# Introduction to various distance measures for Machine Learning:

Several supervised and unsupervised learning algorithms, such as K-Nearest Neighbours, K-means clustering, Self-Organizing Maps, and Support Vector Machines, rely on distance measures as their foundation.  
  
Our machine learning outcomes are impacted by the distance measure we use, therefore it's critical to consider which measure best fits the situation. As a result, we need to exercise caution while selecting our measures. However, before we can decide, we must comprehend the various distance metrics available to us and how they operate.

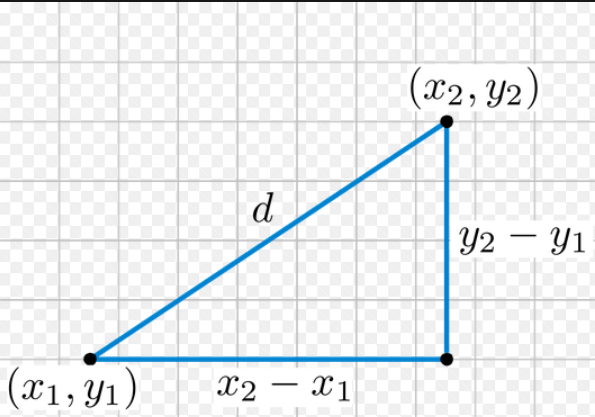
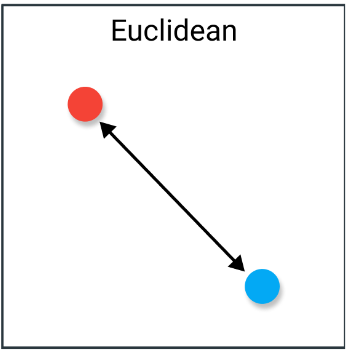
Distance measures calculate the difference between two objects in a problem space, such as features in a data set. The calculated distance can then determine the similarity of the features - the smaller the distance, the more similar the features.

# Euclidean distance:

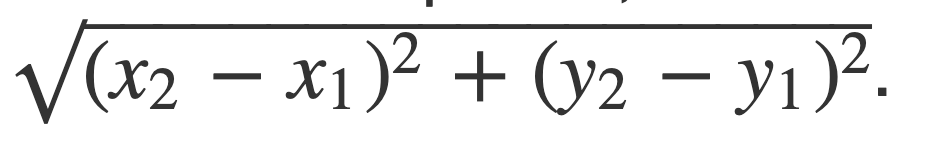
Euclidean distance is the shortest distance between any two points in a metric space.

Given its intuitive usability, straightforward implementation, and reliable performance across multiple scenarios, it has become the most popular distance measure and the default choice for many applications.

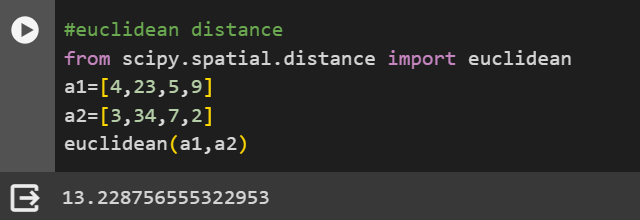
The Euclidean distance can also be referred to as a L2-norm regularization (Ridge regression) .

Formula: If the points (x1, y1) and (x2, y2) are in 2-dimensional space, then the Euclidean distance between them is



Let’s see how we can do this in Scipy:

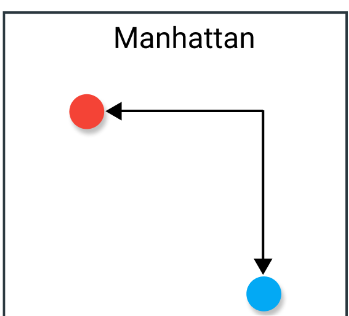
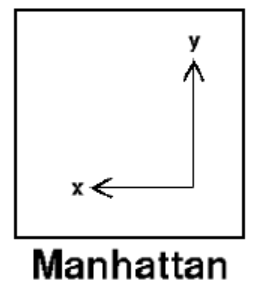


# Manhattan distance:

Manhattan Distance is the sum of absolute differences between points across all the dimensions.

In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates.

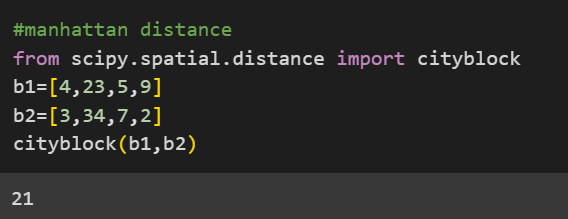
The Manhattan distance can also be referred to as a L1-norm regularization (Lasso regression) .

**Formula**: In a plane with p1 at (x1, y1) and p2 at (x2, y2)

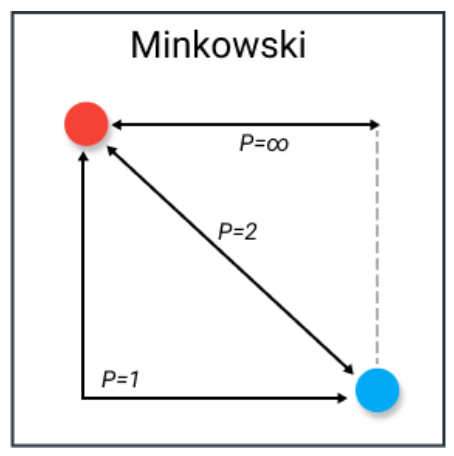
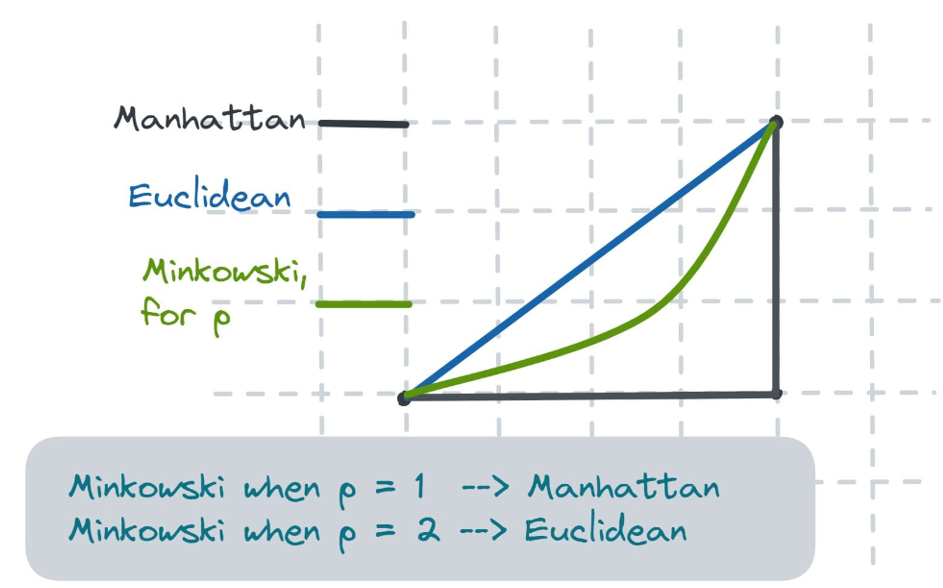


Let’s see how we can do this in Scipy:

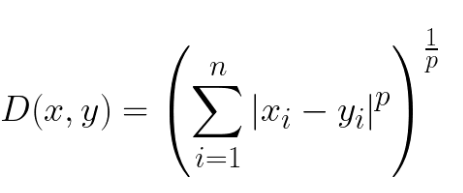


# Minkowski distance:

This distance is a generalization that combines aspects of both the Euclidean distance and the Manhattan distance and it can be used to calculate distance for any order or degree. The Minkowski distance formula is a weighted combination of the absolute differences between the elements in two vectors.

**Formula:** The Minkowski distance of order p between two points is defined as



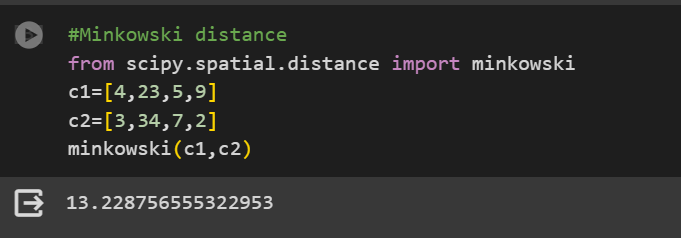
What makes this distance measure especially fascinating is how parameter p is utilized.

By adjusting this parameter, we can modify the distance metrics to closely resemble others.

Common values of p are:

* p=1 — Manhattan distance
* p=2 — Euclidean distance

Let’s see how we can do this in Scipy:

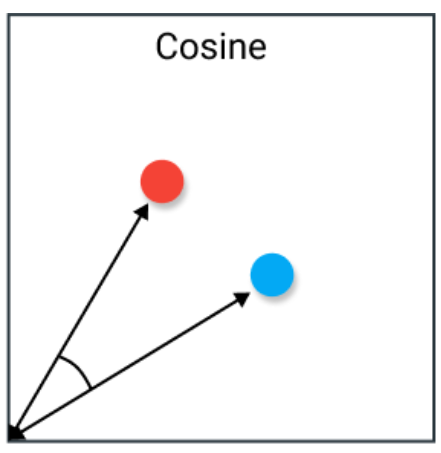
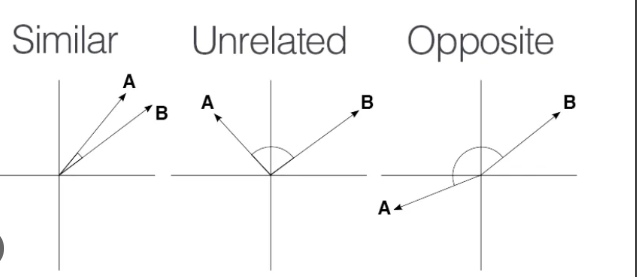


# Cosine Similarity:

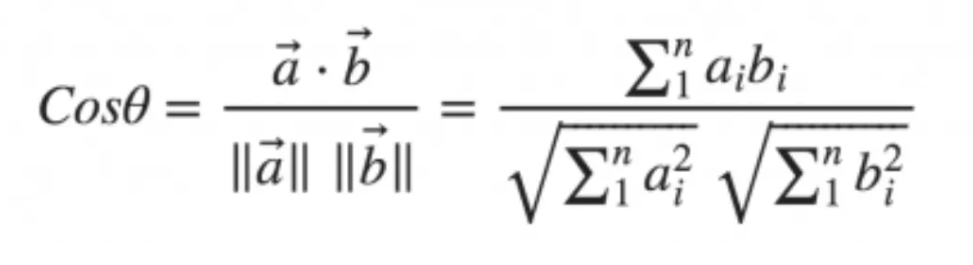
When it comes to measuring similarity between vectors, cosine similarity is a widely used method.

The cosine similarity is particularly valuable in natural language processing applications as it calculates the angle between two vectors, making it ideal for text data analysis.

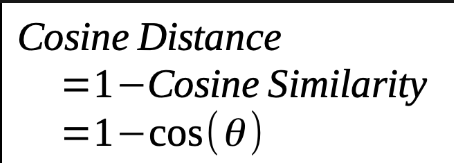
It measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

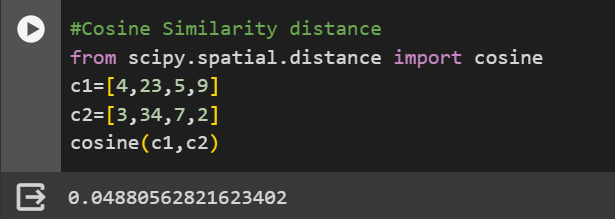
Cosine Similarity Formula:



Cosine Distance Formula:

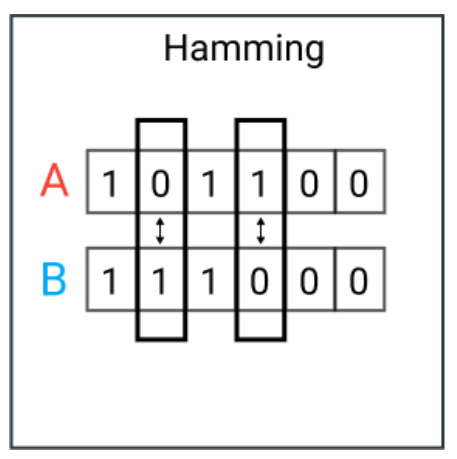


Let’s see how we can do this in Scipy:

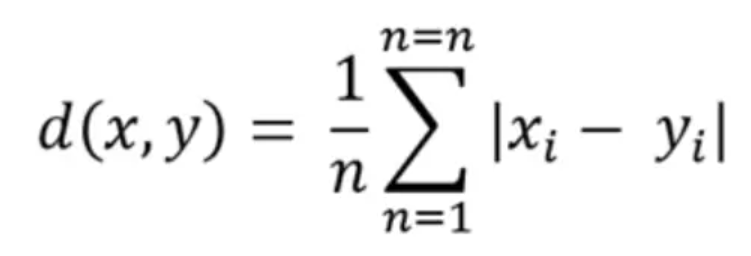


## Hamming Distance:

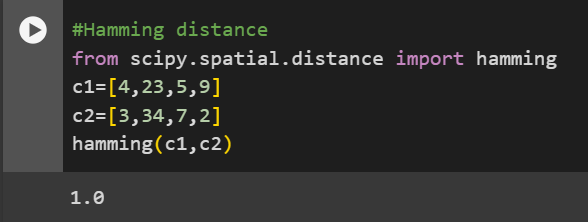
Hamming distance is used to measure the difference between two binary vectors, and it counts the number of positions at which the corresponding bits are different. This measure is commonly used in error-correcting codes, network coding, and DNA sequencing. It can also be used to compare the similarity of strings by computing the number of characters that differ from each other.



Formula:



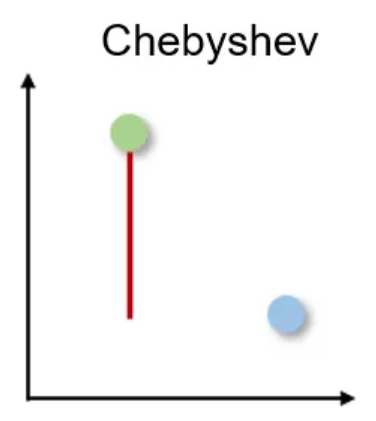
Let’s see how we can do this in Scipy:



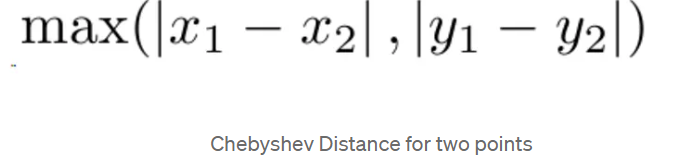
## Chebyshev Distance:

Chebyshev distance is a measure of the maximum difference between the elements in two vectors. It is useful when dealing with high-dimensional data or in optimization problems.

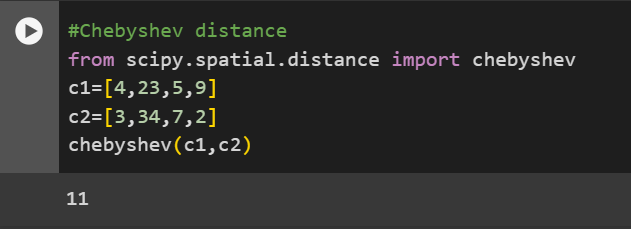
The distance measure is often used in warehouse logistics in which the longest path determines the time it takes to get from one point to the next.



Formula:



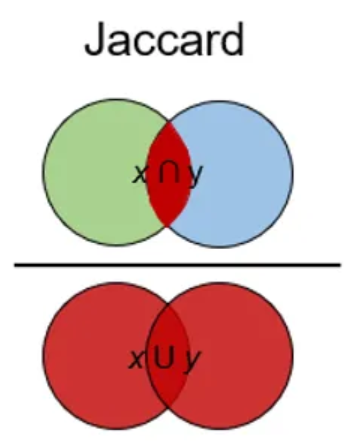
Let’s see how we can do this in Scipy:



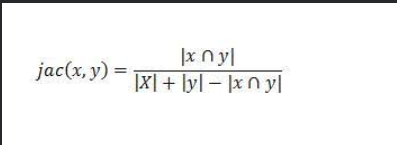
## Jaccard Similarity:

Jaccard similarity is a measure of similarity between two sets. It calculates the size of the intersection between two sets divided by the size of their union. This measure is useful in text classification or recommendation systems.

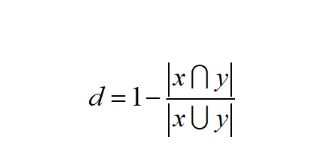
It is often used for binary data to compare the prediction of a deep learning model for image recognition with labelled data or to compare text patterns in documents based on the overlap of words.



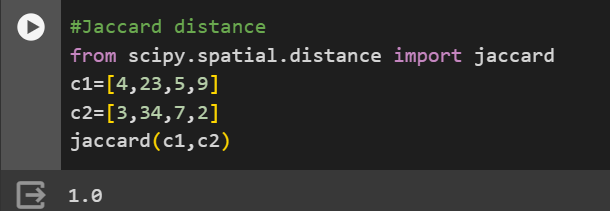
Jaccard Similarity Formula:



Jaccard Distance Formula:



Let’s see how we can do this in Scipy:



A quick comparison of the various distance measures discussed so far:

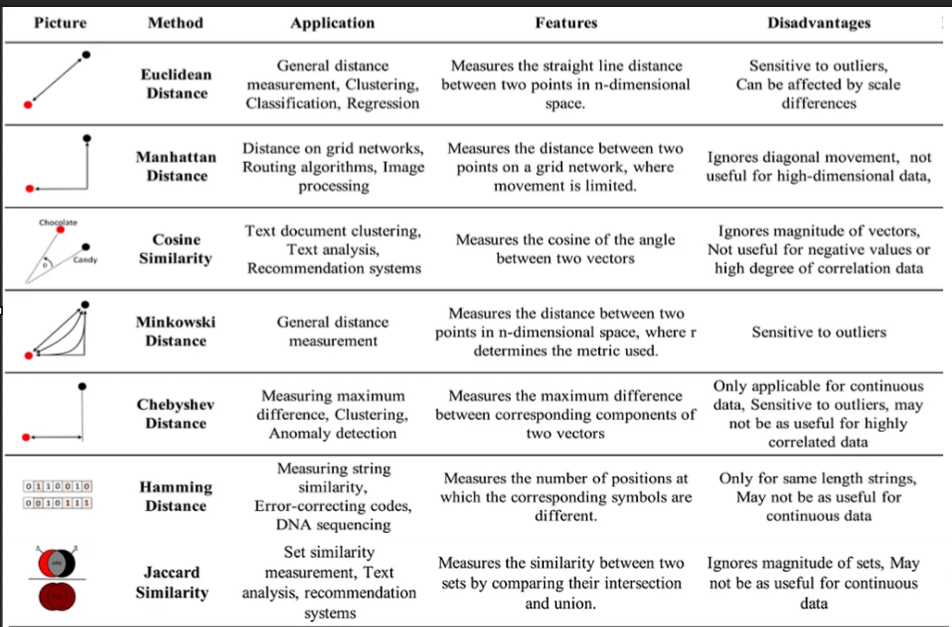


Image Credit: [Mahmoud Harmouch](https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681)