layer's units are fed to the activation function, which aggregates them into a single value that is passed onto the following layer's units. The middle layers are said to be hidden because their activation values are not directly accessible from the outside.

Types of neural networks



Feed-forward Neural Network

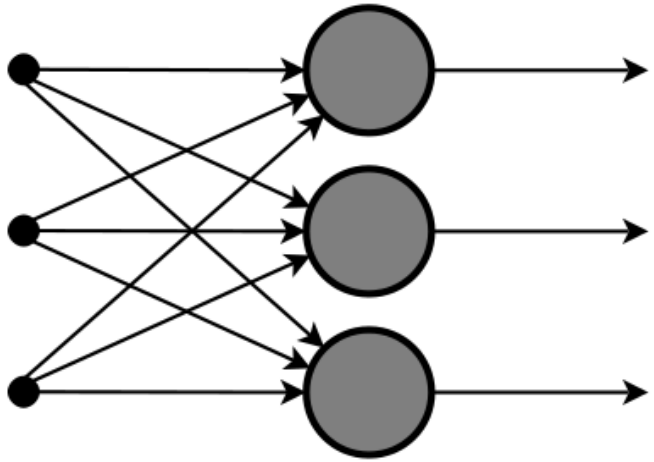
Data only moves in one direction from input to output.

Along the way, the sum of the products of inputs and weights is calculated.

The final result is passed to the outputs for processing.

Mainly used in **facial recognition and computer vision**

They are equipped to handle data that contains a lot of noise.



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Recurrent Neural Network (RNN) Long Short Term Memory

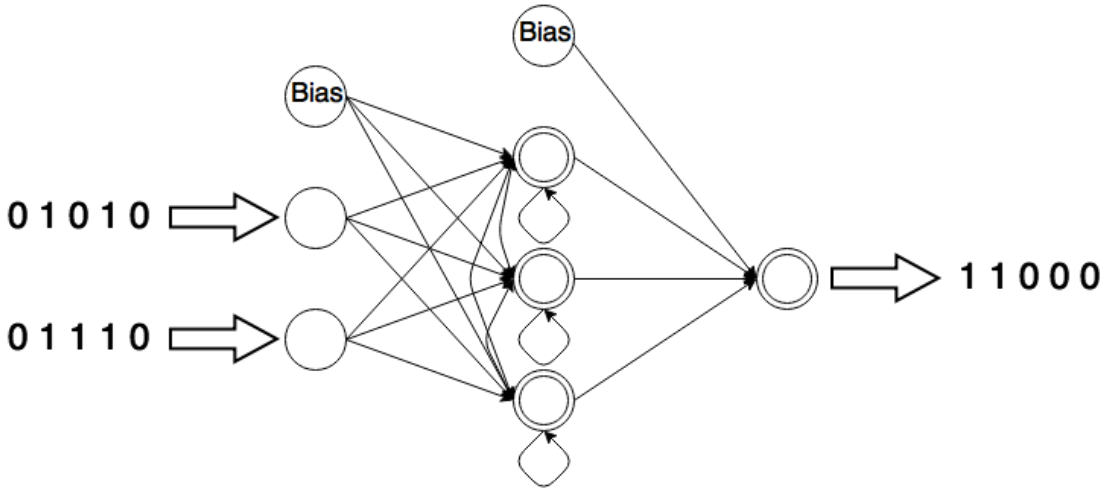
It stores the output of a layer and feeds back the input.

feedforward network , the sum of inputs and weights being calculated.

In the following layers, the recurrent process begins: at each iteration, the node remembers some information it had in the previous iteration. Acts like a memory cell while computing and performing the operation.

The network starts out the same as the feedforward network , but it has memory of the information so it can use it later.

Very effective for **converting voice to text**



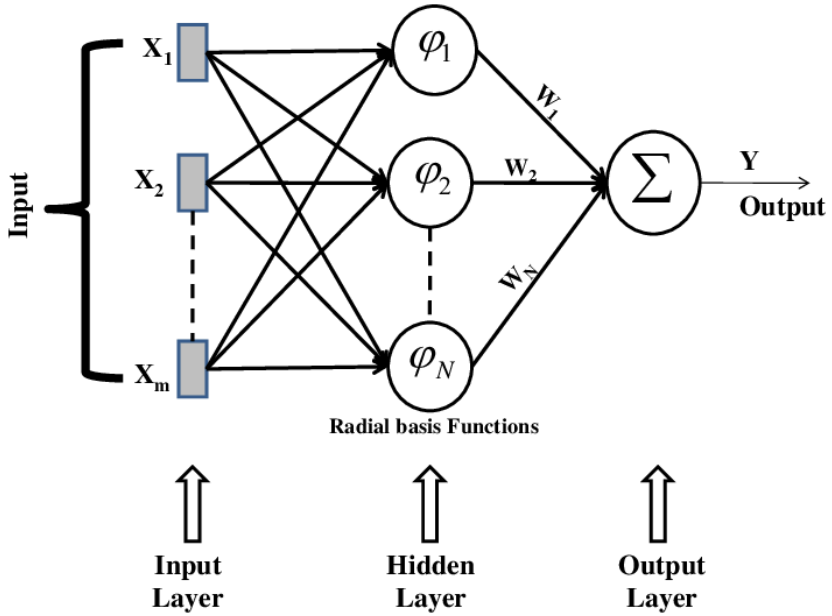
https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Radial Basis Function Neural Network

The distance of any point from the center is considered.

It has two layers: inner and outer. The inner layer has the features combined with the radial basis function (function whose value depends on the distance between the input and a fixed point). The output from these features is then used when calculating the same output in the next iteration.

Mainly used in **power restoration systems.**



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

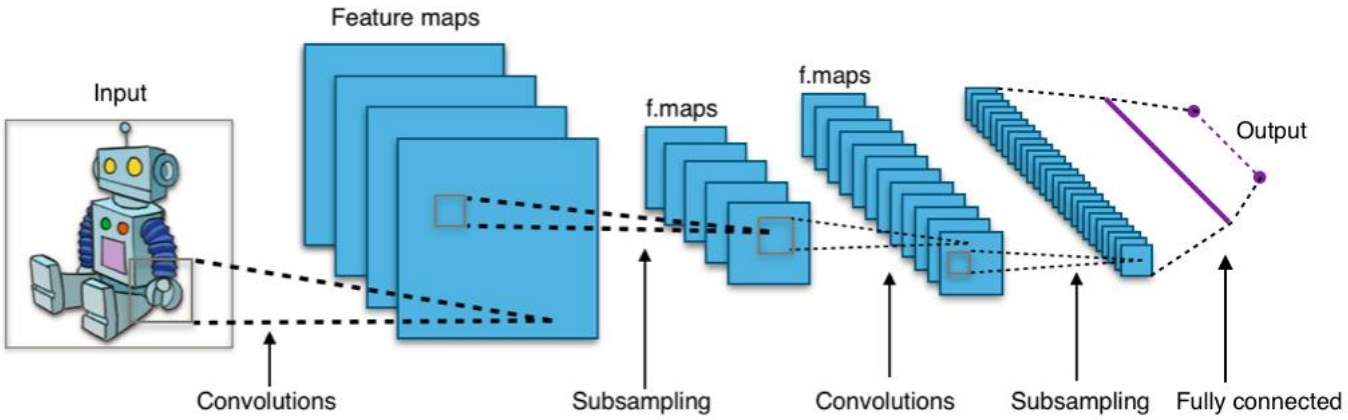
Convolutional Neural Networks (CNN)

Main objective: extract features from the input image.

Convolution preserves the spatial relationship between pixels by extracting image features using small squares of input data.

Composed of one or more fully connected convolutional layers.

Used in **computer vision** , **object recognition** (eg autonomous vehicles).



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

perceptron

Algorithm for supervised learning of binary classifiers

It allows neurons to learn and process elements in the training set one at a time.

Performs calculations to decide whether or not an input belongs to a specific class.

There are two main types of perceptrons :

- single layer perceptrons (single layer)
- multilayer perceptrons (multiple layers)

They are classified as feed-forward neural networks : they move only in one direction.

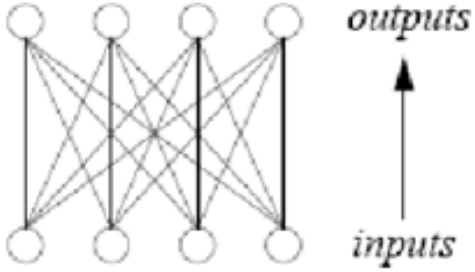
single layer perceptron

Has no prior knowledge of any input: initial weights are randomly assigned.

Sums all weighted inputs (weights), and if the sum is above a threshold , the perceptrons are on.

Input values are presented to the perceptron and:

- if the predicted output is the same as the desired one, the performance is considered satisfactory and no changes are made to the weights.
- if the predicted output does not match the (adjusted to reduce the error.



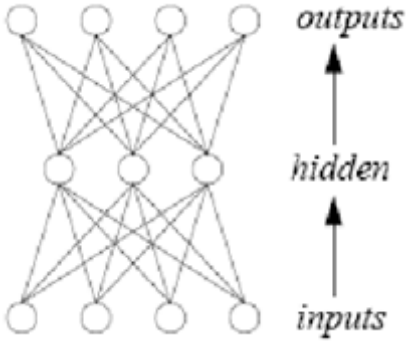
https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Multilayer perceptron

It has the same structure as the *single layer perceptron* , but with one or more hidden layers added.

The algorithm consists of two phases:

- the *forward phase* , where activations are propagated from the input layer to the output layer
- the *backward phase* , where the error is propagated backwards in order to modify the weights and *bias values* .



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

extra information

For more information about neural networks, see:

https://towardsdatascience.com/understanding-neural-networks-19020b758230

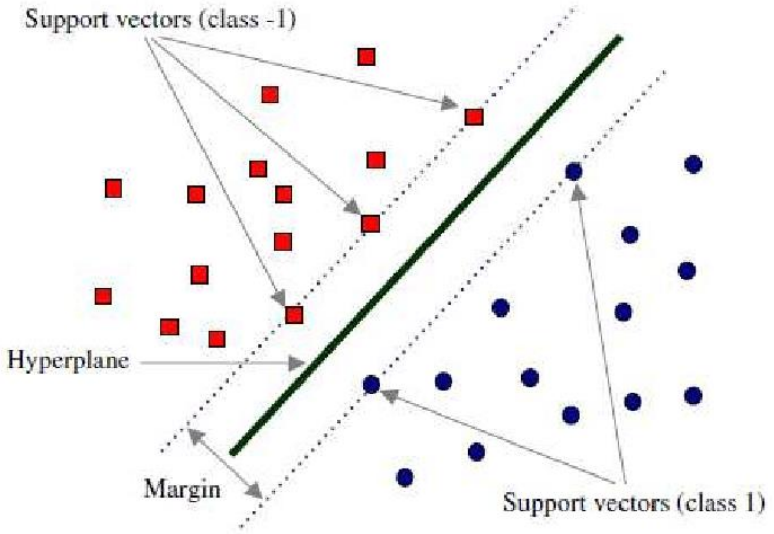
deep Learning

Neural networks with many hidden layers

Support vector Machines

Support vector machines : concepts

Objective: find the best separation of classes



Concepts:

- **Hyperplane** :
  - 1d: point that best separates the classes
  - 2d: line that best separates the classes
  - 3d: plane that best separates classes
- **Support vectors** : points closest to the hyperplane
- **Margin** : distance between the closest points of each class
- **Kernel** : mathematical function used to transform the input data to another format (eg linear, nonlinear , polynomial, …) and quantifies the similarity (distance) between two observations

Support vector machines : parameters

- **Gamma** : defines the influence of a singular point
  - Higher values: low influence
  - Lower values: high influence
- **C** (regularization parameter): controls the tradeoff :
  - smoother decision boundary (lower values)
  - rank training points correctly (higher values).

class `sklearn.svm.` **SVC**(*, *C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

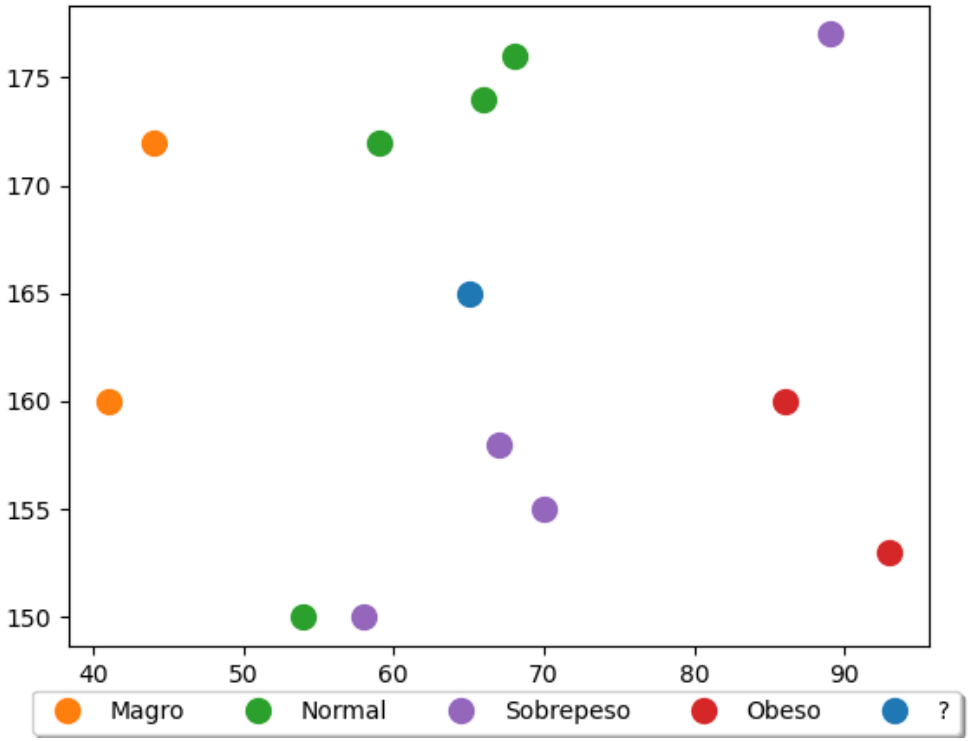
Advantages and disadvantages

Benefits	Disadvantages
• Versatile for specific *kernel functions* • Efficient in terms of memory • Effective when the number of dimensions is greater than the number of samples	• Subject to errors and *overfitting* when dealing with noisy data (eg overlapping points with the same *label*) • Long computation time when dealing with very large datasets • Does not provide a probabilistic explanation of results

nearest neighbor
Nearest Neightboro

Problem

Peso	Altura	IMC
41	160	Magro
44	172	Magro
54	150	Normal
58	150	Sobrepeso
59	172	Normal
66	174	Normal
67	158	Sobrepeso
68	176	Normal
70	155	Sobrepeso
86	160	Obeso
89	177	Sobrepeso
93	153	Obeso
65	**165**	**?**



Nearest K Algorithm Neighbors

Input:

- O: Set of observations with *label*
- N: New observation without *label*
- K: number of neighbors to consider

Steps:

1. Calculate the similarity of *N* to O_i
2. Get the *K* nearest neighbors
3. Determine *label (N)* according to neighbors' *labels*
    - Rating: fashion
    - Regression: average

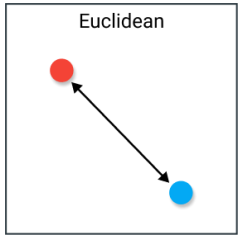
Output:

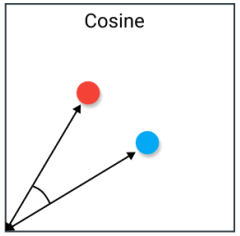
- *Label (N)*

similarity calculation

euclidean

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



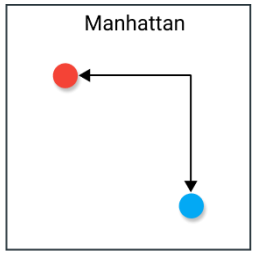


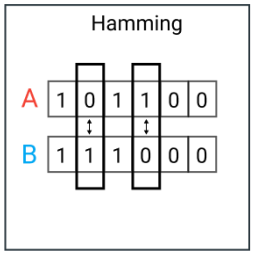
Sew similarity (angle between vectors)

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Manhattan

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$



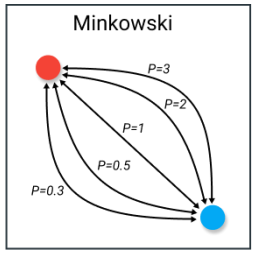


Hamming (distance between strings)

Number of different characters between strings

Minkowski

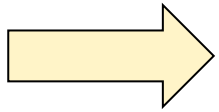
$$d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$



Distance measurements: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

In the example

Peso	Altura	IMC	Distância
67	158	Sobrepeso	7,3
66	174	Normal	9,1
59	172	Normal	9,2
70	155	Sobrepeso	11,2
68	176	Normal	11,4
58	150	Sobrepeso	16,6
54	150	Normal	18,6
86	160	Obeso	21,6
44	172	Magro	22,1
41	160	Magro	24,5
89	177	Sobrepeso	26,8
93	153	Obeso	30,5
65	**165**	**?**	

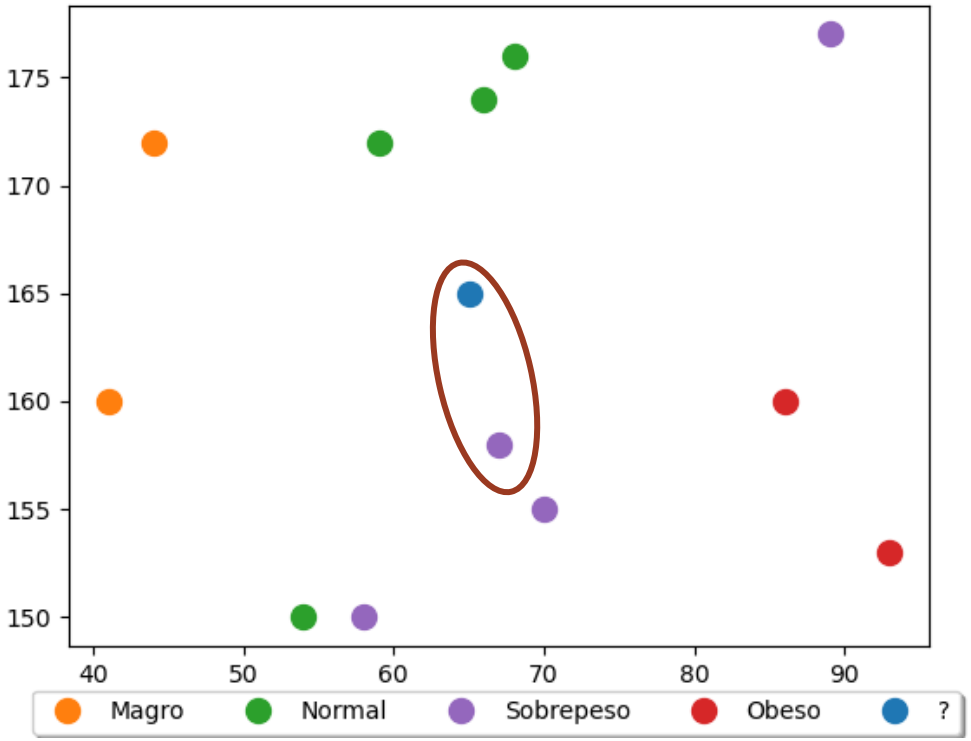


Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

In the example

K=1

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

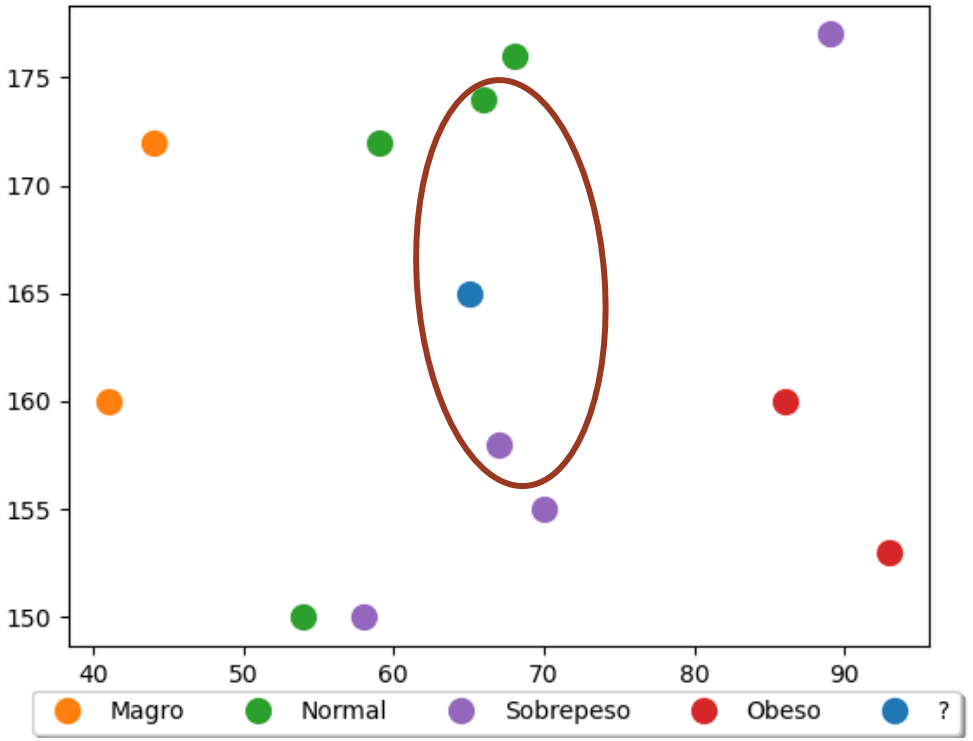


Label : Overweight

In the example

K=2

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

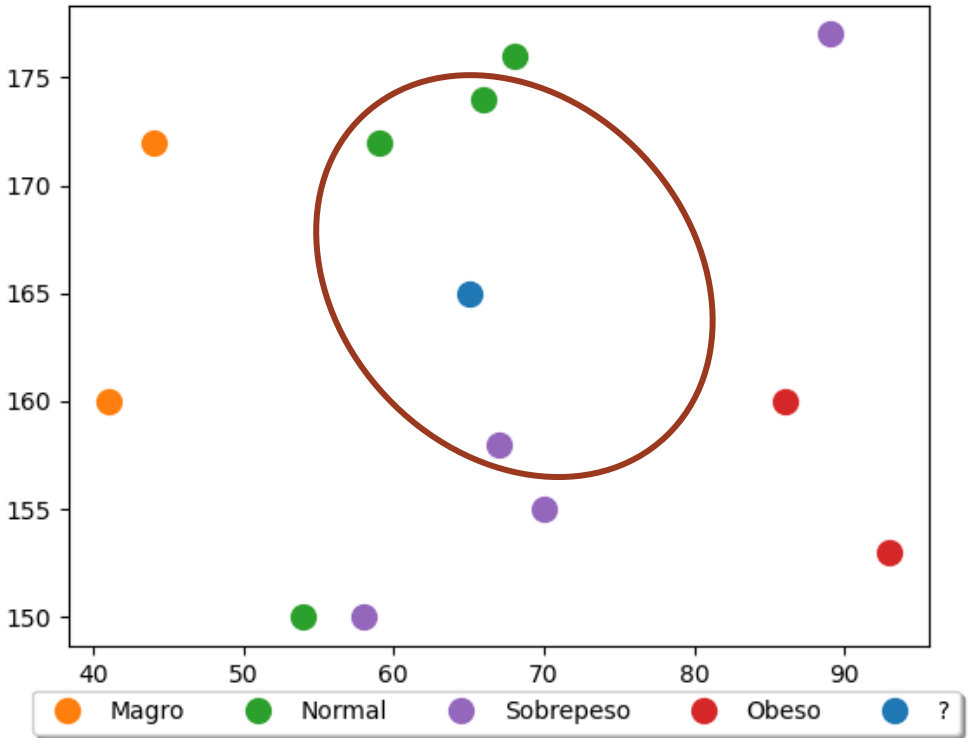


Label : Normal / Overweight

In the example

K=3

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



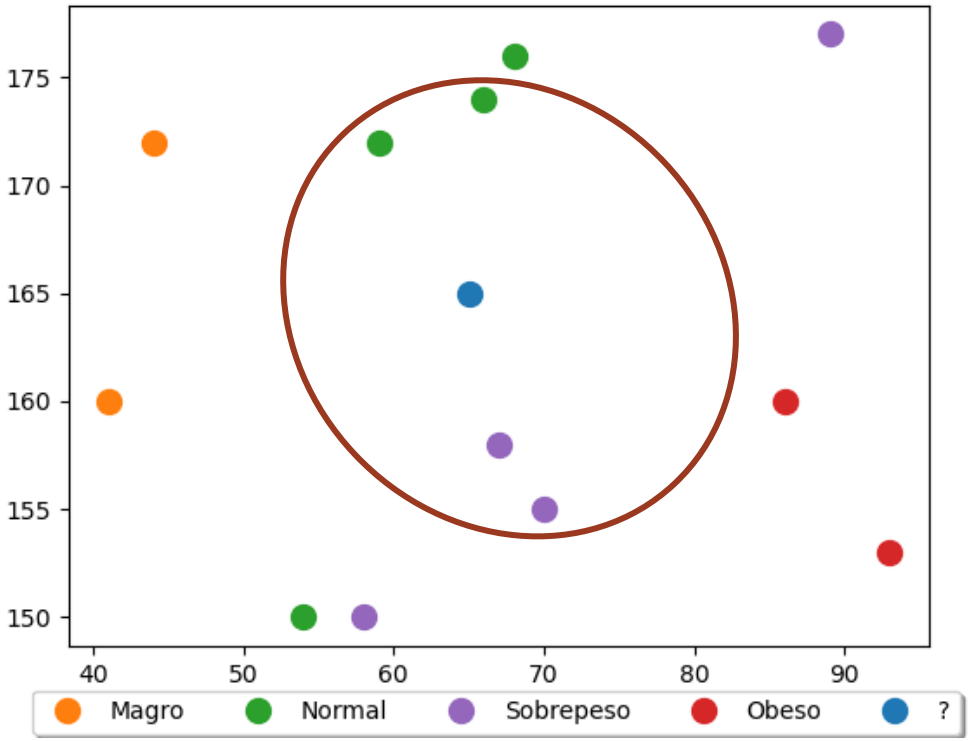
Label : Normal

In the example

K=4

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

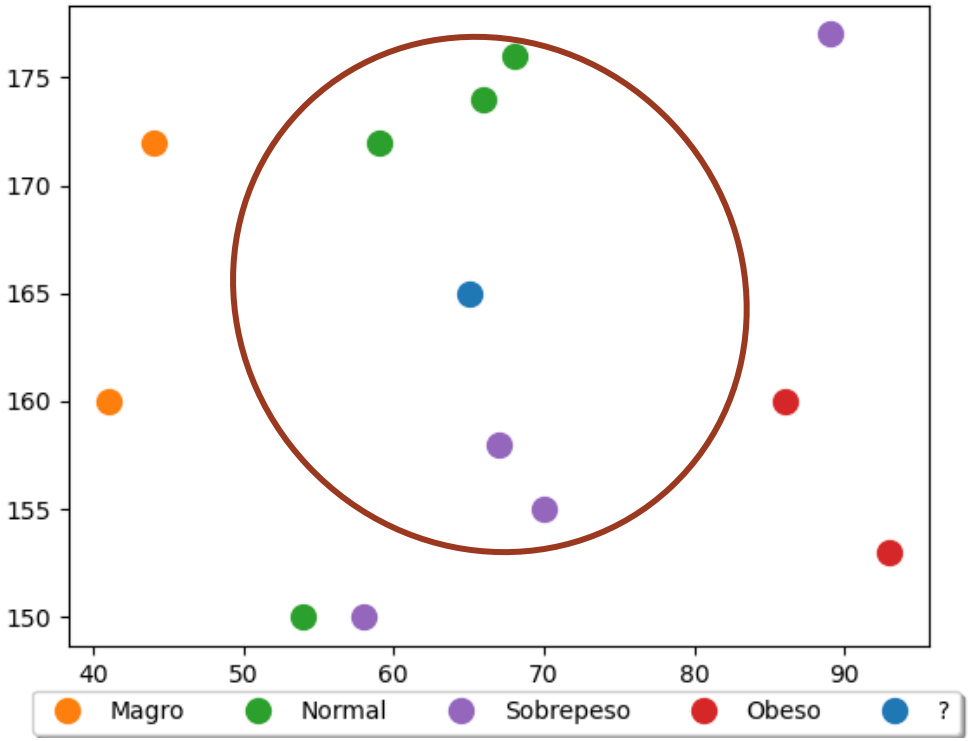
Label : Normal / Overweight



In the example

K=5

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



Label : Normal

Determination of the best K

- There is no structured method to find the best value for K. It is necessary to test several values in trial and error.

- Choosing lower values for K can introduce noise and will have a greater influence on the result.
    - K = 1: sensitive to outliers

- Higher values of K will have lower variance but higher *bias* . It is computationally more “expensive”.
    - K = n, *n* is the number of training data samples: all new samples will have the same *label*
        - Rating: fashion
        - Regression: average

- Use *cross validation* : separate training set into training and validation and use it to evaluate different K values, choosing the one that allows better performance (smallest error: validation *error minimization*).

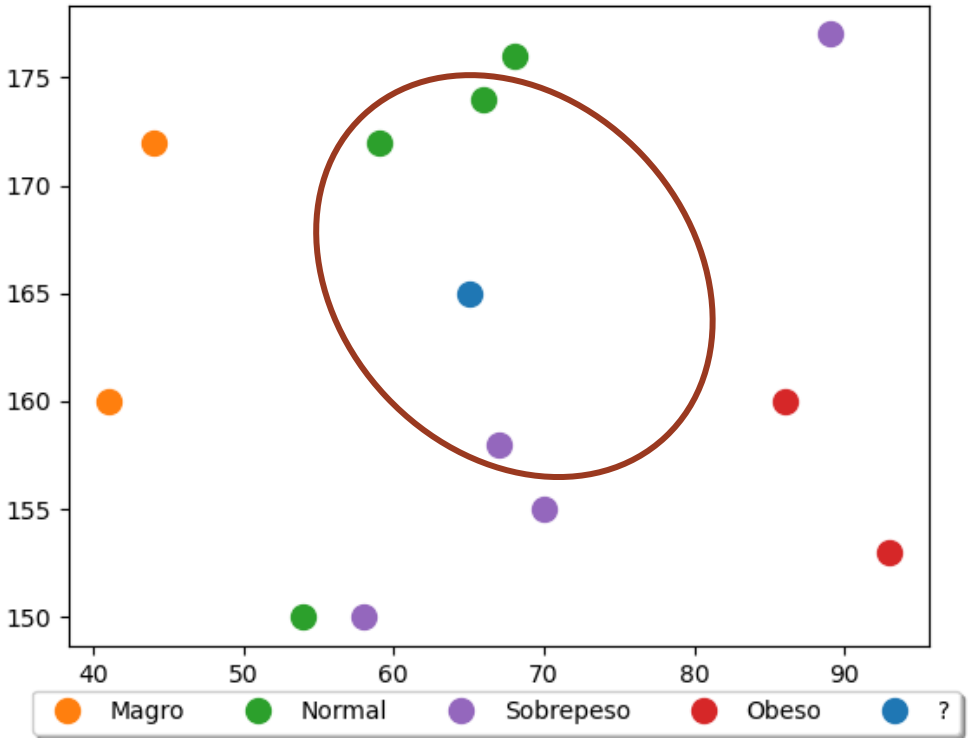
- In general, in practice, we can choose $\boldsymbol{K = \sqrt{n}}$, where *n* is the number of training data samples.

- Try to keep the value of K odd to avoid “tie” between two classes

In the example

$$\sqrt{12} = 3,46 \approx 3$$

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



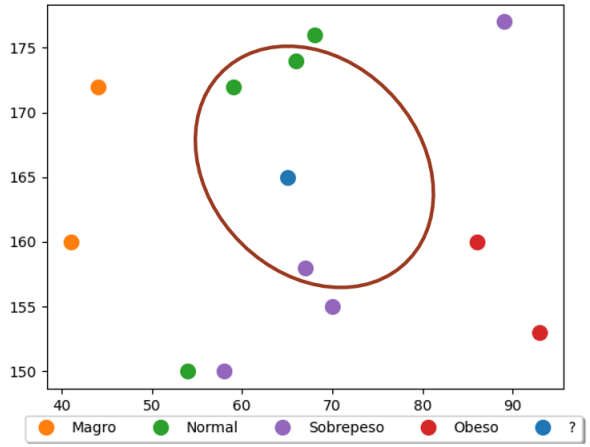
Label : Normal

KNN in Python

K=3

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

Label: Normal



Code:

```
import pandas as pd
data = pd.read_csv ("imc.csv")

x= data.iloc [:,0:2]
y= data.iloc [:,2]

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier ( n_neighbors =3, weights =' distance ', metric =" euclidean ")
knn.fit (x, y)

print(" prediction (65,165): ", knn.predict ([[65, 165]]))
```

Output:

```
prediction (65,165): ['Normal']
```

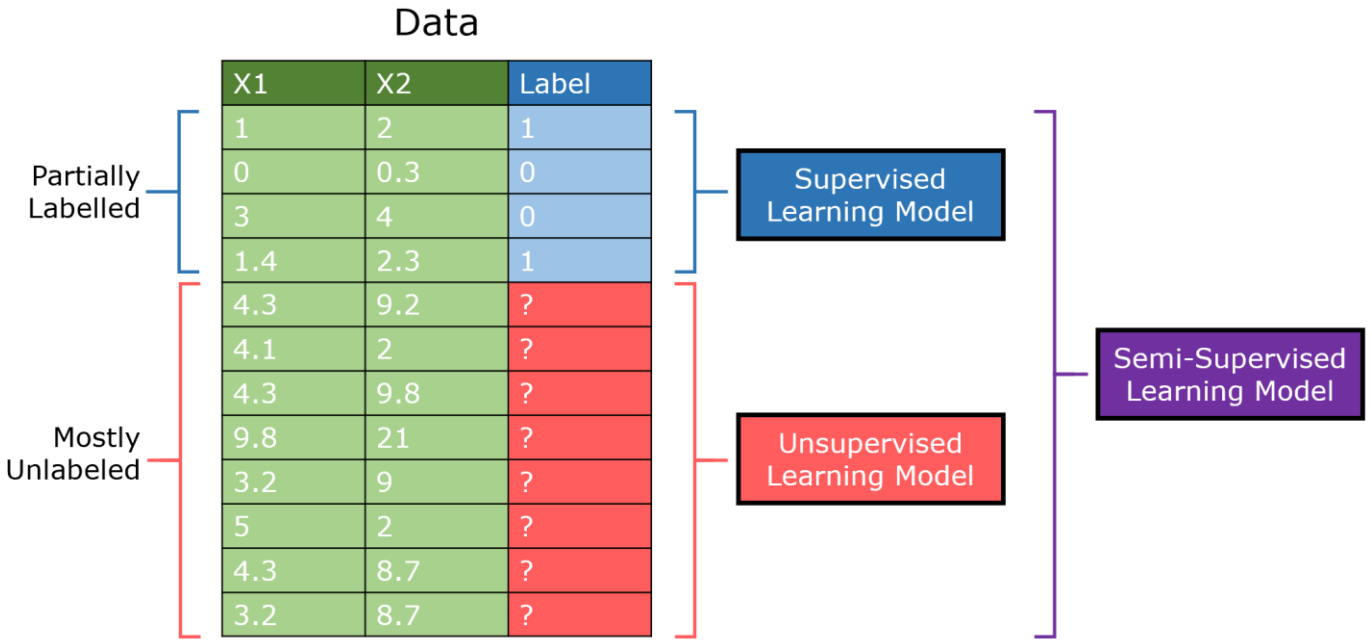
Advantages and disadvantages of using K Nearest Neighbors

Benefits	Disadvantages
• simple to implement • Flexible • Handles naturally with *multiclass* • Can perform well in practice with enough data	• It is necessary to determine the value of K • The computation cost is very high because we need to calculate the distance from each new instance to all training instances • Data storage • We must ensure that we use a meaningful distance (similarity) function.

Semi-supervised

Semisupervised learning

- *Labels* are “expensive”
- We can have few examples with *label* and many without *label*
    - Eg: Fraudulent activities



Framework

1. The algorithm uses for training a limited portion of the *dataset containing labeled*
    1. Result: “partially trained” model

2. This model will classify the portion of the *dataset* that does not contain *labels*
    1. Result: “ pseudo-classified ”

3. Join the two portions of examples
    1. It allows to combine aspects of descriptive and predictive learning

In theory, you can use any algorithm that is used for *supervised learning*

transfer learning
Transfer Learning

Transfer learning

See attached document “transfer_learning.pdf”

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.