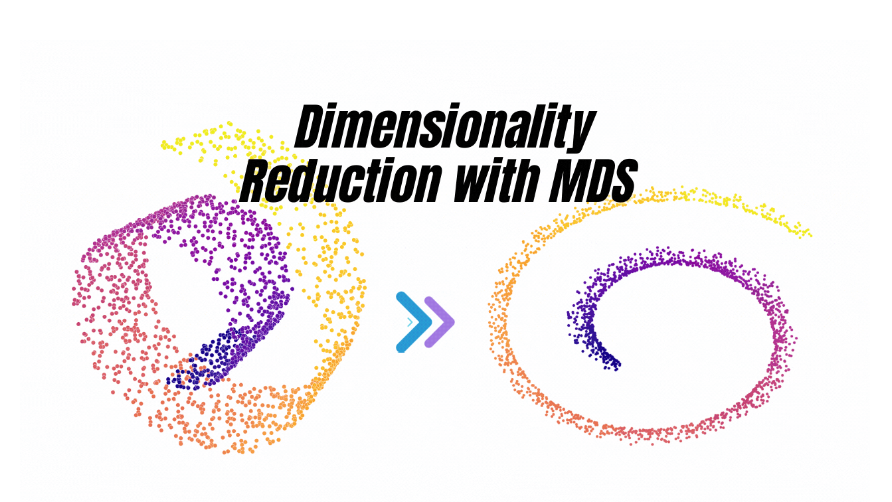
**MDS: Multidimensional Scaling — Smart Way to Reduce Dimensionality in Python**

**Manifold learning is an approach to non-linear dimensionality reduction**



[Multidimensional scaling](https://en.wikipedia.org/wiki/Multidimensional_scaling) ([MDS](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html#sklearn.manifold.MDS)) seeks a low-dimensional representation of the data in which the distances respect well the distances in the original high-dimensional space.

In general, [MDS](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html#sklearn.manifold.MDS) is a technique used for analyzing similarity or dissimilarity data. It attempts to model similarity or dissimilarity data as distances in a geometric spaces. The data can be ratings of similarity between objects, interaction frequencies of molecules, or trade indices between countries.

Working with large data presents many challenges, one of them being a loss of efficiency and performance in your models due to too high dimensionality.

Luckily, many dimensionality reduction techniques are available that can help us overcome challenges by enabling us to remove “less important” data.

# Types of Multidimensional Scaling (MDS)

There are two major types of MDS, metric (classical) and non-metric. While both aim to find the best lower-dimensional representation of your high-dimensional data, their differences arise in the type of data they are designed to work with.

* **Metric (classical) MDS** — is also known as **Principal Coordinate Analysis (PCoA).**Make sure not to confuse it with Principal Component Analysis (PCA), a separate yet similar technique.  
  Metric MDS attempts to model the similarity/dissimilarity of data by calculating distances between each pair of points using their **geometric coordinates**. The key here is the ability to measure a distance using a linear scale. E.g., a distance of 10 units would be considered twice as far as a distance of 5 units.
* **Non-metric MDS**— is designed to deal with ordinal data. E.g., you may have asked your customers to rate your products on a scale of 1 to 5, where 1 is terrible, and 5 is amazing. Here, a product with a rating of 2 is **not**necessarily twice as good as a product with a rating of 1. It’s the order that matters (1 < 2 < 3 < 4 < 5) rather than the absolute value. This is the kind of situation where you would use non-metric MDS.

As mentioned in the intro, in this article, I focus on **metric MDS**. Note, though, Sklearn’s implementation of the MDS algorithm in Python lets you easily switch between metric and non-metric approaches.

# How does metric Multidimensional Scaling (metric MDS) actually work?

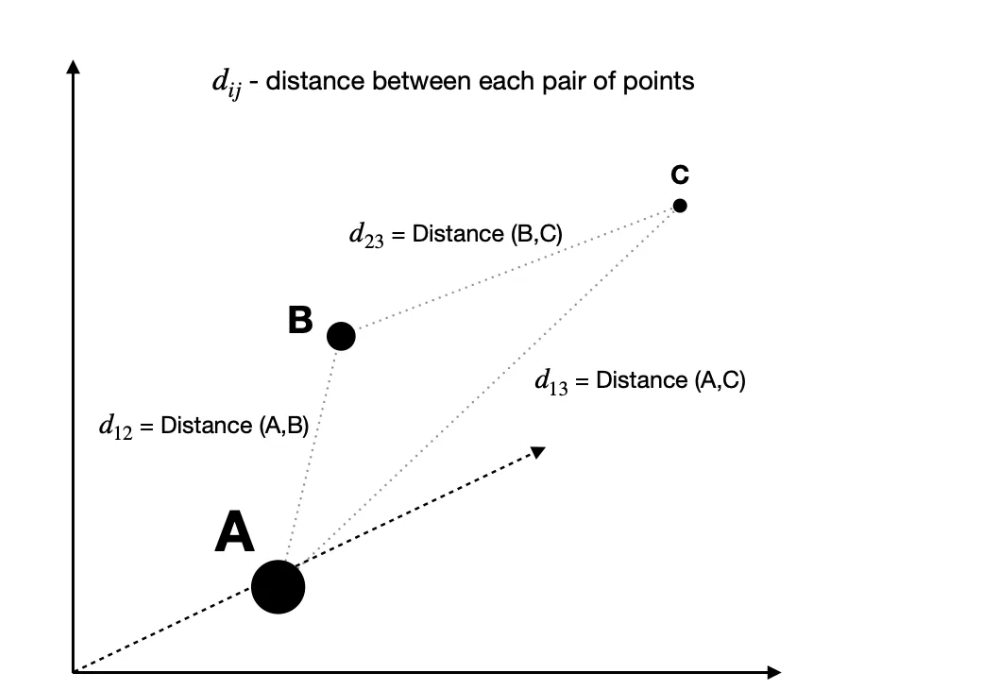
In general, the metric MDS calculates distances between each pair of points in the original high-dimensional space and then maps it to lower-dimensional space while preserving those distances between points as well as possible.

*Note, the number of dimensions for the lower-dimensional space can be chosen by you. Typically, one would choose either 2D or 3D as it allows for the data to be visualized.*

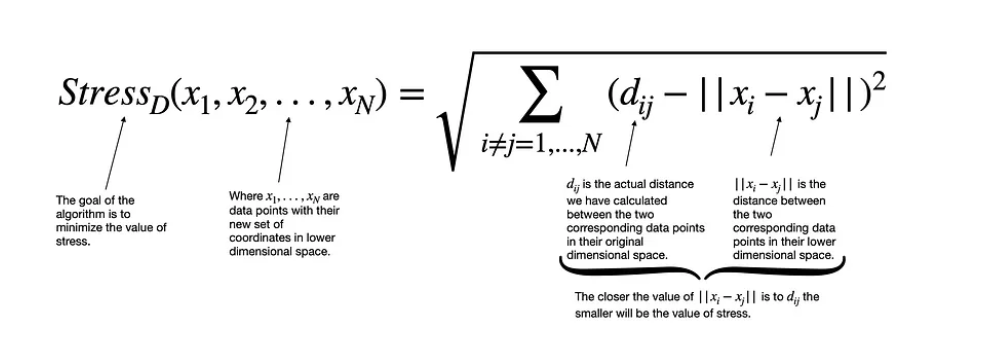
So let’s take a look at high-level steps performed by metric MDS.

## Steps used by metric MDS algorithm

**Step 1** — The algorithm calculates distances between each pair of points, as illustrated below.



**Step 2** — With the original distances known, the algorithm attempts to solve the optimization problem by finding a set of coordinates in a lower-dimensional space that minimizes the value of Stress.



Multiple approaches can be used to optimize the above cost function, such as Kruskal’s steepest descent method or De Leeuw’s iterative majorization method. However,

One important thing to note is that both aforementioned methods are iterative approaches, sometimes giving different results since they are sensitive to the initial starting position.

However, Sklearn’s implementation of MDS allows us to specify how many times we want to initialize the process. **In the end, the configuration with the lowest stress is picked as the final result.**