Get unlimited access Open in app

Published in Towards Data Science

Mahmoud Harmouch Follow

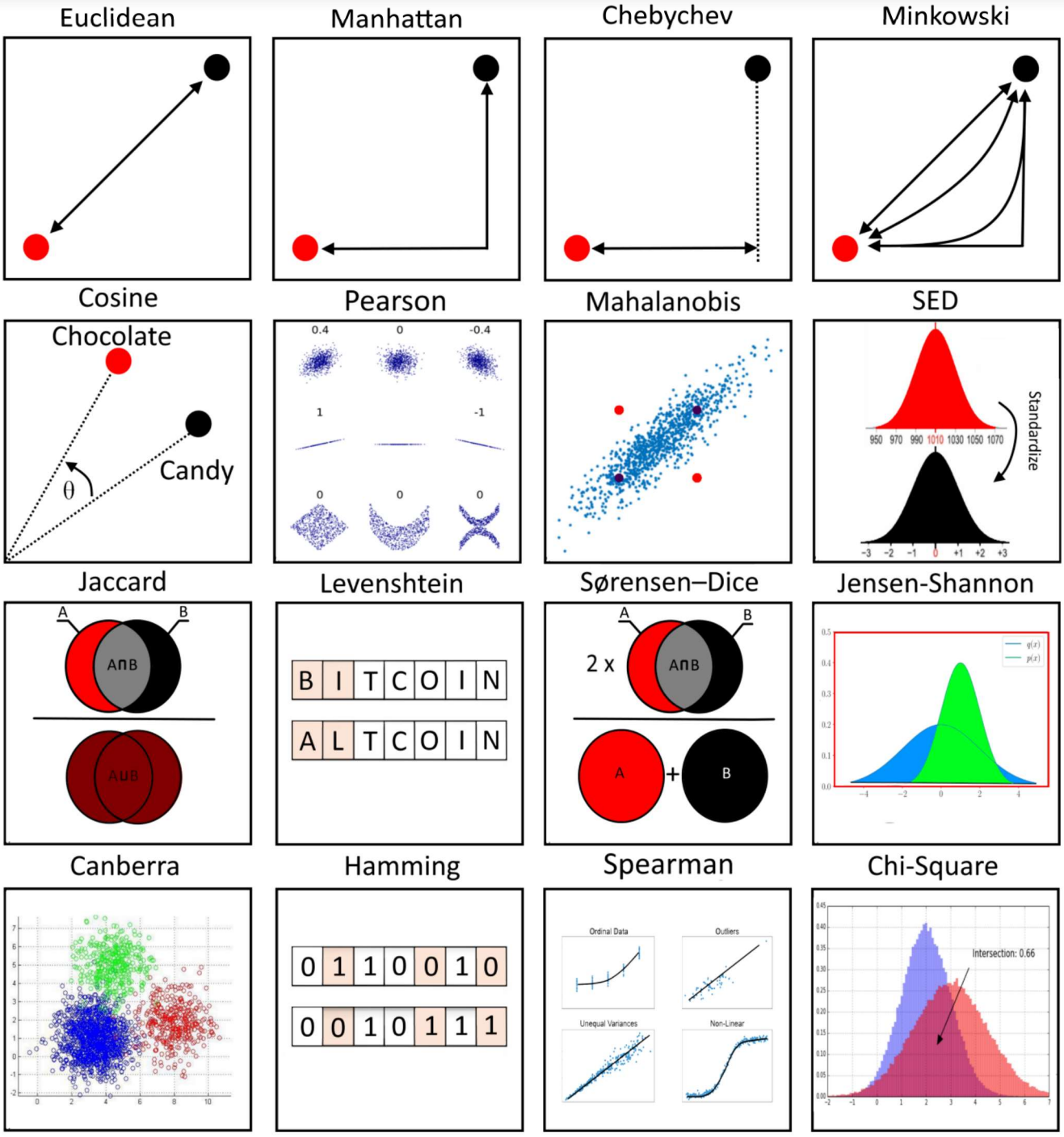
Mar 13, 2021 · 25 min read · Listen

Save

17 types of similarity and dissimilarity measures used in data science.

The following article explains various methods for computing distances and showing their instances in our daily lives. Additionally, it will introduce you to the pydist2 package.

Get unlimited access Open in app



Various ML metrics. Inspired by Maarten Grootendorst.

“There is no Royal Road to Geometry.” — Euclid

> ***Quick note**: Everything written and visualized has been created by the author unless it was specified. Illustrations and equations were generated using tools like Matplotlib, Tex, Scipy, Numpy and edited using GIMP.*

Get unlimited access Open in app

In data science, the similarity measure is a way of measuring how data samples are related or closed to each other. On the other hand, the dissimilarity measure is to tell how much the data objects are distinct. Moreover, these terms are often used in clustering when similar data samples are grouped into one cluster. All other data samples are grouped into different ones. It is also used in classification(e.g. KNN), where the data objects are labeled based on the features' similarity. Another example is when we talk about dissimilar outliers compared to other data samples(e.g., anomaly detection).

The similarity measure is usually expressed as a numerical value: It gets higher when the data samples are more alike. It is often expressed as a number between zero and one by conversion: zero means low similarity(the data objects are dissimilar). One means high similarity(the data objects are very similar).

Let's take an example where each data point contains only one input feature. This can be considered the simplest example to show the dissimilarity between three data points A, B, and C. Each data sample can have a single value on one axis(because we only have one input feature); let's denote that as the x-axis. Let's take two points, A(0.5), B(1), and C(30). As you can tell, A and B are close enough to each other in contrast to C. Thus, the similarity between A and B is higher than A and C or B and C. In other terms, A and B have a strong correlation. Therefore, the smaller the distance is, the larger the similarity will get.

- **Metric:**

A given distance(e.g. dissimilarity) is meant to be a metric if and only if it satisfies the following four conditions:

1- Non-negativity: $d(p, q) \geq 0$, for any two distinct observations p and q.

2- Symmetry: $d(p, q) = d(q, p)$ for all p and q.

3- Triangle Inequality: $d(p, q) \leq d(p, r) + d(r, q)$ for all p, q, r.

4- $d(p, q) = 0$ only if $p = q$.

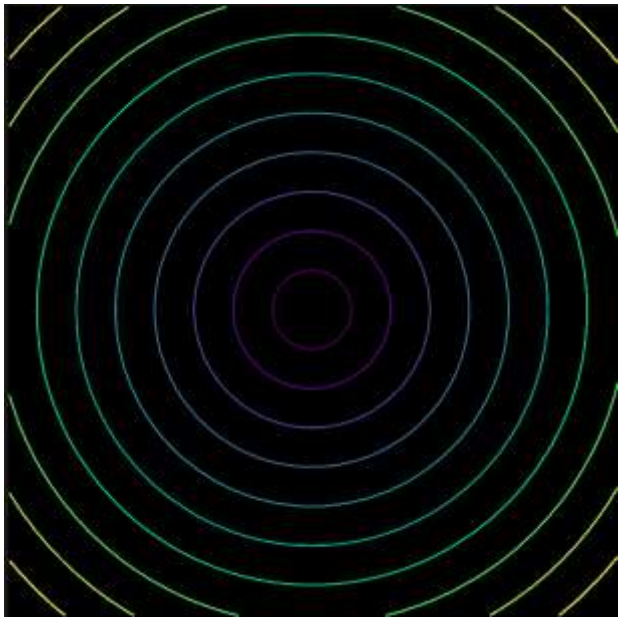
Get unlimited access Open in app

performance of the classifier. Therefore, the way you compute distances between the objects will play a crucial role in the classifier algorithm's performance.

🅱. Distance Functions:

The technique used to measure distances depends on a particular situation you are working on. For instance, in some areas, the euclidean distance can be optimal and useful for computing distances. Other applications require a more sophisticated approach for calculating distances between points or observations like the cosine distance. The following enumerated list represents various methods of computing distances between each pair of data points.

⓪. L2 norm, Euclidean distance.



Euclidean Contours.

The most common distance function used for numeric attributes or features is the Euclidean distance which is defined in the following formula:

Get unlimited access Open in app

$$d(P,Q) = ||P-Q||_0 = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$
$$= \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \ldots + (p_n-q_n)^2}$$

where:

$P = (p_1, p_2, \ldots, p_n)$, and $Q = (q_1, q_2, \ldots, q_n)$

Euclidean distance between two points in n-dimensional space.

As you may know, this distance metric presents well-known properties, like symmetrical, differentiable, convex, spherical…

In 2-dimensional space, the previous formula can be expressed as:

$$d(P,Q) = ||P-Q||_0 = \sqrt{\sum_{i=1}^{2}(p_i - q_i)^2}$$
$$= \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2}$$

where:

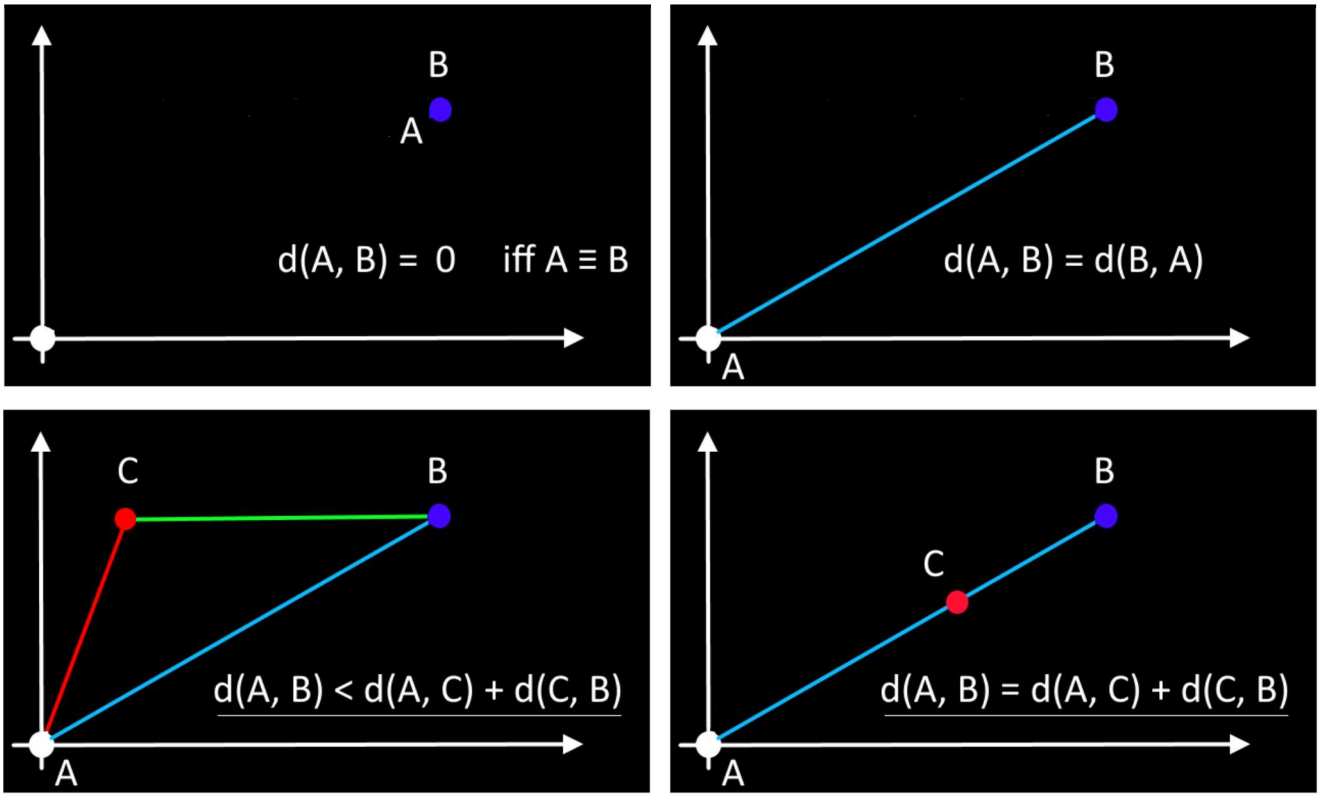
$P = (p_1, p_2)$, and $Q = (q_1, q_2)$

Euclidean distance between two points in 2-dimensional space.

which is equal to the length of the hypotenuse of a right-angle triangle.

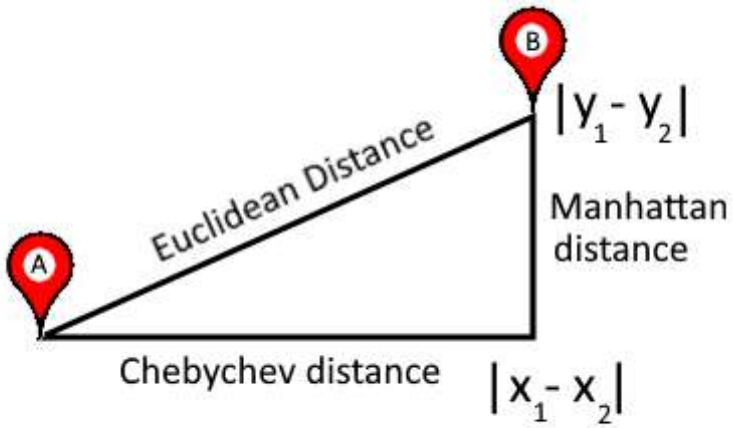
Moreover, the Euclidean distance is a metric because it satisfies its criterion, as the following illustration shows.

Get unlimited access Open in app



The Euclidean distance satisfies all the conditions for being a metric.

Furthermore, the distance calculated using this formula represents the smallest distance between each pair of points. In other words, it is the shortest path to go from point A to point B(2-D Cartesian coordinate system), as the following figure illustrates:



The Euclidean distance is the shortest path(Excluding the case of a wormhole in a quantum world).

Get unlimited access Open in app

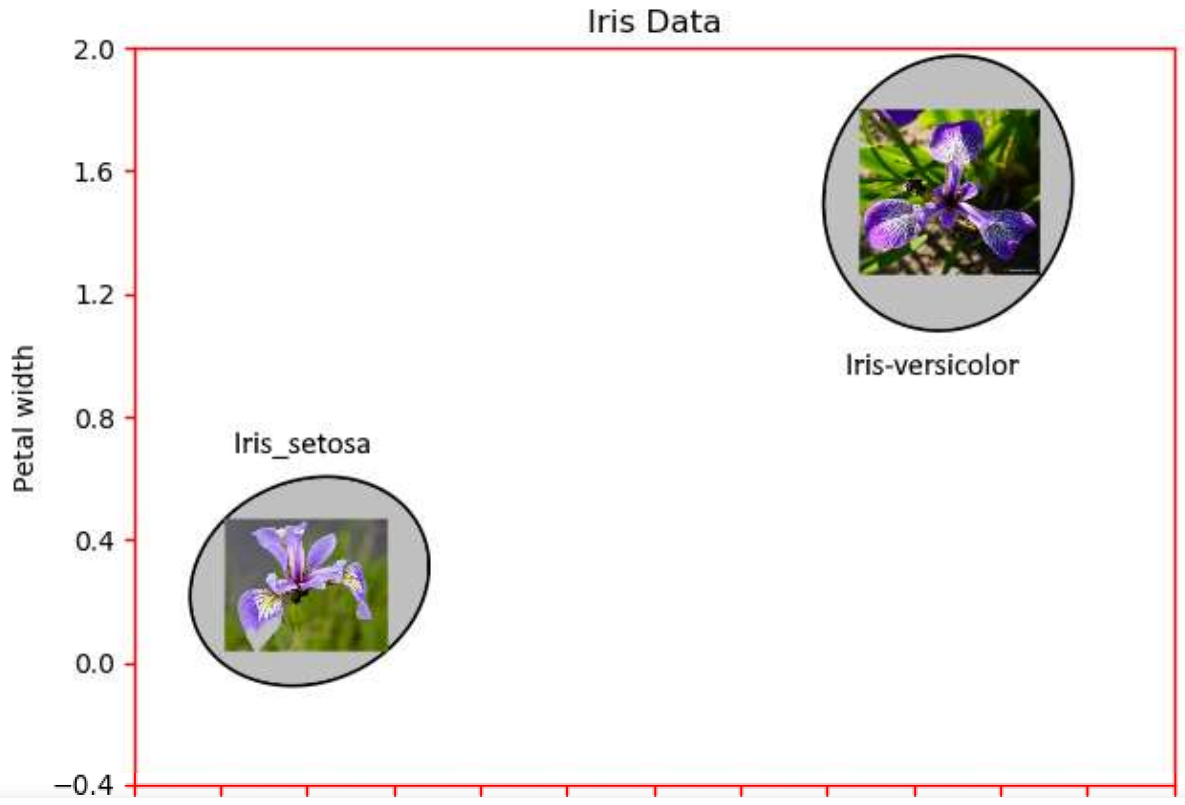
you want to use other metrics like the Manhattan distance, which will be explained later throughout this article.

Another scenario where the euclidean distance fails to give us useful information is in a plane's flight path that follows the earth's curvature, not a straight line(unless the earth is flat, which is not!).

However, before going any further, let's explain how we can use the euclidean distance in the context of machine learning.

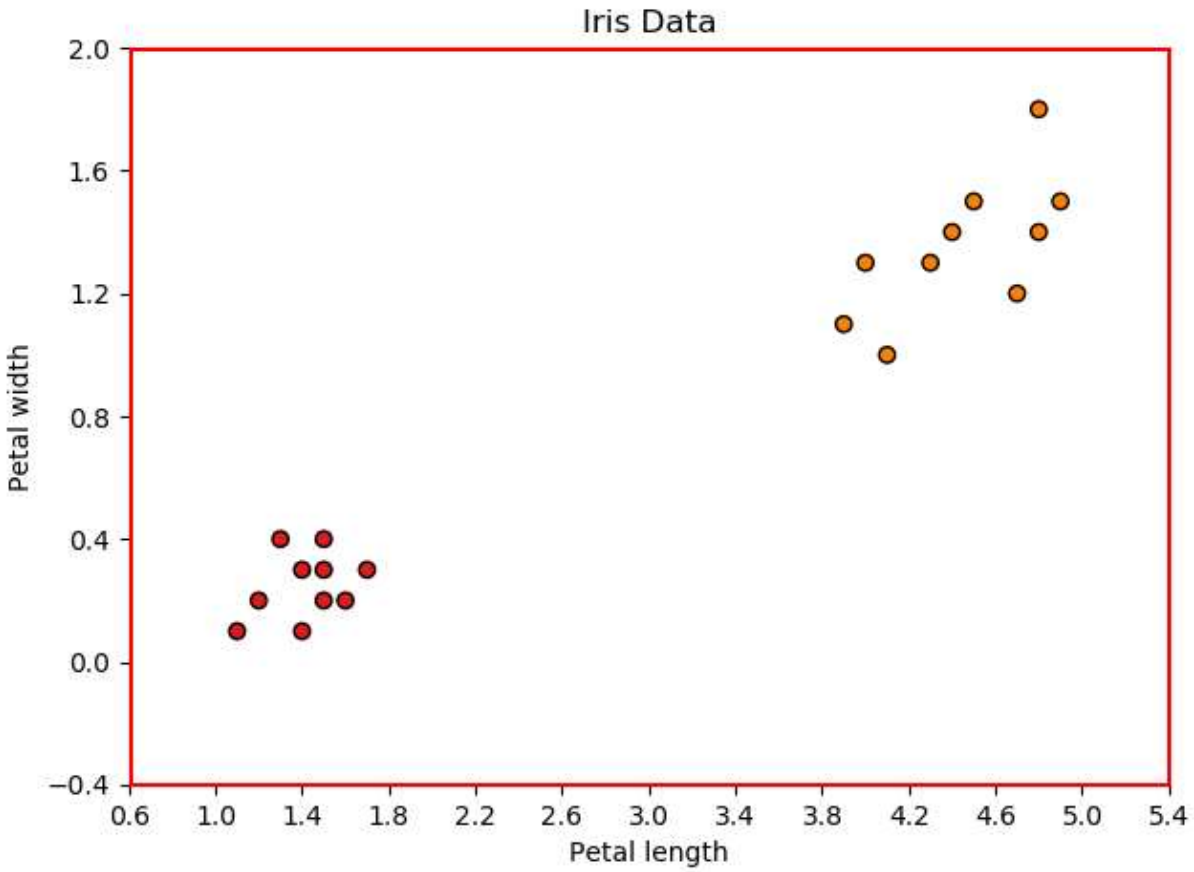
One of the most famous classification algorithms, the KNN algorithm, can benefit from using the euclidean distance to classify data. To demonstrate how the KNN works with the euclidean metric, the popular iris dataset from the Scipy package has been chosen.

As you may know, this dataset contains three kinds of flowers: Iris-Setosa, Iris-Versicolor, and Iris-Virginica, having the following four features: sepal length, sepal width, petal length, petal width. Therefore, we have a 4-dimensional space where each data point can be represented in.



Get unlimited access Open in app

For simplicity and demonstration purposes, let's choose only two features: petal length, petal width, and excluding Iris-virginica data. In this way, we can plot the data points in a 2-D space where the x-axis and the y-axis represent the petal length and the petal width, respectively.

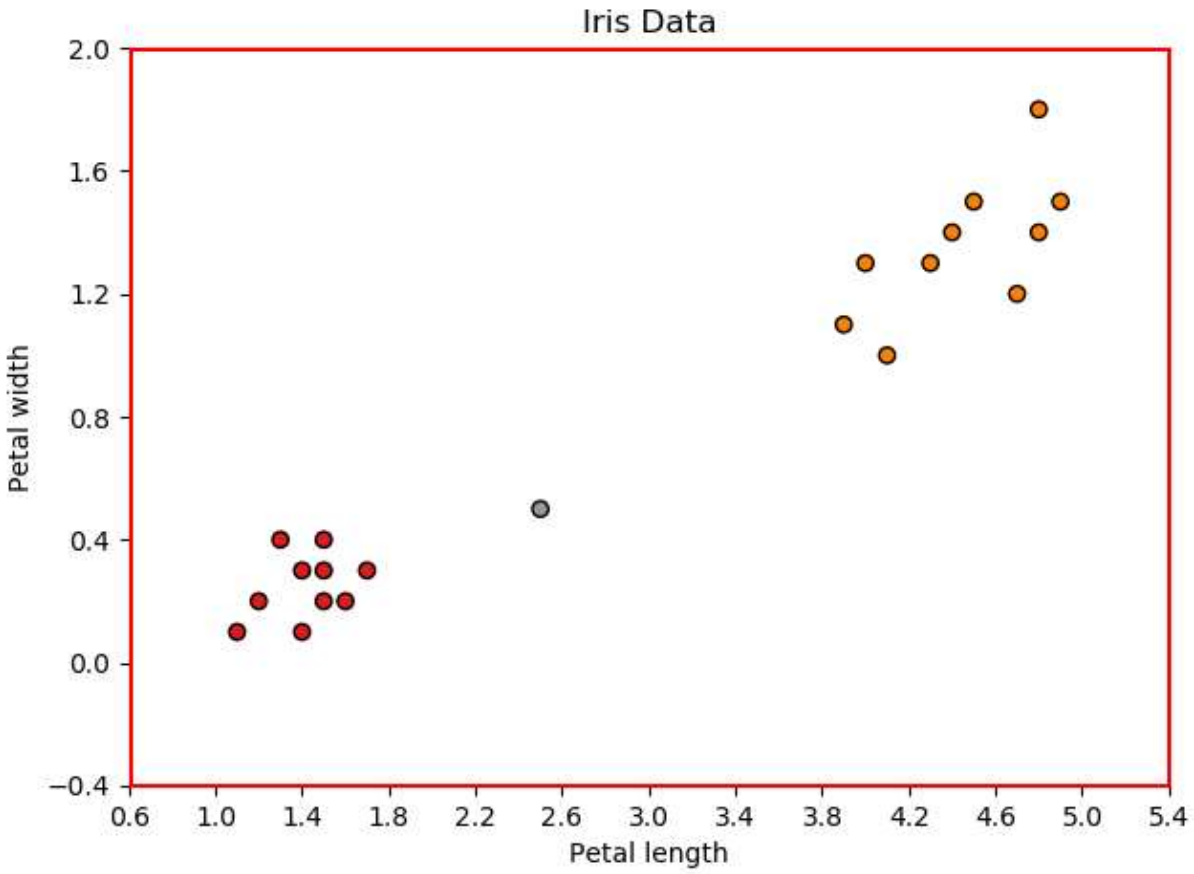


Training dataset.

Each data point came along with its own label: Iris-Setosa or Iris-versicolor(0 and 1 in the dataset). Thus, the dataset can be used in KNN classification because it is a supervised ML algorithm by its nature. Let's assume that our ML model(KNN with k = 4) has been trained on this dataset where we have selected two input features and only twenty data points, as the previous graph shows.

Until this point, everything looks great, and our KNN classifier is ready to classify a new data point. Therefore, we need a way to let the model decide where the new data point can be classified.

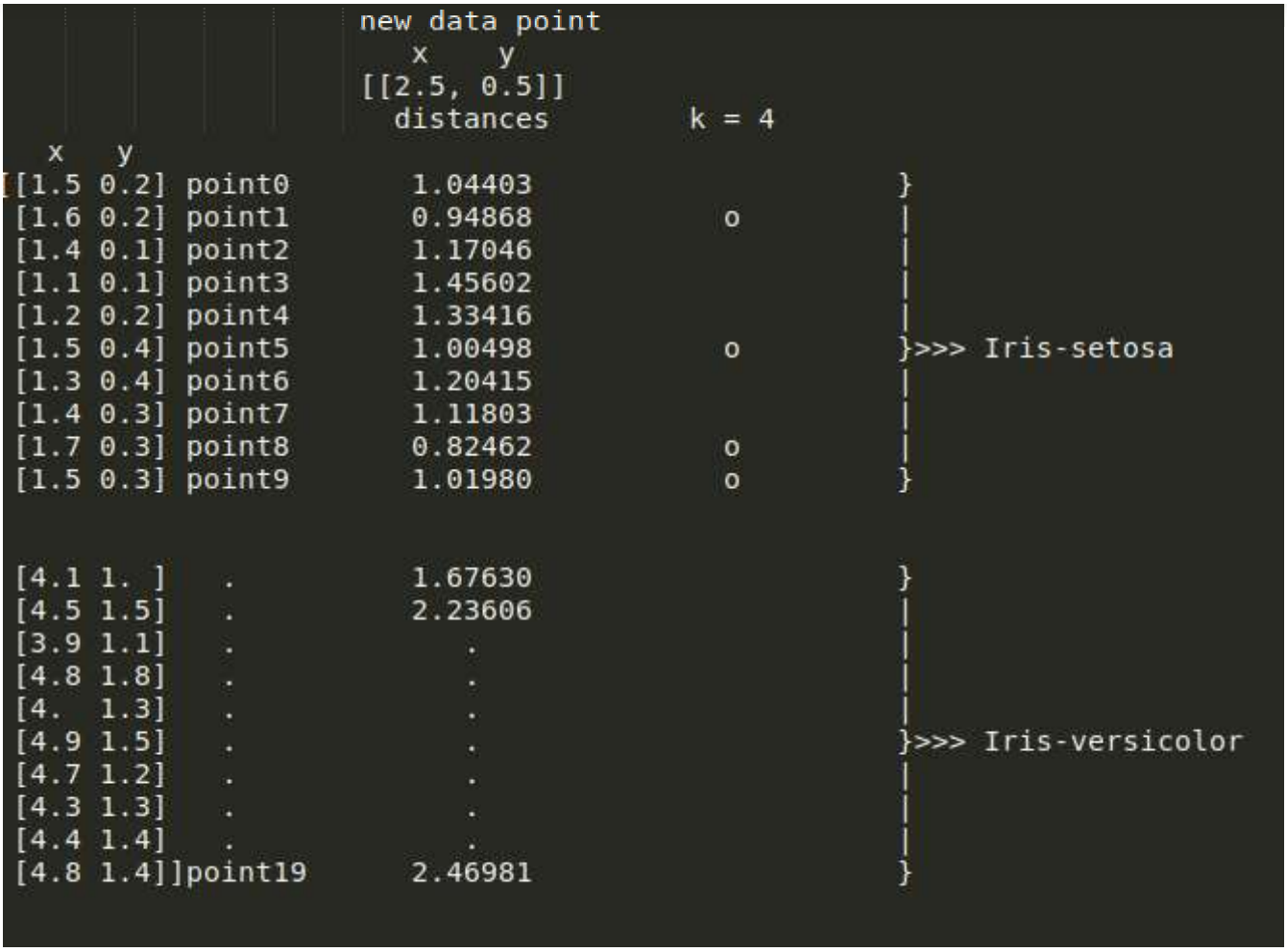
Get unlimited access Open in app



Predict the label for a new data point.

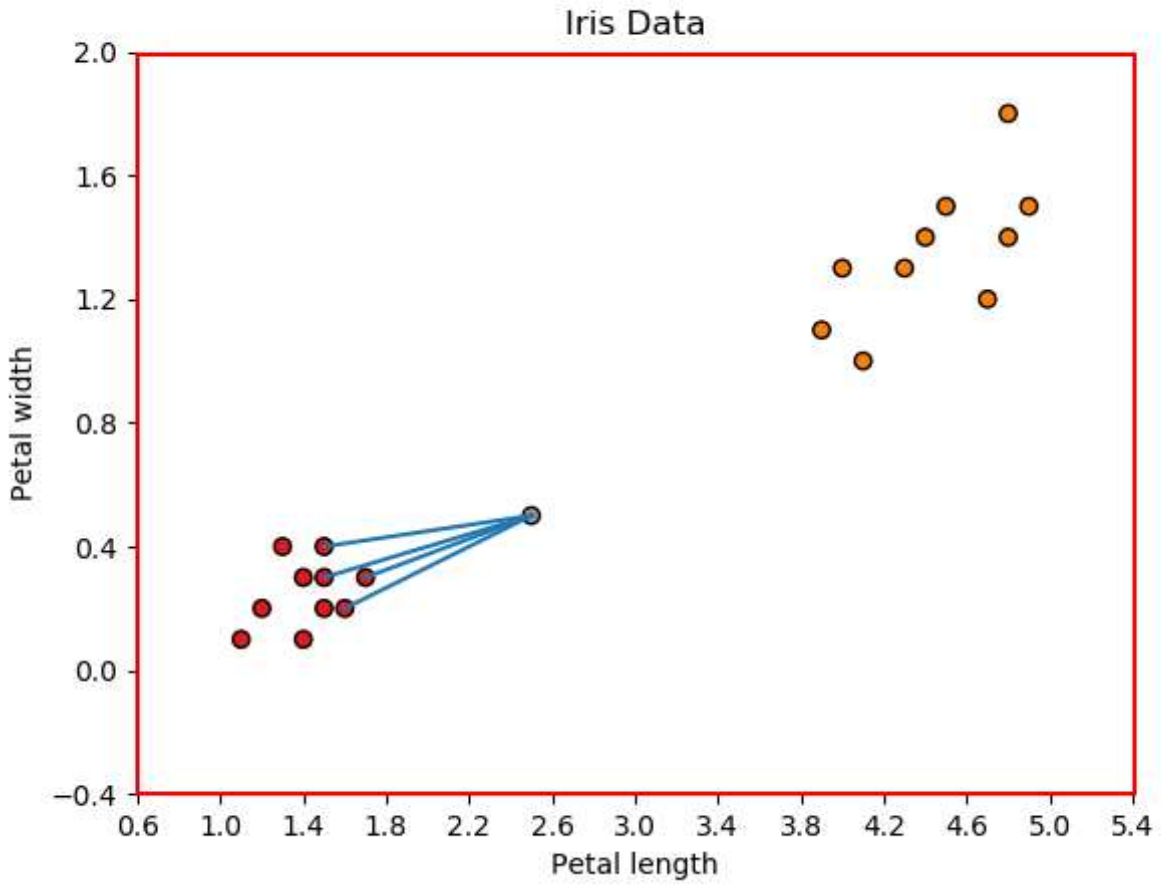
As you may be thinking, the euclidean distance has been chosen to let each trained data point vote where the new data sample can fit: Iris-Setosa or Iris-versicolor. Thus, the euclidean distance has been calculated from the new data point to each point of our training data, as the following figure shows:

Get unlimited access Open in app



With k = 4, The KNN classifier requires to chose the smallest four distances, which represents the distance from the new point to the following points: point1, point5, point8, and point9 as the graph shows:

Get unlimited access Open in app



Four neighbors voted for Iris-Setosa.

Therefore, the new data sample has been classified as Iris-Setosa. Using this analogy, you can imagine higher dimensions and other classifiers. Hopefully, You get the idea!

As stated earlier, each domain requires a specific way of computing distances. As we progress more throughout this article, you will find out what is meant by stating this.

①. Squared Euclidean distance.

Computing distances using this approach avoids the need to use the squared root function. As the name reflects, the SED is equal to the euclidean distance squared. Therefore, SED can reduce computational work while calculating distances between observations. For instance, it can be used in clustering, classification, image processing, and other domains.

Get unlimited access Open in app

$$d(P,Q) = \sum_{i=1}^{n}(p_i - q_i)^2$$

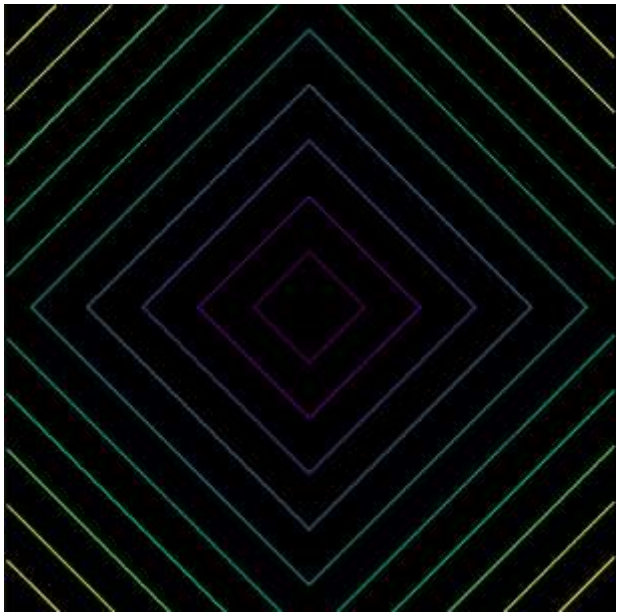
$$= (p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2$$

where:

$$P = (p_1, p_2, \ldots, p_n), \text{ and } Q = (q_1, q_2, \ldots, q_n)$$

Squared Euclidean distance between two points in n-D space.

②. L1 norm, City Block, Manhattan, or taxicab distance.



Manhattan Contours.

This metric is very useful in measuring the distance between two streets in a given city, where the distance can be measured in terms of the number of blocks that separate two different places. For instance, according to the following image, the distance between point A and point B is roughly equal to 4 blocks.

Get unlimited access Open in app

Manhattan distance in real world

This method was created to solve computing the distance between source and destination in a given city where it is nearly impossible to move in a straight line because of the buildings grouped into a grid that blocks the straight pathway. Hence the name City Block.

You could say that the distance between A and B is the euclidean one. But, as you may notice that this distance is not useful. For instance, you need to have a useful distance to estimate the travel time or how long you have to drive. Instead, it would help if you had the shortest path using streets. So it depends on the situation on how you can define and use distance.

In n-dimensional space, the Manhattan distance is expressed as:

$$d(P,Q) = ||P-Q||_1 = \sum_i^n |p_i - q_i|$$

$$= |p_1 - q_1| + |p_2 - q_2| + \ldots + |p_n - q_n|$$

where:

$P = (p_1, p_2, \ldots, p_n)$, and $Q = (q_1, q_2, \ldots, q_n)$

Manhattan distance between two points in n-D space.

Get unlimited access Open in app

$$d(P,Q) = ||P-Q||_1 = \sum_i^2 |p_i - q_i|$$

$$= |p_1 - q_1| + |p_2 - q_2|$$

where:

$P = (p_1, p_2)$, and $Q = (q_1, q_2)$

Manhattan distance between two points in 2-D space.

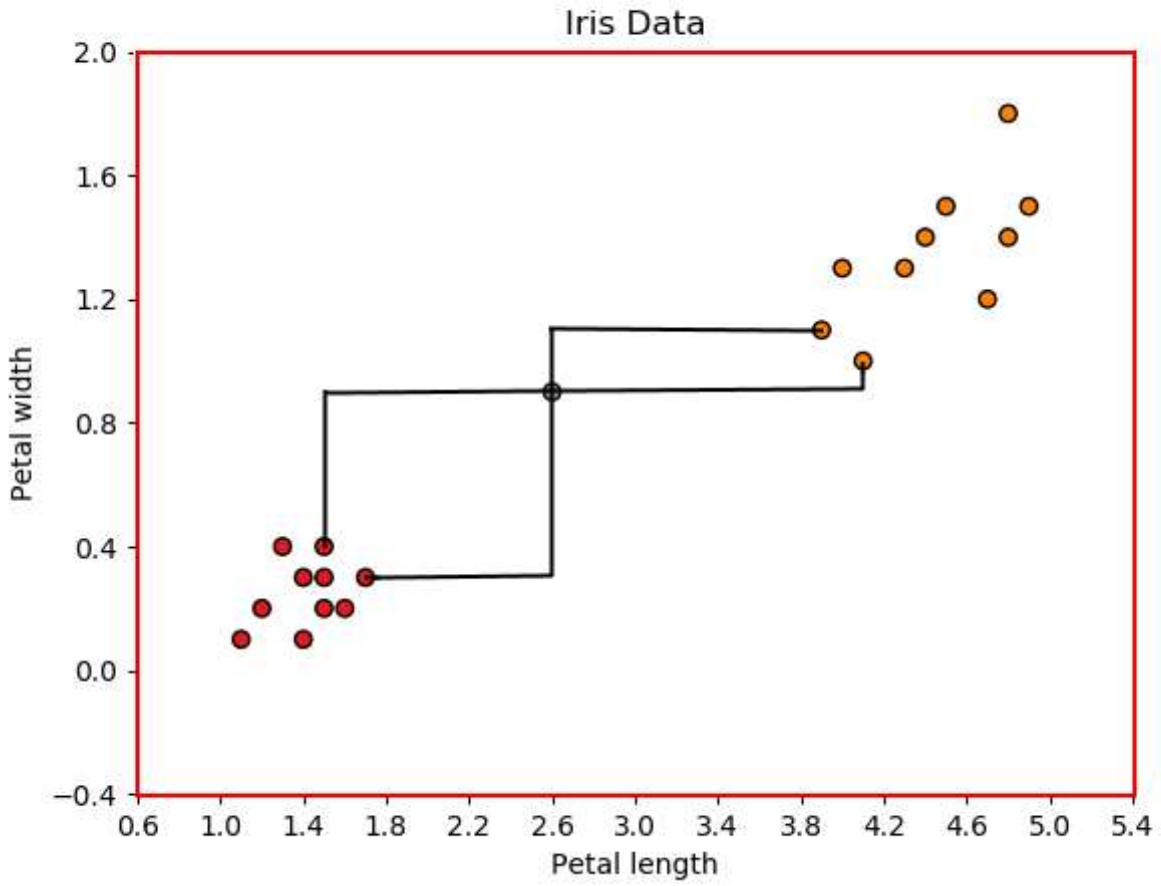
Recall from the previous KNN example, Computing the manhattan distance from a new data point to the training data will produce the following values:

```
                    new data point
                       x    y
                    [[2.6, 0.9]]
                      distances        k = 4
   x   y
[[1.5 0.2] point0       1.8                    }
 [1.6 0.2] point1       1.7                    |
 [1.4 0.1] point2       2.0                    |
 [1.1 0.1] point3       2.3                    |
 [1.2 0.2] point4       2.1                    |
 [1.5 0.4] point5       1.6             o      }>>> Iris-setosa
 [1.3 0.4] point6       1.8                    |
 [1.4 0.3] point7       1.8                    |
 [1.7 0.3] point8       1.5             o      |
 [1.5 0.3] point9       1.7                    }


 [4.1 1. ] point10      1.6             o      }
 [4.5 1.5] point11      2.5                    |
 [3.9 1.1] point12      1.5             o      |
 [4.8 1.8] point13      3.1                    |
 [4.   1.3] point14     1.8                    |
 [4.9 1.5] point15      2.9                    }>>> Iris-versicolor
 [4.7 1.2] point16      2.4                    |
 [4.3 1.3] point17      2.1                    |
 [4.4 1.4] point18      2.3                    |
 [4.8 1.4]]point19      2.7                    }
```

KNN classification using Manhattan distance(tie!)

Get unlimited access Open in app

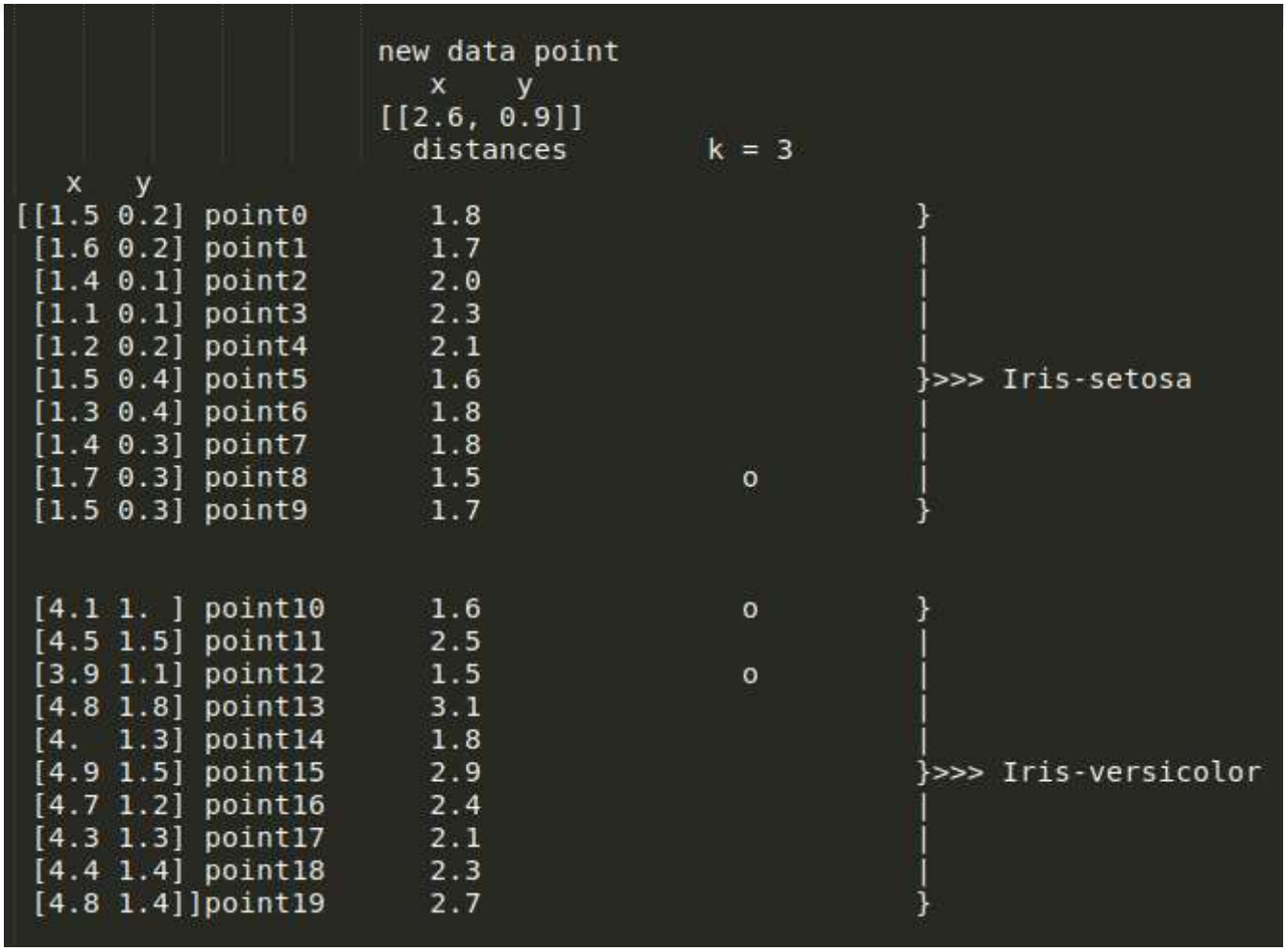


Manhattan distance: a tie!

I think you may have encountered this problem somewhere. An intuitive solution would be changing the value of k, decreasing by one if k is larger than one, or increasing by one, otherwise.

However, you will get different behavior for the KNN classifier for each of the previous solutions. For instance, in our example, k=4. Changing it to k=3 would result in the following values:

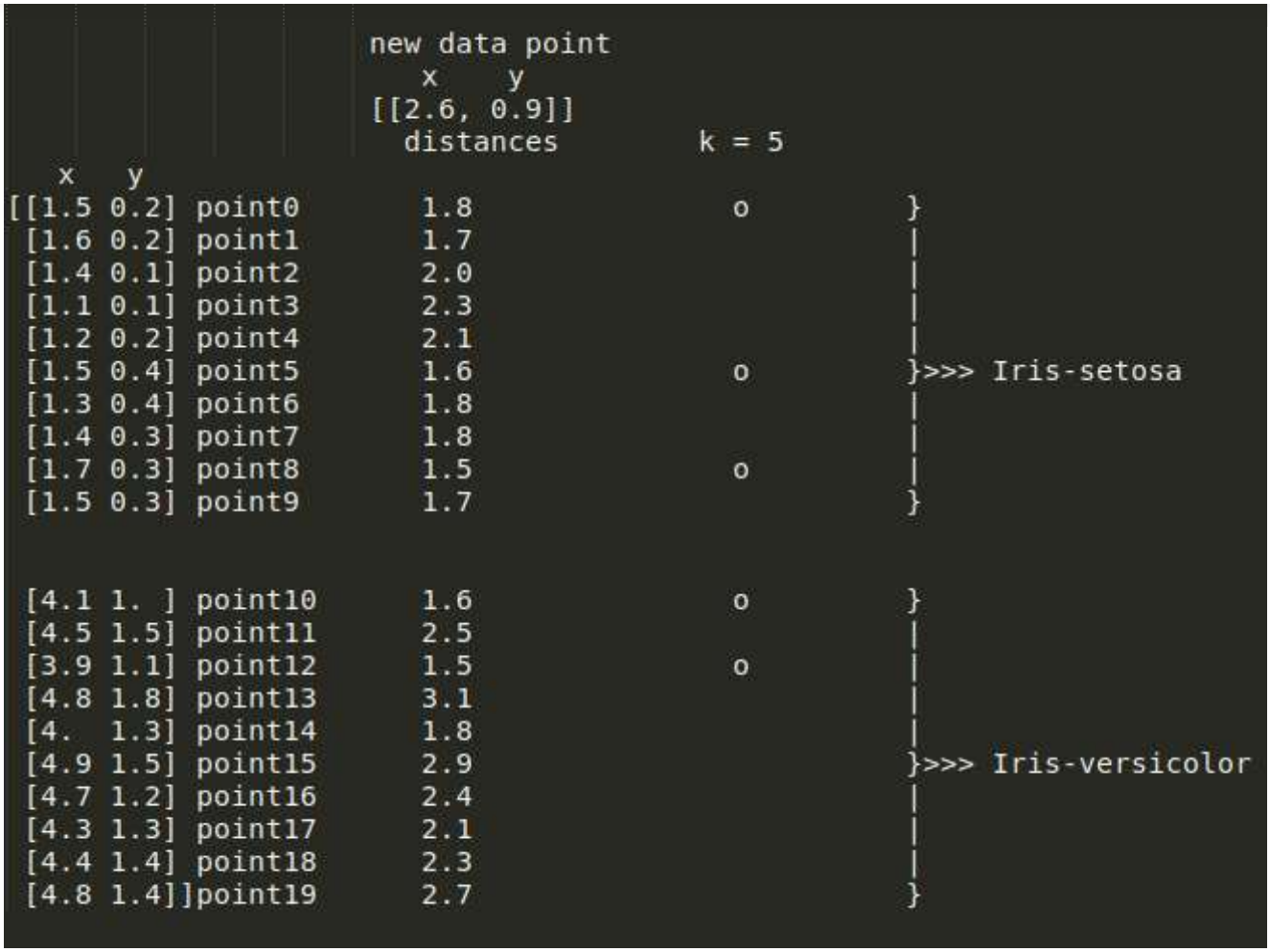
Get unlimited access Open in app



Decreasing k by one.

And the flower has been classified as Iris-versicolor. In the same manner, Changing it to k=5 would result in the following values:

Get unlimited access Open in app

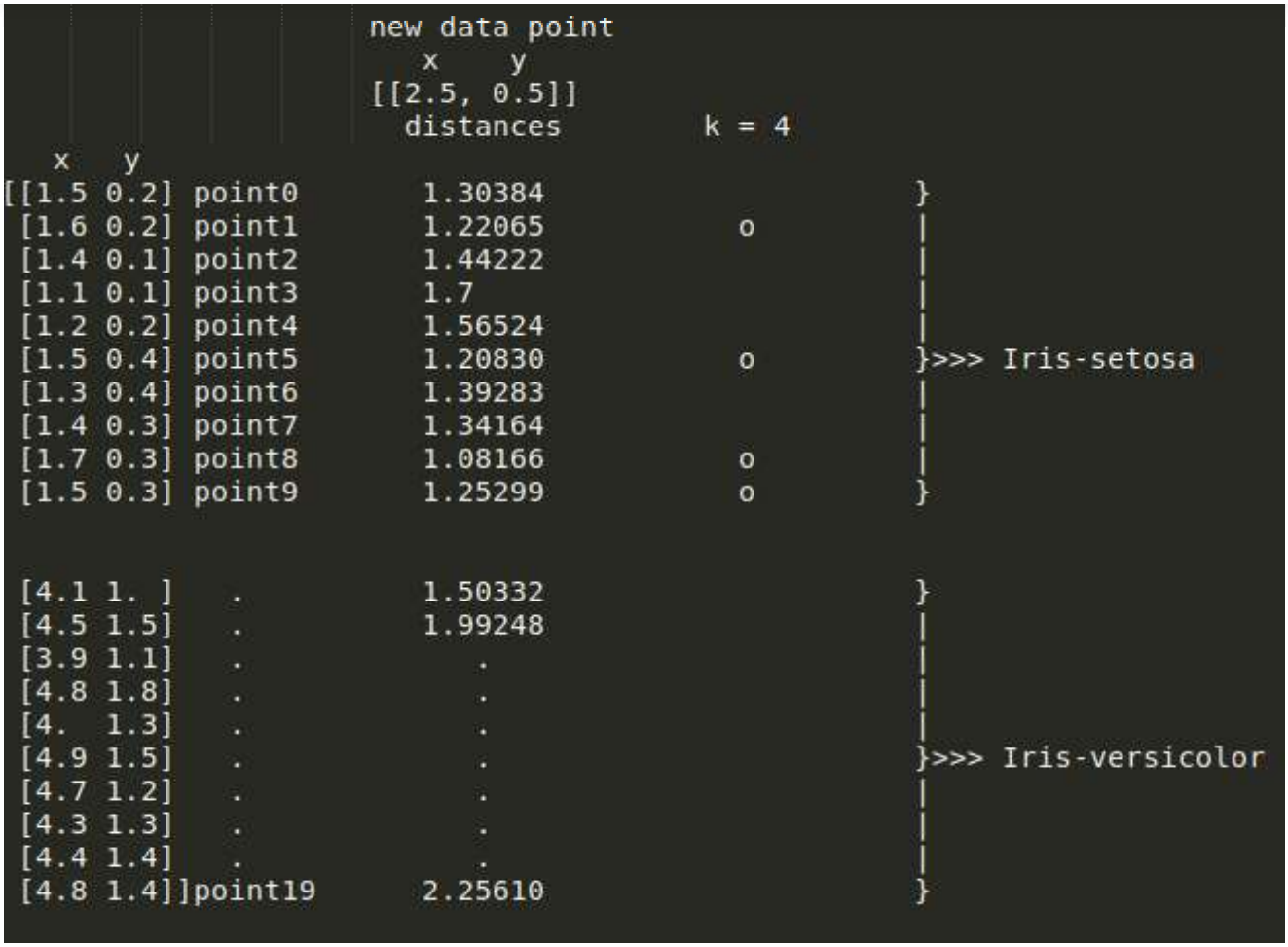


Increasing k by one.

And the flower is classified as Iris-Setosa. So, it is up to you to decide whether you need to increase or decrease the value of k.

However, someone would argue that you can change the metric of measure if it is not a constraint for the problem. For example, computing the euclidean distance would solve this:

Get unlimited access Open in app



Changing the distance metric would also break the tie.

And the flower is strongly classified as Iris-Setosa.

In my opinion, if you don't have to change the Manhattan distance and use the same value for k, adding a new dimension or feature, if available, would also break the tie. For instance, adding the sepal width as a new dimension would lead to the following results:

Get unlimited access Open in app

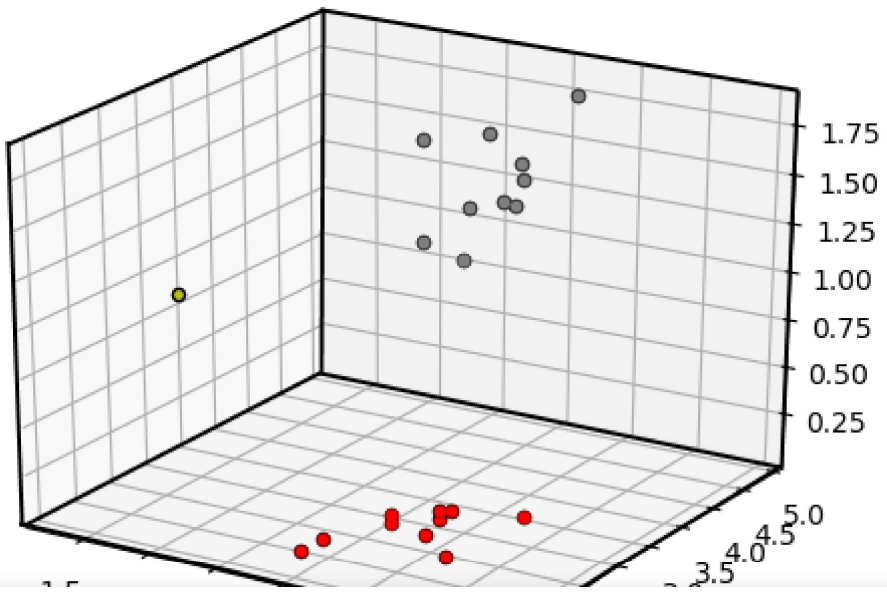
```
                    new data point
                      x    y    z
                    [[1.3, 2.6, 0.9]]
                      distances        k = 4
   x   y   z
[[3.7 1.5 0.2] point0       4.2                    }
 [3.4 1.6 0.2] point1       3.8                    |
 [3.  1.4 0.1] point2       3.8                    |
 [3.  1.1 0.1] point3       4.0                    |
 [4.  1.2 0.2] point4       4.8                    |
 [4.4 1.5 0.4] point5       4.7                    }>>> Iris-setosa
 [3.9 1.3 0.4] point6       4.4                    |
 [3.5 1.4 0.3] point7       4.0                    |
 [3.8 1.7 0.3] point8       4.0                    |
 [3.8 1.5 0.3] point9       4.2                    }

 [2.7 4.1 1. ] point10      3.0              o     }
 [2.2 4.5 1.5] point11      3.4              o     |
 [2.5 3.9 1.1] point12      2.7              o     |
 [3.2 4.8 1.8] point13      5.0                    |
 [2.8 4.  1.3] point14      3.3              o     |
 [2.5 4.9 1.5] point15      4.1                    }>>> Iris-versicolor
 [2.8 4.7 1.2] point16      3.9                    |
 [2.9 4.3 1.3] point17      3.7                    |
 [3.  4.4 1.4] point18      4.0                    |
 [2.8 4.8 1.4]]point19      4.2                    }
```

Adding a new feature to the model.

And the flower is classified as Iris-Versicolor.

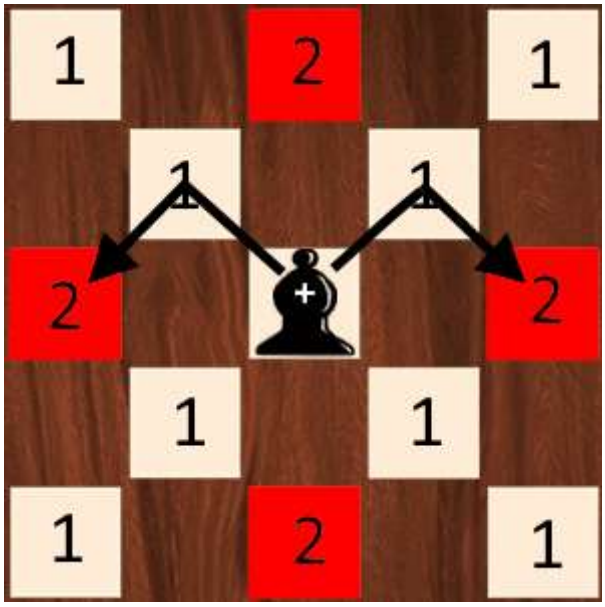
And here goes the plot in 3-D space where x-axis, y-axis, and z-axis represent sepal width, petal length, and petal width, respectively:



Get unlimited access Open in app

Computing Manhattan distance is computationally faster than the previous two methods. As the formula shows, it only requires additions and subtractions, which turns out to be much faster than computing square root and the power of two.

If you have ever played chess, The Manhattan distance is used by bishops in order to move between two horizontal or vertical blocks of the same color:



Bishop uses Manhattan distance(if you don't see it, just imagine it by rotating the chessboard by 45°).

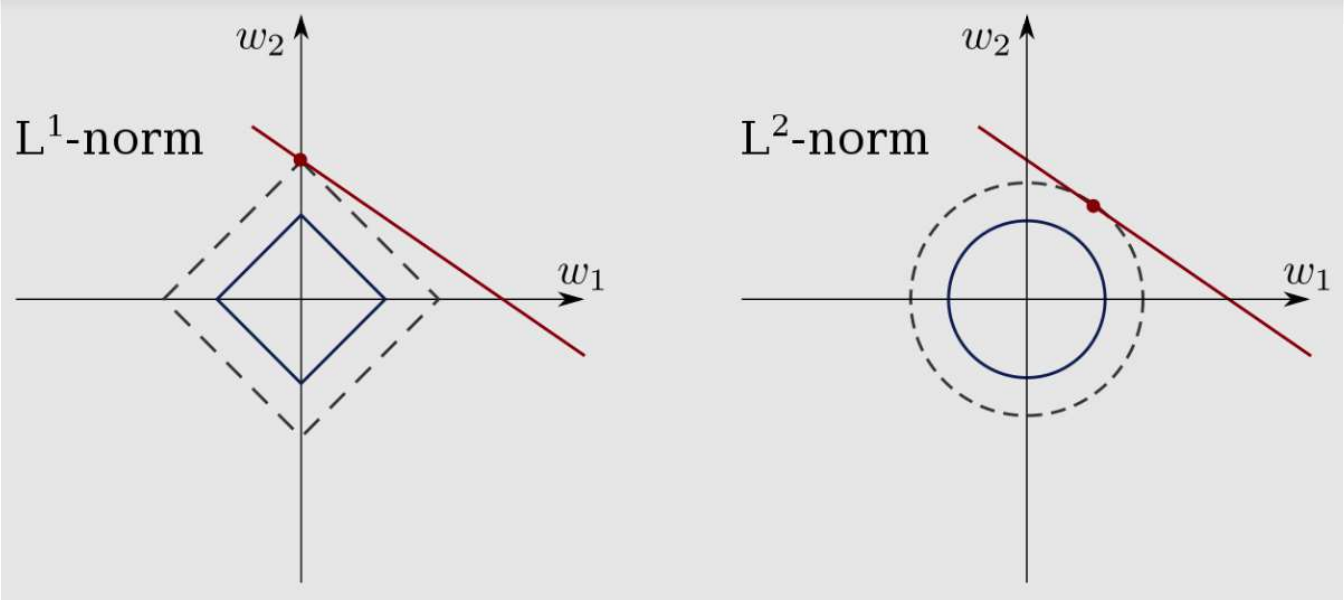
In other terms, the number of moves(distance) required to get the bishop over the red squares is equal to the Manhattan distance, which is two.

Aside from that, the Manhattan distance would be preferred over the Euclidean distance if the data present many outliers.

Moreover, L1-norm gives a more sparse estimation than l2-norm.

Besides that, L1-norm and L2-norm are commonly used in Regularization for a neural network to minimize the weights or zero out some values, like the one used on lasso regression.

Get unlimited access Open in app



Forms of the constraint regions for lasso and ridge regression(Source: Wikipedia).

As you can see in the figure above, L1-norm tries to zero out the W1 weight and minimize the other one. However, L2-norm tries to minimize both W1 and W2 weights(like W1 = W2).

In the meantime, I don't want this article to take a deep dive into regularization because its main goal is to explain common distance functions while stating some usage here and there and making it as digestible as it would be. Thus, let's move on.

③. Canberra distance.

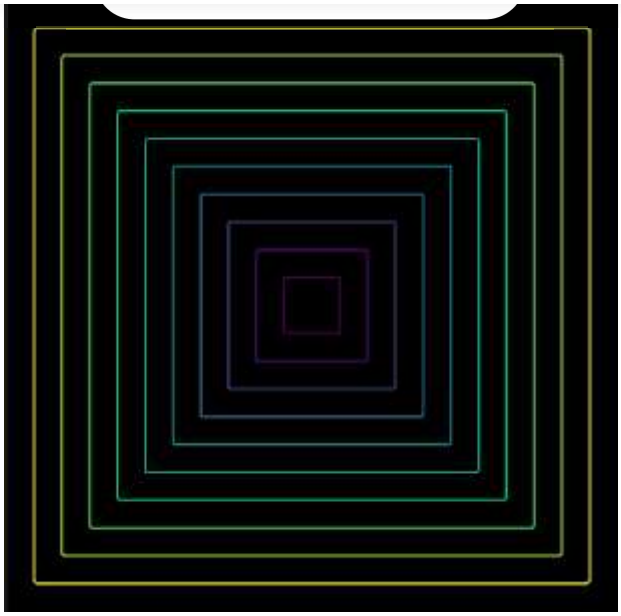
It is a weighted version of manhattan distance used in Clustering, like Fuzzy Clustering, classification, computer security, and ham/spam detection systems. It is more robust to outliers in contrast to the previous metric.

$$d(P,Q) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i + q_i|}$$

$$= \frac{|p_1 - q_1|}{p_1 + q_1} + \frac{|p_2 - q_2|}{p_2 + q_2} + \ldots + \frac{|p_n - q_n|}{p_n + q_n}$$

Canberra distance.

Get unlimited access Open in app



Chebyshev contours.

The Chebyshev distance among two n-D observations or vectors is equal to the maximum absolute value of the variations between the data samples' coordinates. In a 2-D world, the Chebyshev distance between data points can be determined as the sum of absolute differences of their 2-dimensional coordinates.

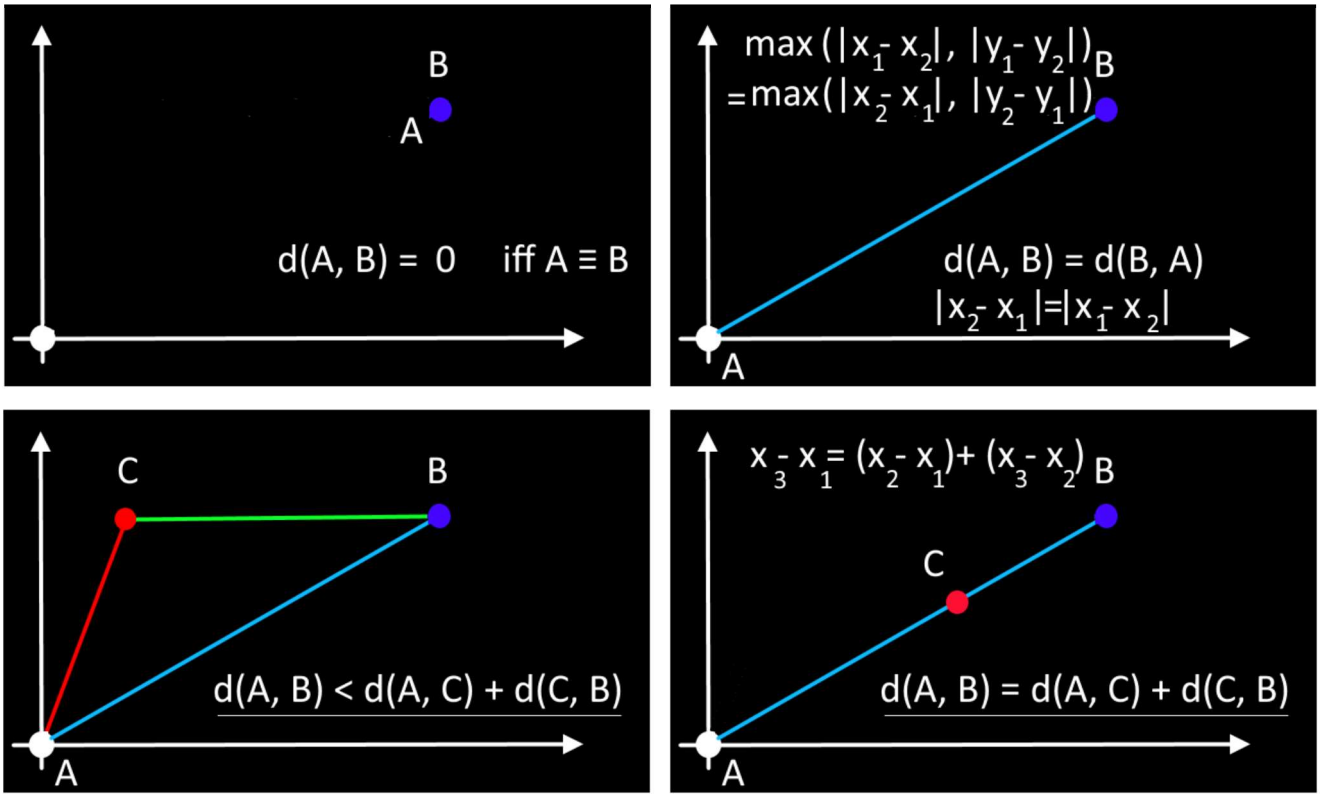
The Chebyshev distance between two points P and Q is defined as:

$$d(P,Q) = max|p_i - q_i|$$
$$= |p_1 - q_1| \text{ or } |p_2 - q_2| \text{ or } \ldots \text{ or } |p_n - q_n|$$

Chebyshev distance

The Chebyshev distance is a metric because it satisfies the four conditions for being a metric.

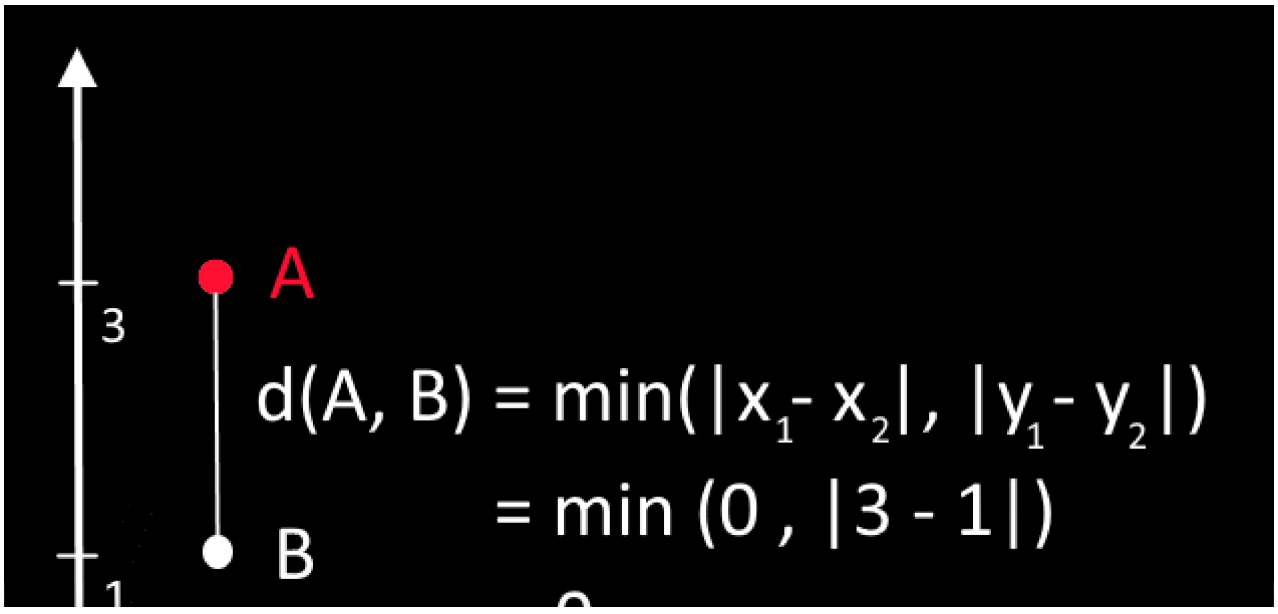
Get unlimited access Open in app



The Chebyshev distance satisfies all the conditions for being a metric.

However, you may be wondering if the min function can also be a metric!

The min function is not a metric because there is a counterexample(e.g. horizontal or vertical line) where d(A, B) = 0 and A != B. But, It should be equal to zero only if A = B!!!!

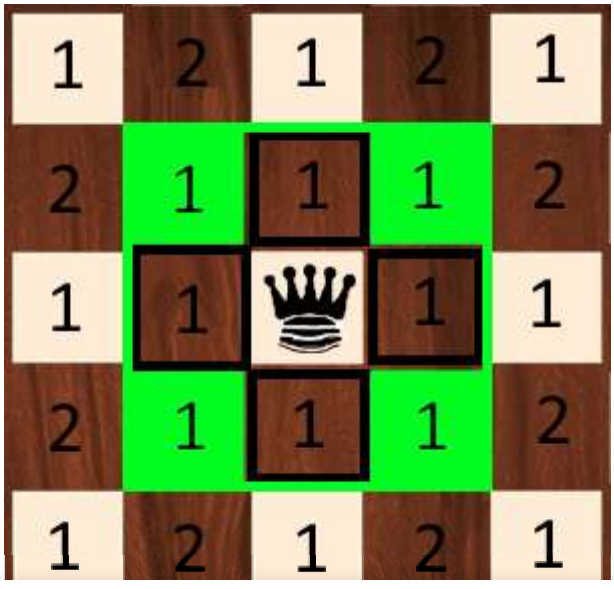


Get unlimited access Open in app

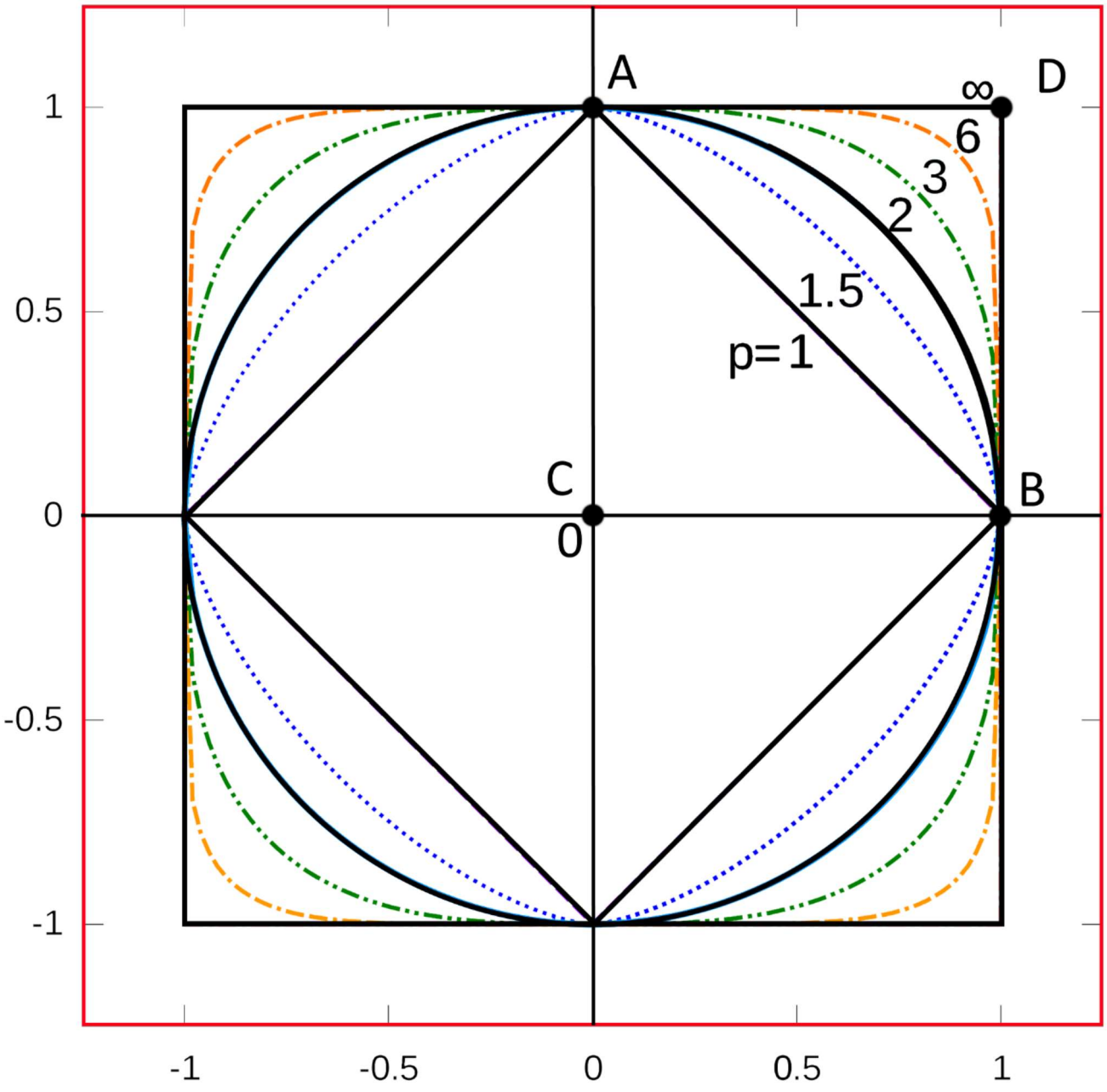
One of the use cases you can think of that uses Chebyshev metric is trading stocks, cryptocurrencies where the features are like volume, Bid, Ask… For instance, you need to find a way that tells the most cryptocurrency that has a big gap between rewards and losses. And it turns out that Chebyshev distance is a good fit for that particular situation.

Another common scenario where the Chebyshev distance is used in chessboard where the number of moves for a king, or queen, is equal to the distance in order to reach a neighbor square as the following figure shows:

The king uses Chebyshev distance to move around.



Get unlimited access Open in app



Minkowski contours for different values of p.

The Minkowski distance is just a generalization of the previous distance metrics: Euclidean, Manhattan, and Chebyshev. It is defined as the distance between two observations in the n-D space as the following formula demonstrate:

$$d(P,Q) = |P_i - Q_i|_p = \left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{1/p}$$

Get unlimited access Open in app

- **p = 1:** Manhattan distance.
- **p = 2:** Euclidean distance.
- **p → +∞ :** Chebyshev distance, logical OR(point D = **A** or **B** = **1** or **1** = 1).
- **p → 0:** logical AND(point C = **A** AND **B** = Zero).
- **p → -∞ :** min distance(the symmetric of the point D).

⑥. Cosine distance.

This metric is widely used in text mining, natural language processing, and information retrieval systems. For instance, it can be used to measure the similarity between two given documents. It can also be used to identify spam or ham messages based on the length of the message.

The Cosine distance can be measured as follows:

$$
\begin{aligned}
Cosine\_Distance &= 1 - Cosine\_Similarity \\
&= 1 - cos(\boldsymbol{P}, \boldsymbol{Q}) \\
&= 1 - \frac{\boldsymbol{P} \cdot \boldsymbol{Q}}{||\boldsymbol{P}|| \cdot ||\boldsymbol{Q}||} \\
&= 1 - \frac{\sum_{i=1}^{n} \boldsymbol{P}_i \cdot \boldsymbol{Q}_i}{\sqrt{\sum_{i=1}^{n} \boldsymbol{P}_i^2} \cdot \sqrt{\sum_{i=1}^{n} \boldsymbol{Q}_i^2}}
\end{aligned}
$$

Cosine distance.

Where P and Q represent two given points. These two points can represent the frequencies of words in documents which is explained in the following example.

Let's take, for instance, three documents that contain the following phrases:

- **Document A:** " I love to drink coffee in the morning "

Get unlimited access Open in app

- **Document C:** " My friend and I work at a coffee shop in our hometown. He tells some good jokes in the morning. We like to begin the day by drink a cup of tea each."

Computing the frequency, occurrence for each word would result in the following:

	I	love	to	drink	in	the	morning	like	my	friend	work	...
A	1	1	1	1	1	1	1	0	0	0	0	
B	1	0	1	1	0	0	0	1	0	0	0	
C	1	1	1	1	1	1	1	1	1	1	1	

words' frequencies.

Before computing the number of occurrences, you already knew a priori that documents A and B are very similar in the meaning: "I love to drink coffee." However, document C contains all the words of document A but very dissimilar in the meaning, from the frequencies table. To solve this issue, you need to compute the cosine similarity to find whether they are similar or not.

On the other hand, this can illustrate how information retrieval, or search engine, works. Think of document A as a query(short message) for a given source(image, text, video…) and document C as the web page that needs to be fetched and returned as a response to the query.

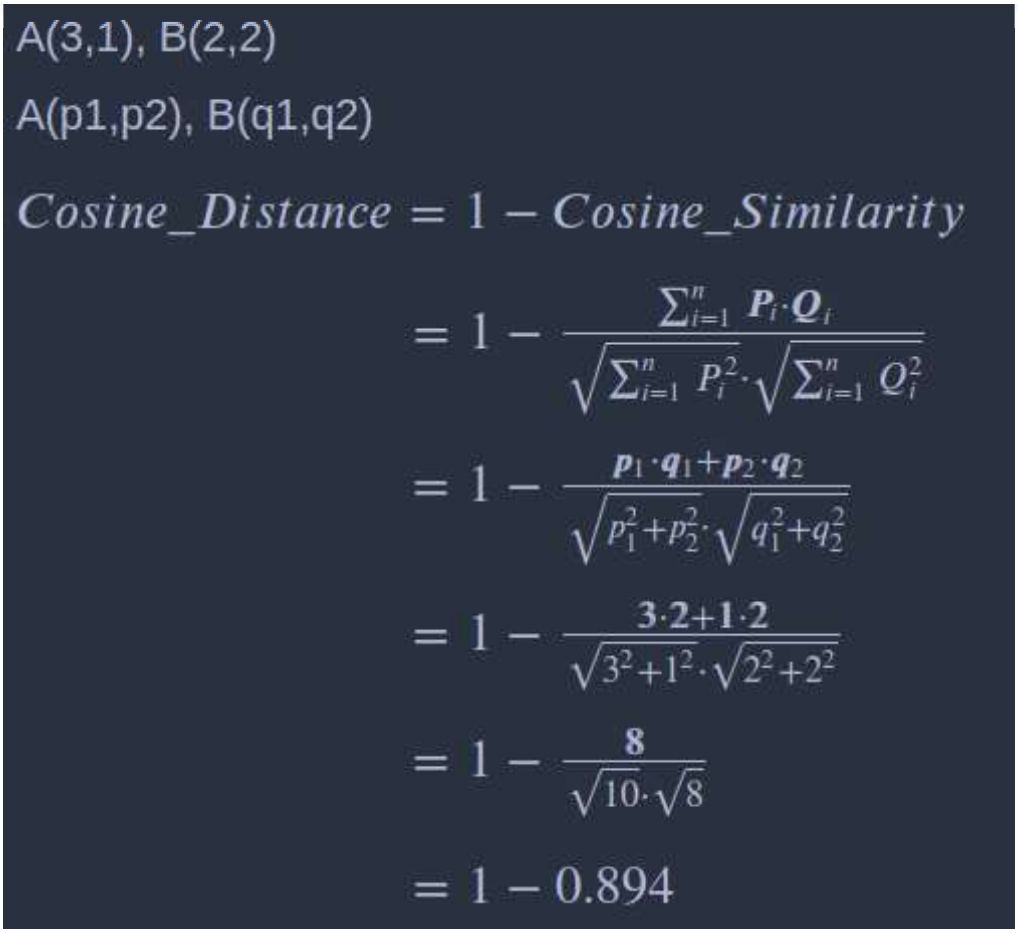
On the other side, the euclidean distance would fail to give the correct distance between short and large documents because it would be huge in this situation. Using the cosine similarity formula would compute the difference between the two documents in terms of directions and not magnitude.

To illustrate this, let's take the following two documents:

- **Document A:** " Bitcoin Bitcoin Bitcoin Money"
- **Document B:** " Money Money Bitcoin Bitcoin"

And let's denote by the word "Bitcoin" as the x-axis and the word "Money" as the y-axis. This means that document A can be represented as a vector A(3,1) and document B as B(2,2).

Get unlimited access Open in app



Computing the cosine similarity

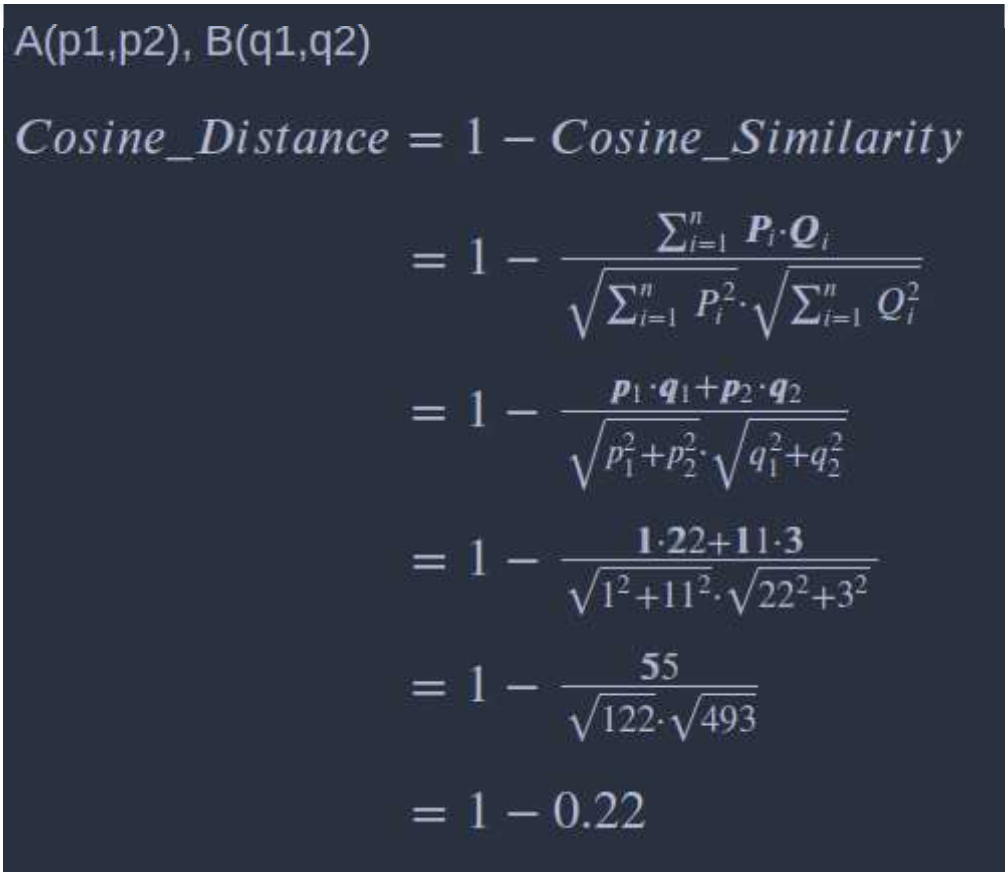
Cosine_Similarity = 0.894 means that documents A and B, are very similar. The cos(angle) is large(close to one) means the angle is small(26.6°), the two documents A and B are closed to each other.

However, you can't interpret the value of the Cosine Similarity as a percentage. For instance, the value 0.894 doesn't mean that document A is 89.4%, similar to B. It means that documents A and B are very similar, but we don't know how much percentage! There is no threshold for that value. In other words, You can interpret the value of the Cosine Similarity as follows:

> *The larger it gets, the more likely that documents A and B are similar, and vice versa.*

Let's take another example for A(1, 11) and B(22, 3)

Get unlimited access Open in app



Computing the cosine similarity

However, the euclidean distance would give a large number like 22.4, which doesn't tell the relative similarity between the vectors.

On the other hand, the cosine similarity also works well for higher dimensions.

Another interesting application of cosine similarity is the OpenPose project.

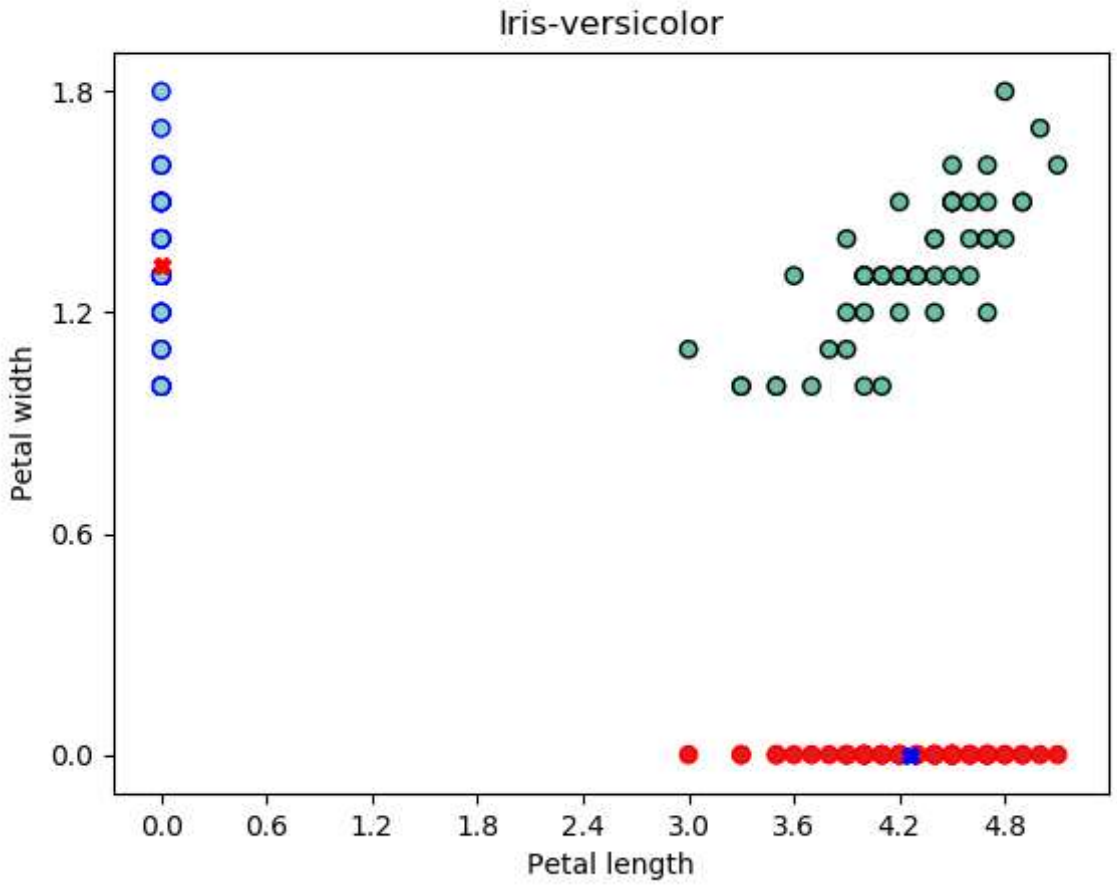
Congrats 🎇! You have made it halfway 🏁. Keep it up🏃!

⑦. Pearson Correlation distance.

The Correlation distance quantifies the strength of the linear, monotonic relationship between two attributes. Furthermore, It uses the covariance value as an initial computational step. However, the covariance itself is hard to interpret and doesn't show how much the data are close or far from the line representing the trend between the measurements.

To show what correlation means, let's go back to our Iris dataset and plot the Iris-

Get unlimited access Open in app

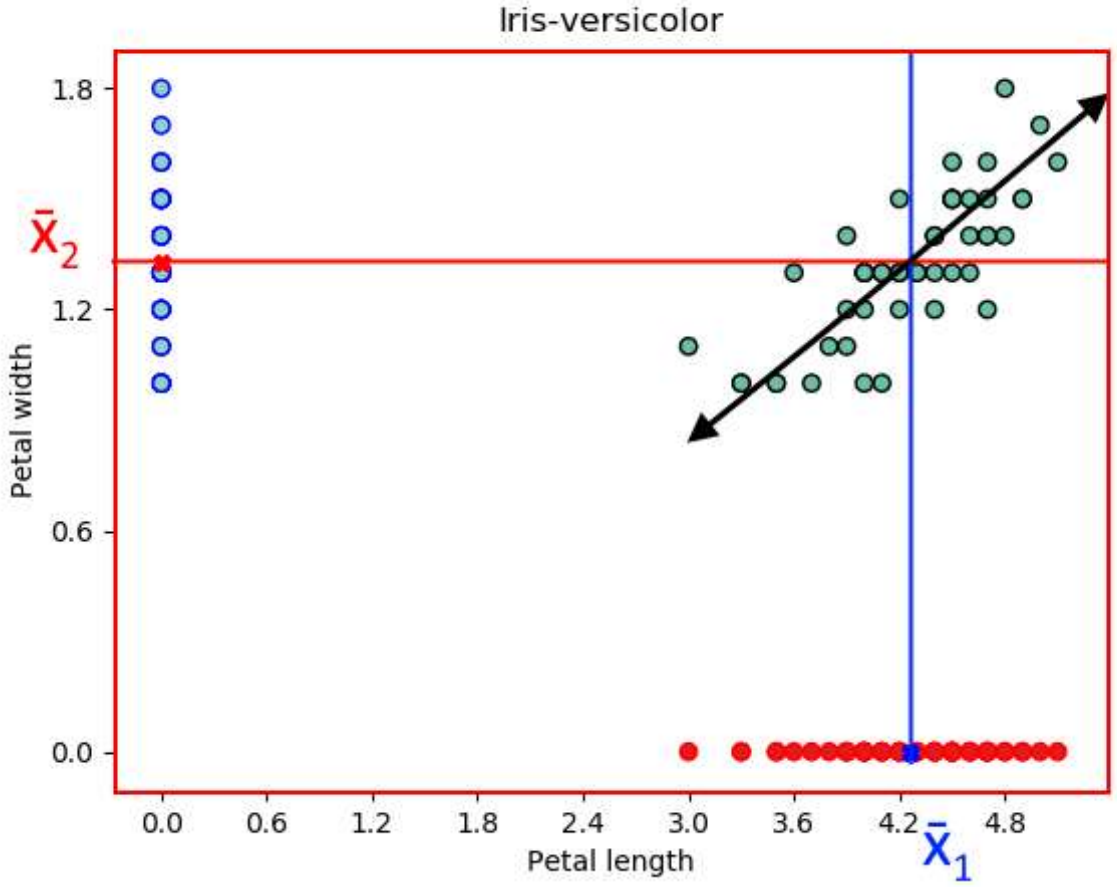


Iris-Setosa samples with two features' measurements.

The sample mean values and the variance for the two features for the same flower samples have been estimated, as the next figure shows.

Generally speaking, we can say that flowers with relatively low petal length values also have relatively low values in petal width. Also that flowers with relatively high values in petal length also have relatively high values in petal width. Additionally, we can summarize this relationship with a line.

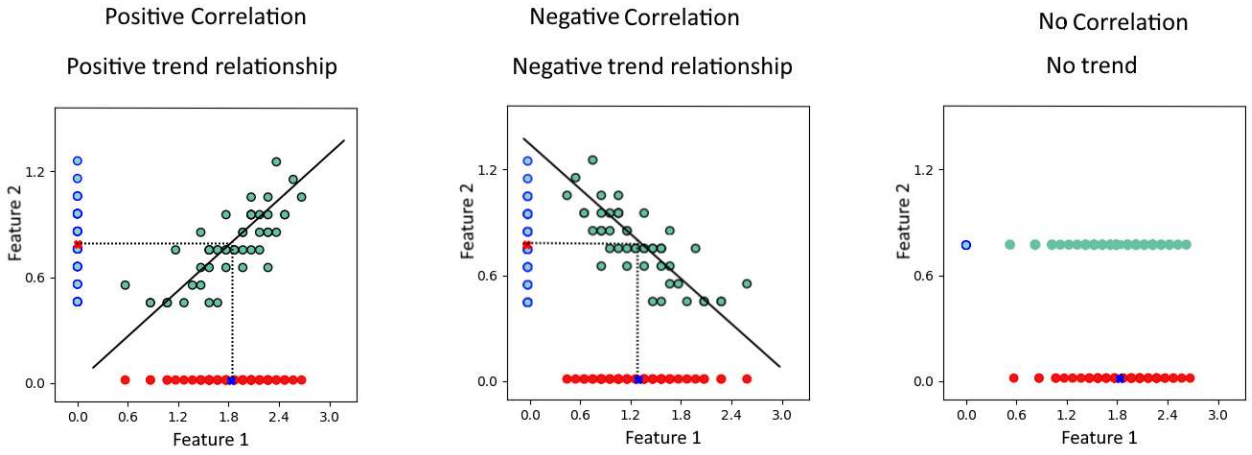
Get unlimited access Open in app



the sample mean and variance estimation.

This line represents the positive trend where both values of petal length and petal width increase together.

The Covariance value can classify three types of relationships:



three types of correlations

Get unlimited access Open in app

$$Correlation\_Distance = 1 - Correlation\_Similarity$$

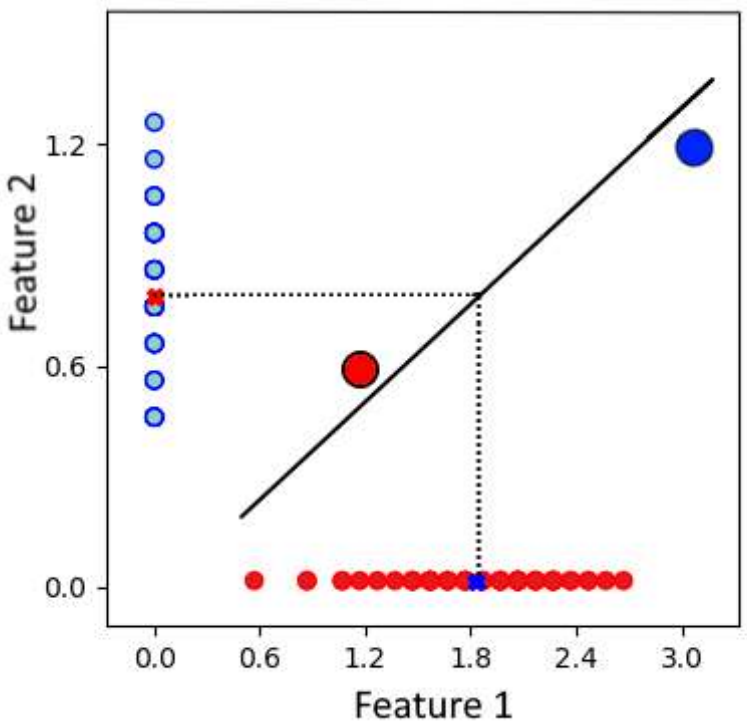
$$= 1 - \frac{Covariance(P,Q)}{\sqrt{Variance(P)} \cdot \sqrt{Variance(Q)}}$$

$$= 1 - \frac{\sum_{i=1}^{n}(p_{ij} - (1/n) \cdot \sum_{j=1}^{n} p_{ij}) \cdot (q_{ij} - (1/n) \cdot \sum_{j=1}^{n} q_{ij})}{\sqrt{\sum_{i=1}^{n}(p_{ij} - (1/n) \cdot \sum_{j=1}^{n} p_{ij})^2} \cdot \sqrt{\sum_{i=1}^{n}(q_{ij} - (1/n) \cdot \sum_{j=1}^{n} q_{ij})^2}}$$

Correlation distance

Where the numerator represents the covariance value of the observations, and the denominator represents the square root of the variance for each feature.

Let's take a simple example to demonstrate how we can compute this formula.



The red and blue points have the following coordinates, respectively:

A(1.2, 0.6) and B (3.0, 1.2).

The estimated sample means for both measurements are equals to:

Get unlimited access Open in app

A(1.2, 0.6) and B (3.0, 1.2)

$$\overline{P} = \frac{\sum_{j=1}^{n} p_{ij}}{n} = 1.82$$

$$\overline{Q} = \frac{\sum_{j=1}^{n} q_{ij}}{n} = 0.75$$

$$Correlation\_Distance = 1 - Correlation\_Similarity$$

$$= 1 - \frac{Covariance(P,Q)}{\sqrt{Variance(P)} \cdot \sqrt{Variance(Q)}}$$

$$= 1 - \frac{\sum_{i=1}^{n}(p_{ij} - (1/n) \cdot \sum_{j=1}^{n} p_{ij}) \cdot (q_{ij} - (1/n) \cdot \sum_{j=1}^{n} q_{ij})}{\sqrt{\sum_{i=1}^{n}(p_{ij} - (1/n) \cdot \sum_{j=1}^{n} p_{ij})^2} \cdot \sqrt{\sum_{i=1}^{n}(q_{ij} - (1/n) \cdot \sum_{j=1}^{n} q_{ij})^2}}$$

$$= 1 - \frac{(1.2-1.82).(0.6-0.75)+(3.0-1.82).(1.2-0.75)}{\sqrt{(1.2-1.82)^2+(3.0-1.82)^2} \cdot \sqrt{(0.6-0.75)^2+(1.2-0.75)^2}}$$

$$= 1 - \frac{\mathbf{0}.624}{\sqrt{3.2449} \cdot \sqrt{0.225}}$$

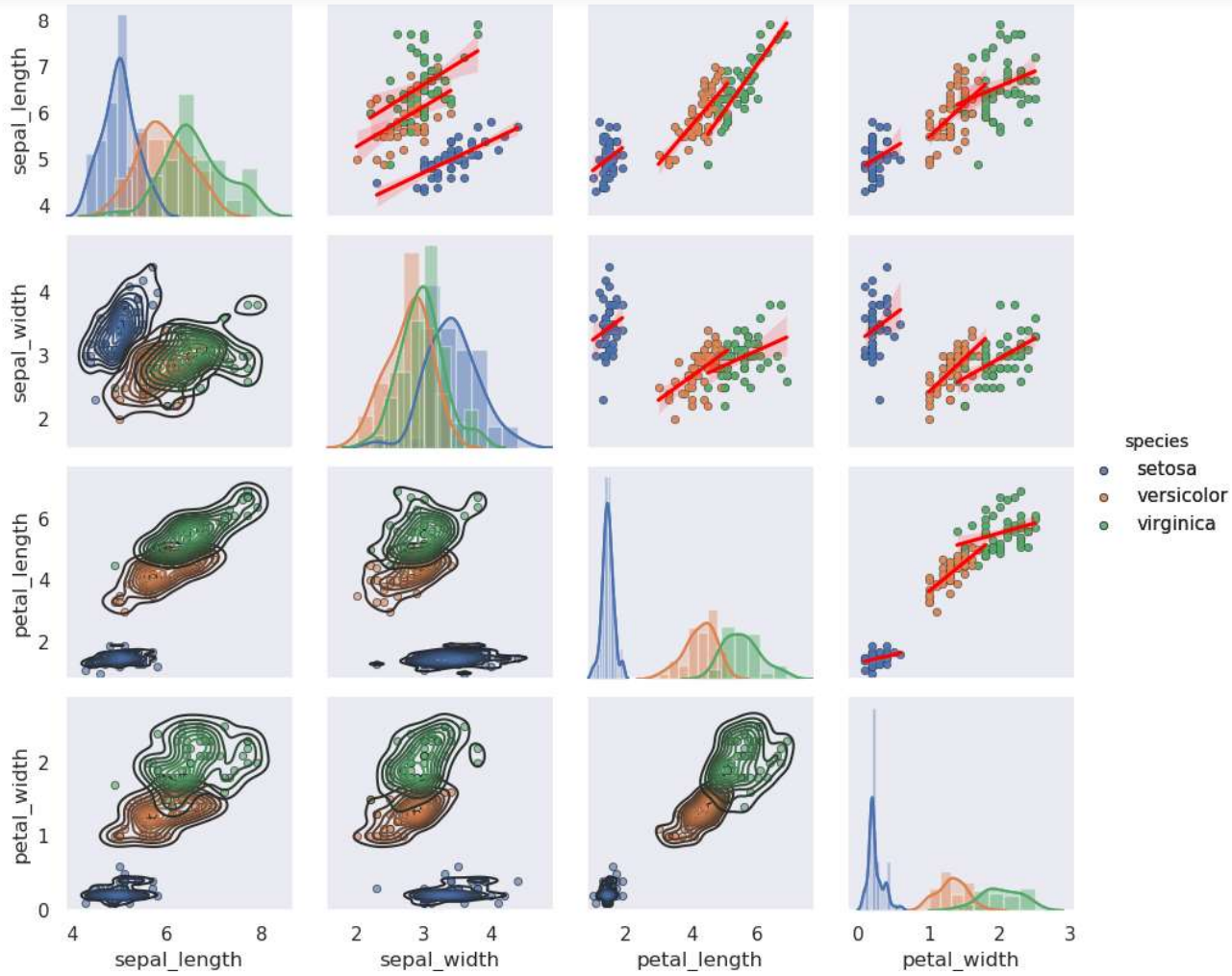
$$= 0.26932$$

One last point on this metric is that correlation doesn't mean causation. For instance, an iris-Setosa with a relatively small petal length value doesn't mean that the petal width's value should also be small. It is a sufficient but not Necessary Condition! One could say that small petal length probably contributes to small petal width, but not the only cause!

⑧. Spearman correlation.

Like Pearson correlation, Spearman correlation is used whenever we are dealing with bivariate analysis. However, unlike Pearson correlation, Spearman correlation is used when both variables are rank-ordered. It can be used for both categorical and numerical attributes.

Get unlimited access Open in app



Correlation matrix for iris dataset.

Spearman correlation index can be calculated using the following formula:

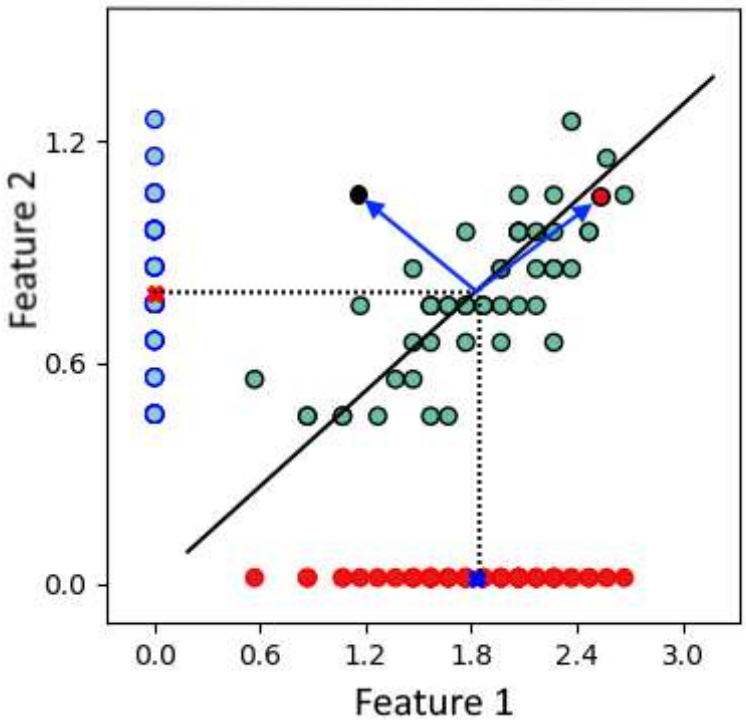
$$\rho = \frac{\sum_{i=1}^{n}(r(p_i)-\overline{r(p)})(r(q_i)-\overline{r(q)})}{\sqrt{\sum_{i=1}^{n}(r(p_i)-\overline{r(p)})^2 \cdot \sum_{i=1}^{n}(r(q_i)-\overline{r(q)})^2}} = 1 - \frac{6\sum_{i=1}^{n}(r(p_i)-r(q_i))^2}{n(n^2-1)}$$

$$Where,\ r(p_i) = rank\ of\ p_i$$
$$r(q_i) = rank\ of\ q_i$$
$$\overline{r(p)} = mean\ rank\ of\ p$$
$$\overline{r(q)} = mean\ rank\ of\ q$$
$$n = number\ of\ pairs$$

Get unlimited access Open in app

⑨. Mahalanobis distance.

It is a metric of measure mostly used in multivariate statistical testing where the euclidean distance fails to give the real distance between observations. It measures how far away a data point from the distribution.



Two points with the same value of the ED from the mean.

As shown in the previous image, the red and blue points have the same Euclidean distance from the mean. However, they don't fall in the same region or cluster: The red point is more likely similar to the data set. But the blue one is considered an outlier because it is far away from the line that represents the direction of the greatest variability in the dataset(major axis regression). Therefore, the Mahalanobis metric was introduced to solve this issue.

The Mahalanobis metric tries to decrease the covariance between the two features or attributes in the sense that you can rescale the previous plot into new axes. And these new axes represent the eigenvectors like the first eigenvector as previously shown.

This first direction of the eigenvector greatly influences the data classification because it has the largest eigenvalue. Furthermore, the dataset is spread out along this

Get unlimited access Open in app

distances from the mean between the previous two data points. And that's what the Mahalanobis metric does.

$$d(P,Q) = \sqrt{(P-Q)^T \cdot C^{-1} \cdot (P-Q)}$$

Mahalanobis distance between two objects P and Q.

Where **C** represents the covariance matrix between the attributes or features.

To demonstrate this formula's usage, let's compute the distance between A(1.2, 0.6) and B (3.0, 1.2) from our previous example in the correlation distance section.

Let's now evaluate the covariance matrix, which is defined as follows:

$$Covariance_{matrix} = \begin{pmatrix} S_p^2 & S_{pq} \\ S_{qp} & S_q^2 \end{pmatrix} = \begin{pmatrix} Var_{[P]} & Cov_{[P,Q]} \\ Cov_{[Q,P]} & Var_{[Q]} \end{pmatrix}$$

Covariance matrix in 2d space

Where Cov[P,P] = Var[P] and Cov[Q,Q]= Var[Q], and

Get unlimited access Open in app

$$Cov(P,Q) = \sum \frac{(P_i - \overline{P})(Q_i - \overline{Q})}{N-1} = \sum \frac{p_i q_i}{N-1}$$

Where:

Cov(P, Q) = Covariance of corresponding scores in the two sets of data

N = Number of points in each set of data

$\overline{P}$ = Sample mean of the N points in the first data set

$P_i = i^{th}$ row point in the first set of scores

$p_i = i^{th}$ deviation score in the first set of scores

$\overline{Q}$ = Sample mean of the N points in the second data set

$Q_i = i^{th}$ row point in the second set of scores

$q_i = i^{th}$ deviation score in the second set of scores

Covariance formula between two features.

Therefore, the Mahalanobis distance between the two objects A and B can be calculated as follows:

A(1.2, 0.6) and B (3.0, 1.2)

$$\overline{P} = \frac{\sum_{j=1}^{n} p_{ij}}{n} = 1.82$$

$$\overline{Q} = \frac{\sum_{j=1}^{n} q_{ij}}{n} = 0.75$$

$$Covariance_{matrix} = \begin{pmatrix} 2.3531 & 0.8585 \\ 0.8585 & 0.8585 \end{pmatrix}$$

$$Mahalanobis_{distance}(P,Q)^2 = (P-Q)^T \cdot C^{-1} \cdot (P-Q)$$

$$= \begin{pmatrix} 1.2-3.0 \\ 0.6-1.2 \end{pmatrix}^T \cdot \begin{pmatrix} 2.3531 & 0.8585 \\ 0.8585 & 0.8585 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1.2-3.0 \\ 0.6-1.2 \end{pmatrix}$$

$$= \begin{pmatrix} -1.8 & -0.6 \end{pmatrix} \cdot \begin{pmatrix} 0.66907 & -0.66907 \\ -0.66907 & 1.833897 \end{pmatrix} \cdot \begin{pmatrix} -1.8 \\ -0.6 \end{pmatrix}$$

$$= \begin{pmatrix} -0.802884 & 0.1039878 \end{pmatrix} \cdot \begin{pmatrix} -1.8 \\ \end{pmatrix}$$

Get unlimited access Open in app

Mahalanobis distance example.

In addition to its use cases, The Mahalanobis distance is used in the Hotelling t-square test.

⑩. Standardized Euclidian distance.

Standardization or normalization is a technique used in the preprocessing stage when building a machine learning model. The dataset presents a high difference between the minimum and the maximum ranges of features. This scale distance would affect the ML model when clustering data, leading to a wrong interpretation.

For instance, let's take a situation where we have two different features that present a large difference in range variations. For example, let's say we have a feature that varies from 0.1 to 2 and another feature that goes from 50 to 200. Computing distance using these values would make the second feature more dominant, leading to incorrect results. In other terms, the euclidean distance will be highly influenced by attributes that have the largest values.

That's why standardization is a necessity in order to let the features contribute in an equal manner. It is done by transforming variables to all have the same variance that is equal to one and centralize the features around the mean like the following formula show:

$$Z = \frac{x - \mu}{\sigma}$$

z-score standardization.

The standardized Euclidean distance can be expressed as follows:

$$d = \sqrt{\sum^{n} (p_i - q_i)^2}$$

Get unlimited access Open in app

standardized euclidian distance.

We can apply this formula to compute the distance between A and B.

A(1.2, 0.6) and B (3.0, 1.2)

$$\sigma_1 = 1.53398$$

$$\sigma_2 = 0.57532$$

$$d_{P,Q} = \sqrt{\sum_{i=1}^{n} \left(\frac{p_i - q_i}{\sigma_i}\right)^2}$$

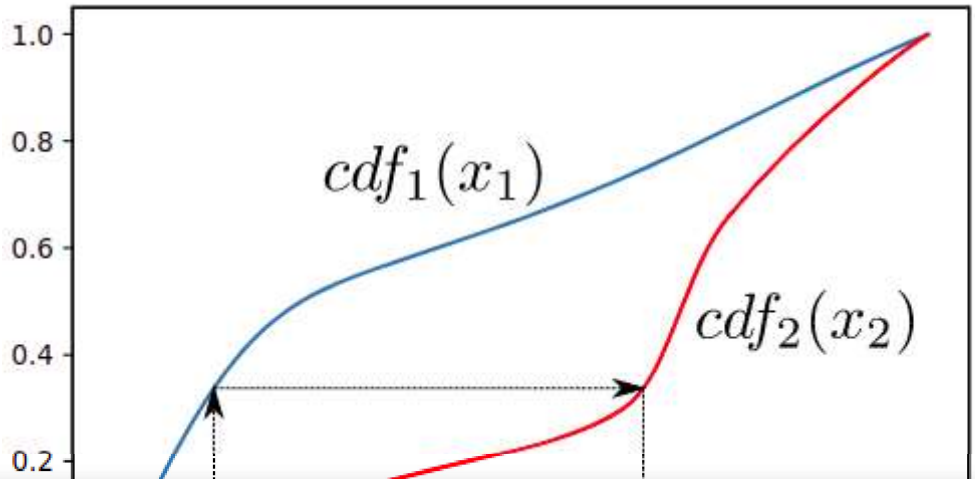
$$= \sqrt{\left(\frac{1.2-3.0}{1.53398}\right)^2 + \left(\frac{0.6-1.2}{0.57532}\right)^2}$$

$$= \sqrt{1.376910195 + 1.087635958}$$

$$= 1.57$$

⑪. Chi-square distance:

Chi-square distance is commonly used in computer vision while doing texture analysis in order to find (dis)similarities between normalized histograms, known as “Histogram matching”.



Get unlimited access Open in app

Histogram matching. Source: Wikipedia

A face recognition algorithm would be a great example that uses this metric in order to compare two histograms. For instance, in the prediction step of a new face, the model computes the histogram from the newly captured image, compared it with the saved histograms(often stored in a .yaml file), and then tries to find the best match for it. This comparison is made via computing the Chi-square distance between each pair of histograms of n bins.

$$d(P,Q) = \sum_{i=1}^{n} \frac{(P_i - Q_i)^2}{P_i + Q_i}$$

Chi-square distance

As you may know, this formula is different from the Chi-square statistics test for standard normal distributions, where it is used to decide whether to retain or reject the null hypothesis using the following formula:

$$x^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Chi-square test formula.

Where O and E represent the observed and expected data values, respectively.

To illustrate how it is used, let's say a survey was done on 1000 persons to test a given vaccine's side effect and see if there any significant difference based on gender. Therefore, each person can be one of four categories:

1- Male without side effects.

2- Male with side effects.

3- Female without side effects.

Get unlimited access Open in app

The null hypothesis is: there is no significant difference in side effects between the two genders.

In order to retain or reject this hypothesis, you compute the Chi-square test value for the following data:

		side effect	No side effect	Total
Males	Observed	320	100	420
	Expected	311	109	
Females	Observed	420	160	580
	Expected	429	151	
Total		740	260	1000

Collected data.

By plugging these values into the Chi-square test formula, you will get 1.7288.

Using a Chi-square table with a degree of freedom equal to 1, you will get a probability between 0.2 and 0.1 > 0.05 → you retain the null hypothesis.

Note that degree of freedom = (number of columns -1) x (number or rows -1)

Besides, I just wanted to give you a quick refresher on the hypothesis testing; I hope you find it helpful. Anyway, let's keep things moving.🏃🏃🏃

⑫. Jensen-Shannon distance.

The Jensen-Shannon distance computes the distance between two probability distributions. It uses the Kullback Leibler divergence(The relative entropy) formula in order to find the distance.

$$J(P,Q) = \frac{1}{2}\left(D(P \parallel R) + D(Q \parallel R)\right)$$

Get unlimited access Open in app

Jensen-Shannon distance.

Where R is the midpoint between P and Q.

Besides, Just a quick note on how you can interpret the value of entropy:

> Low entropy for event A means that there is a high knowing that this event will occur; in other terms, I am less surprised if event A is going to happen, and I am highly confident that it will happen. The same analogy for High entropy.

On the other hand, the Kullback Leibler divergence itself is not a distance metric since it is not symmetric: $D(P \,||\, Q) \,!= D(Q \,||\, P)$.

⑬. Levenshtein distance

A metric for measuring similarity between two strings. It is equal to the minimum number of operations required to transform a given string into another one. There are three types of operations:

1. Substitution.
2. Insertion
3. Deletion

For Levenshtein distance, the substitution cost is two units and one for the other two operations.

For instance, let's take two strings s= “Bitcoin” and t = “Altcoin”. To go from s to t, you need two substitutions of the letters “B” and “I” by the letters “A” and “l”. Thus, the d(t, s) = 2 * 2 = 4.

There are many use cases for the Levenshtein distance like spam filtering,

Get unlimited access Open in app

The hamming distance is equal to the number of digits where two codewords of the same length differ. In the binary world, it is equal to the number of different bits between two binary messages.

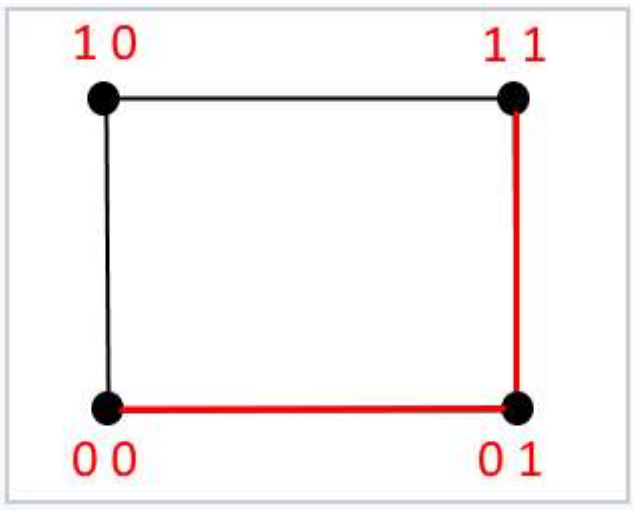
For example, the hamming distance between two messages can be calculated using:

$$H(P,Q) = \sum_{i=1}^{n} |p_i - q_i|$$

Hamming distance.

As you may notice, it looks like the manhattan distance in the context of categorical data.

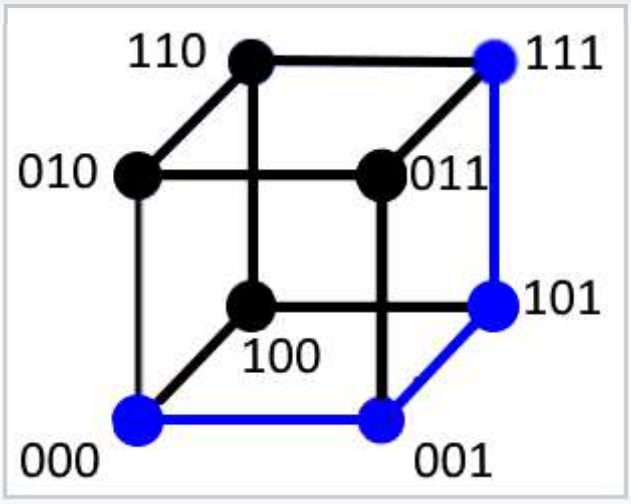
For messages of length 2 bits, this formula represents the number of edges that separate two given binary messages. It can be at most equal to two.



2-bits codewords.

In the same manner, for messages of length 3 bits, this formula represents the number of edges that separates two given binary messages. It can be at most equal to three.

Get unlimited access Open in app



3-bits codewords.

Let's take some example to illustrate how the hamming distance is calculated:

H(100001, 010001) = 2

H(110, 111) = 1

If one of the messages contains all zeros, the hamming distance is called hamming weight and equal to the number of non-zero digits in a given message. In our case, it is equal to the total number of ones.

H(110111,000000) = W (110111) = 5

The hamming distance is used to detect and correct, if possible, errors in received messages that have been transmitted over an unreliable noisy channel.

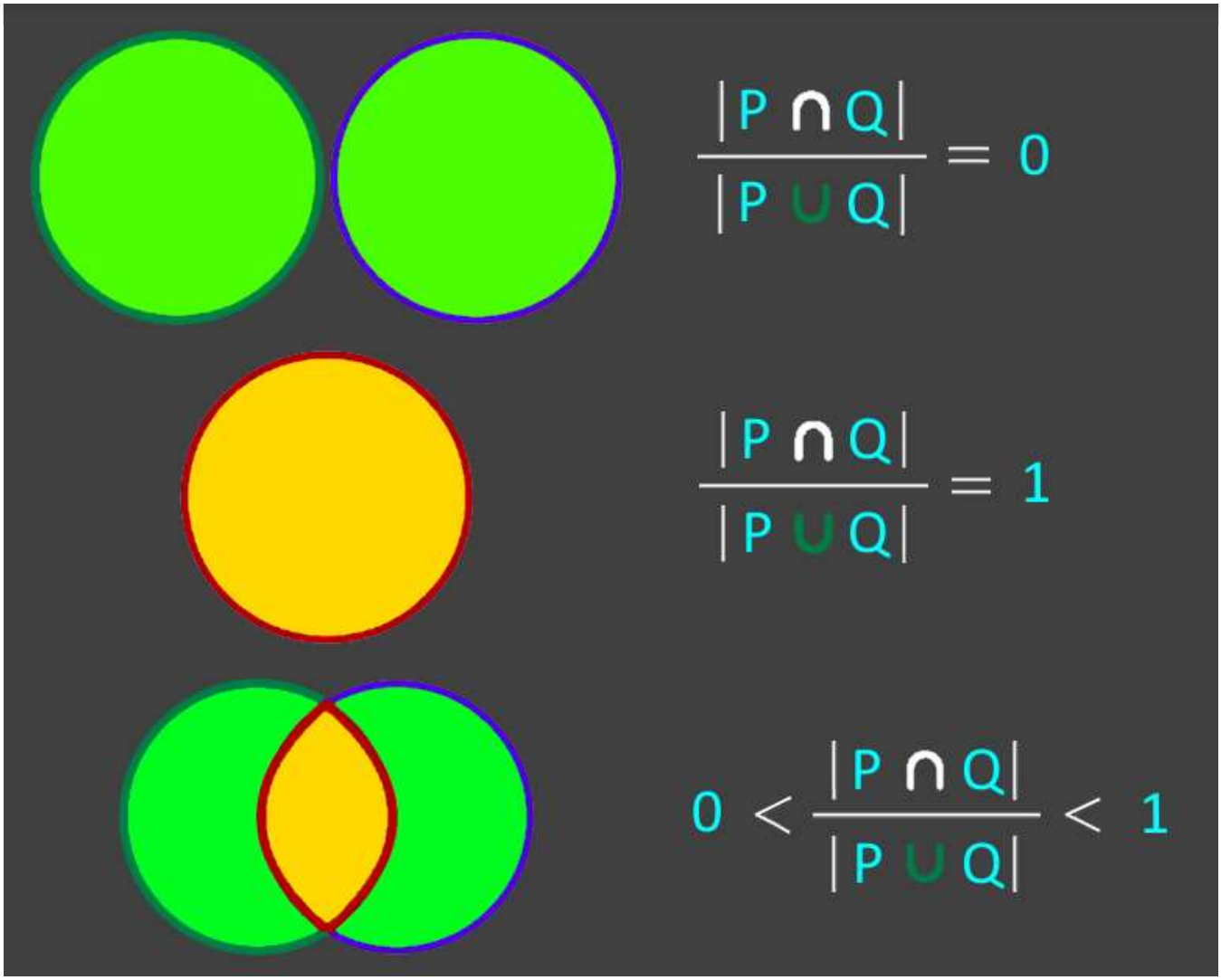
⑮. Jaccard/Tanimoto distance.

A metric used to measure the similarity between two sets of data. One may argue that in order to measure similarity is to compute the size(cardinality, number of elements.) of the intersection between two given sets.

However, the number of common elements alone cannot tell us how relative it is compared to the sets' size. Hence the intuition behind the Jaccard coefficient.

So Jaccard proposes that, in order to measure similarity, you need to divide the size of the intersection by the size of the union for the two sets of data

Get unlimited access Open in app



Jaccard distance.

Jaccard distance is complementary to the Jaccard coefficient to measures dissimilarity between data sets and is calculated by:

Get unlimited access Open in app

$$D(P,Q) = 1 - J(P,Q)$$

$$= 1 - \frac{|P \cap Q|}{|P \cup Q|}$$

$$= \frac{|P \cup Q| - |P \cap Q|}{|P \cup Q|}$$

$$= \frac{|P \cap Q| + |P - Q| + |Q - P| - |P \cap Q|}{|P \cap Q| + |P - Q| + |Q - P|}$$

$$= \frac{|P - Q| + |Q - P|}{|P \cup Q|}$$

Jaccard distance.

The following illustration explains how this formula can be used for non-binary data.

$$P = \{a,b,c\} \quad Q = \{a,b,c,d,e,f,g,h\}$$

$$J = \frac{|P \cap Q|}{|P \cup Q|} = \frac{|\{a,b,c\}|}{|\{a,b,c,d,e,f,g,h\}|} = \frac{3}{8}$$

Jaccard index example.

For binary attributes, the Jaccard similarity is calculated using the following formula:

Get unlimited access Open in app

$$J(P,Q) = \frac{a}{a+b+c}$$

Where:

a = number of positive attributes in both objects.

b = number of positive attributes for the i^{th} object and negative for the j^{th} object.

c = number of negative attributes for the i^{th} object and positive for the j^{th} object.

Jaccard index for binary data.

Jaccard index can be useful in some domains like semantic segmentation, text mining, E-Commerce, and recommendation systems.

Now you may be thinking: “Ok, but you have just mentioned earlier that cosine distance can also be used in text mining. What do you prefer to use as a metric for a given clustering algorithm? What the difference between both metrics anyway?”

Glad you asked this question. In order to answer this, we need to compare each term of the two formulas.

$$J(P,Q) = \frac{|P \cap Q|}{|P \cup Q|} = \frac{|P \cap Q|}{|P \cap Q| + |P - Q| + |Q - P|}$$

$$C(P,Q) = \frac{|P \cap Q|}{\sqrt{|P| \cdot |Q|}} = \frac{|P \cap Q|}{\sqrt{(|P \cap Q| + |P - Q|)(|P \cap Q| + |Q - P|)}}$$

Jaccard and cosine formulas.

The only difference between the two formulas is the denominator term. Instead of computing the union’s size between the two sets with Jaccard, you are computing the magnitude of the dot product between P and Q. Instead of adding the terms in the Jaccard formula’s denominator; you are computing the product between the two in the cosine formula. I don’t know what the interpretation of that is. As far as I know, The dot product gives us how much one vector goes in the direction of another. Other than

Get unlimited access Open in app

The Sørensen–Dice distance is a statistical metric used to measure the similarity between sets of data. It is defined as two times the size of the intersection of P and Q, divided by the sum of elements in each data set P and Q.

$$D(P,Q) = \frac{2 \cdot |P \cap Q|}{|P| + |Q|}$$

Sørensen–Dice coefficient.

Like Jaccard, the similarity values range from zero to one. However, unlike Jaccard, this dissimilarity measure is not a metric since it doesn't satisfy the triangle inequality condition.

Sørensen–Dice is used in lexicography, image segmentation, and other applications.

🇨. Pydist2.

In the past few weeks, while I was researching similarity and dissimilarity measures, I thought it would be a fun/great idea to reimplement these types of measures in python as a coding practice. I took the inspiration from Piotr Dollar's toolbox repo written in Matlab and can also be found on the MathWorks website. Therefore, pydist2 is a python package, 1:1 code adoption of pdist and pdist2 Matlab functions, for computing distance between observations. The list of methods of measuring the distance currently supported by pydist2 is available at read the docs.

🇩. Conclusion.

Hooray!!! You have just reached the end of this article. Hopefully, you didn't get overwhelmed by its content at this point, which I tried to make as concise and clear as possible.

Get unlimited access Open in app

For me, writing this article was an enjoyable experience in my data science journey. It can stress the fact of spending countless hours researching and exploring a mathematical branch of data science and machine learning as well as practicing my coding skills while writing my first python package: pydist2.

If you want to contribute to this project, I am very welcoming to see your pull requests on pydist2 repo. You can Checkout this tutorial if you want to contribute.

If you have noticed any mistakes while reading this article, please mention them in the comments below In order to improve the content.

If you have any suggestions, drop me a message on LinkedIn or send me an email.

If you want to use anything from this article, please cite it as:

> *Mahmoud Harmouch, 17 types of similarity and dissimilarity measures used in data science, medium.com. Mar-14-2021*

If my analytical skills have convinced you and you want to see more, following me and/or sharing this article would help me gain exposure in the data science community and spread useful information.

And that's it for today's blog. Good luck in your own data science journey! Thanks for reading, and see you in the next one!

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get unlimited access Open in app