()

Login

(http://pages.sisense.com/dashboard-software-trial-download.html?source=big_data&cmp=leaderboard)

ANALYTICS (HTTP://BIGDATA-MADESIMPLE.COM/CATEGORY/TECH-AND-TOOLS/ANALYTICS/)
How to recover data from corrupt Elasticsearch Indices in 3 simple steps (http://bigdata-madesimple.com/how-to-recover-data-from-corrupt-elasticsearch-indices-in-3-simple-steps/)

MARKETING (HTTP://BIGDATA-MADESIMPLE.COM/CATEGORY/SECTORS/MARKETING/)
Debunking five digital marketing myths around Big Data (http://bigdata-madesimple.com/debunking-five-digital-marketing-myths-around-big-data/)

DATA SCIENCE

Saimadhu Polamuri (http://bigdata-madesimple.com/contributers/saimadhu-polamuri/)

(http://bigdata-madesimple.com/contributers/saimadhu-polamuri

Follow

(https://www.facebook.com/saimadhu.seven/) (https://twitter.com/saimadhup/) (https://www.linkedin.com/in/saimadhu/)

Implementing the five most popular

Similarity Measures in Python

Login

06th Jun `15, 12:00 AM in Data Science (http://bigdata-madesimple.com/category/tech-and-tools/data-science/)

The buzz term similarity distance measures has a variety of definitions among math and data mining practitioners. As a result,...

(http://bigdata-madesimple.com/5-groups-of-data-scientists-which-group-are-you-in/)

The buzz term *similarity distance measures* has a variety of definitions among math and data mining (http://dataaspirant.com/category/datamining/) practitioners. As a result, concepts involving the term and it's usage can go right over the heads of beginners. So today, I write this post to give simplified and intuitive definitions of similarity measures, as well as to dive into the implementation of five of the most popular similarity measures.

(http://bigc madesimp /ibm-uk-invest-480m-in-cognitive-computing big-data-research/)

Similarity

Similarity is the measure of how alike two data objects are. Similarity in the data mining (http://dataaspirant.com/2014/09/16/data-mining/) context is usually described as a distance with dimensions representing features of the objects. If this distance is small, there will be high degree of similarity; if this distance is

Similarity is the basic building block for techniques such as Recommendation engines, clustering, classification and anomaly detection.

Login

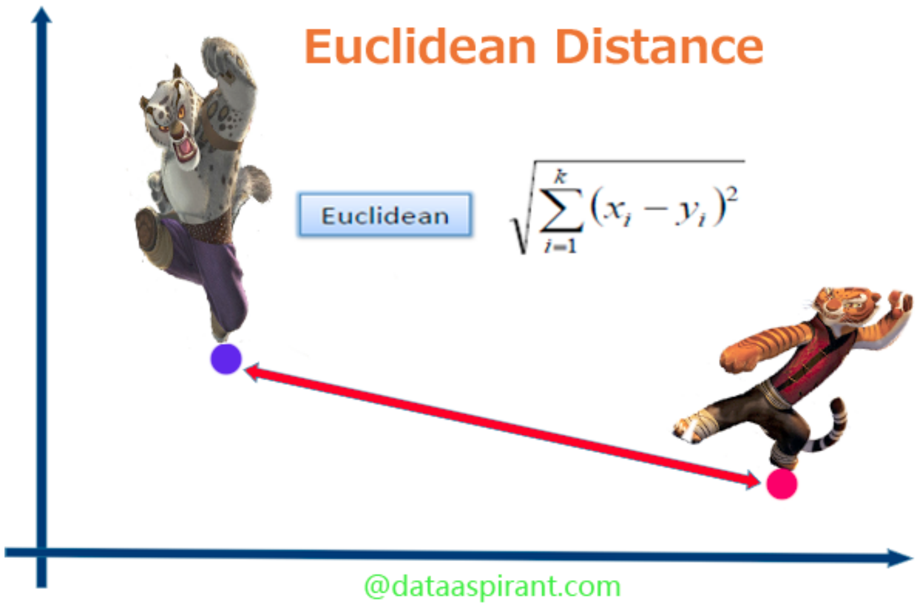
Similarity is subjective and is highly dependent on the domain and application. For example, two fruits can be similar because of their color, size or taste. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each feature must be normalized, or one feature could end up dominating the distance calculation. **Similarity is measured in the range 0 to 1 [0,1].**

Two main considerations about similarity are as follows:

- Similarity = 1 if X = Y (Where X, Y are two objects)
- Similarity = 0 if X ≠ Y

That's all about similarity. So let's dive into implementing five most popular similarity distance measures.

Eucledian Distance:



Login

/wiki/Euclidean_distance) is the most commonly-used of our distance measures. Because of this, Eucledian distance is often just referred to as “distance”. When data is dense or continuous, this is the best proximity measure. The Euclidean distance between two points is the length of the path connecting them. This distance between two points is given by the Pythagorean theorem (http://en.wikipedia.org/wiki/Pythagorean_theorem).

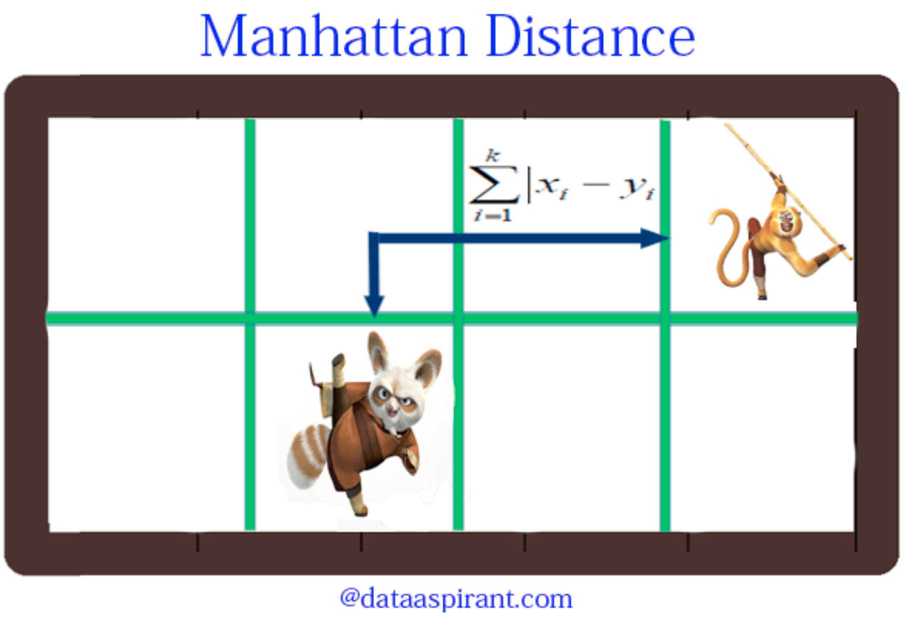
Euclidean distance implementation in Python:

```
#!/usr/bin/env python

from math import*

def euclidean_distance(x,y):

  return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))

print euclidean_distance([0,3,4,5],[7,6,3,-1])
```

Script output:

```
9.74679434481
[Finished in 0.0s]
```

Manhattan Distance



Login

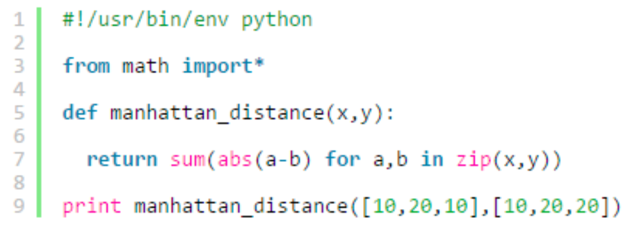
points is the sum of the absolute differences of their Cartesian coordinates. In simple language, it is the absolute sum of the difference between the x-coordinates and y-coordinates of each of the points. Suppose we have a point A and a point B. To find the manhattan distance between them, we just have to sum up the absolute variation along the x and y axes. We find manhattan distance between two points by measuring along axis at right angles.

In a plane with p1 at (x1, y1) and p2 at (x2, y2).

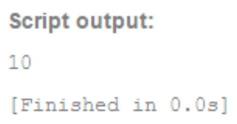
Manhattan distance = |x1 – x2| + |y1 – y2|

This Manhattan distance metric is also known as Manhattan length, rectilinear distance, L1 distance, L1 norm ,city block distance, Minkowski's L1 distance, taxi cab metric, or city block distance.

Manhattan distance implementation in python:



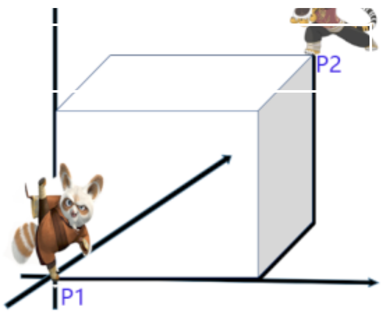
```
#!/usr/bin/env python

from math import*

def manhattan_distance(x,y):

  return sum(abs(a-b) for a,b in zip(x,y))

print manhattan_distance([10,20,10],[10,20,20])
```

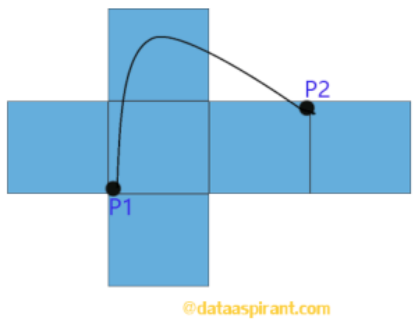


Script output:

```
10
[Finished in 0.0s]
```

Minkowski distance:





Login

The Minkowski distance is a generalized metric form of Euclidean distance and Manhattan distance. It looks like this:

$$d^{MKD}(i,j) = \lambda \sqrt{\sum_{k=0}^{n-1} |y_{i,k} - y_{j,k}|^{\lambda}}$$

In the equation d^MKD is the Minkowski distance between the data record i and j, k the index of a variable, n the total number of variables y and λ the order of the Minkowski metric. Although it is defined for any λ > 0, it is rarely used for values other than 1, 2 and ∞.

Different names for the Minkowski difference arise from the synonyms of other measures:

- λ = 1 is the Manhattan distance. Synonyms are L1-Norm, Taxicab or City-Block distance. For two vectors of ranked ordinal variables the Manhattan distance is sometimes called Foot-ruler distance.
- λ = 2 is the Euclidean distance. Synonyms are L2-Norm and Ruler distance. For two vectors of ranked ordinal variables the Euclidean distance is sometimes called Spear-man distance.
- λ = ∞ is the Chebyshev distance. Synonyms are Lmax-Norm or Chessboard distance.

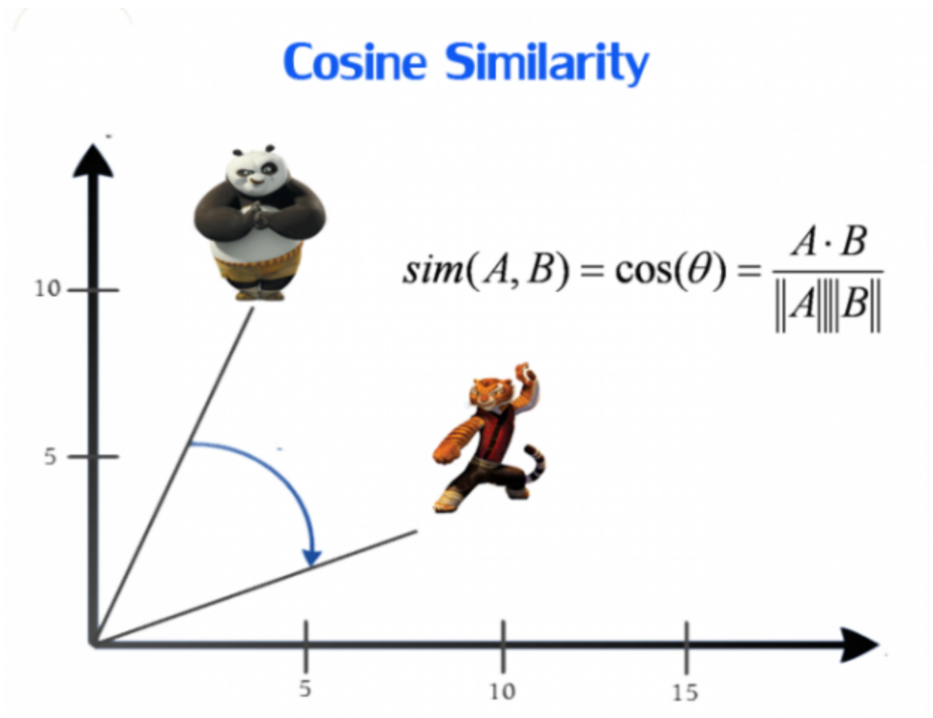
Login

```
#!/usr/bin/env python

from math import*
from decimal import Decimal

def nth_root(value, n_root):

 root_value = 1/float(n_root)
 return round (Decimal(value) ** Decimal(root_value),3)

def minkowski_distance(x,y,p_value):

 return nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_value)

print minkowski_distance([0,3,4,5],[7,6,3,-1],3)
```

Script output:

```
8.373

[Finished in 0.0s]
```

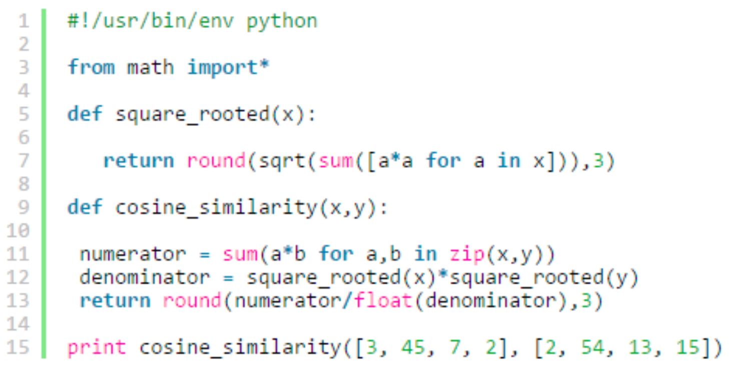
Cosine Similarity



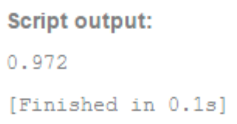
Cosine similarity metric finds the normalized dot product of the two attributes. By determining the cosine similarity, we will effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any other two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors.

Login

Cosine implementation in Python



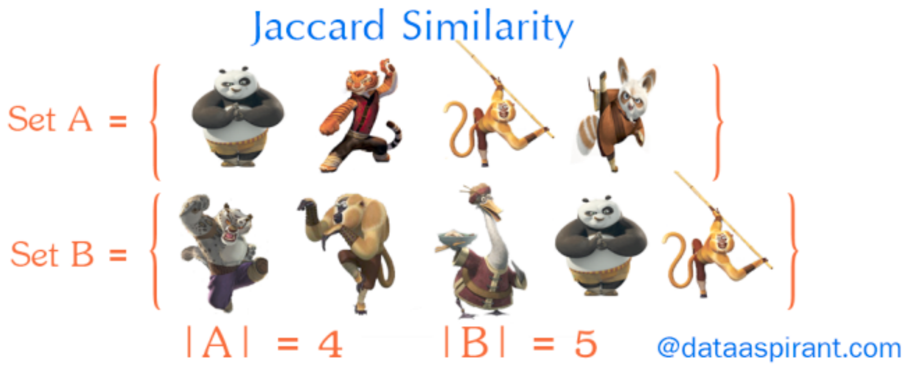
```
#!/usr/bin/env python

from math import*

def square_rooted(x):

   return round(sqrt(sum([a*a for a in x])),3)

def cosine_similarity(x,y):

 numerator = sum(a*b for a,b in zip(x,y))
 denominator = square_rooted(x)*square_rooted(y)
 return round(numerator/float(denominator),3)

print cosine_similarity([3, 45, 7, 2], [2, 54, 13, 15])
```



Script output:

```
0.972
[Finished in 0.1s]
```

Jaccard Similarity:



So far, we've discussed some metrics to find the similarity jaccard similarity to find similarity between *sets*. So first, let's learn the very basics of sets.

Login

Sets:

A set is (unordered) collection of objects {a,b,c}. We use the notation as elements separated by commas inside curly brackets { }. They are unordered so {a,b} = { b,a }.

Cardinality:

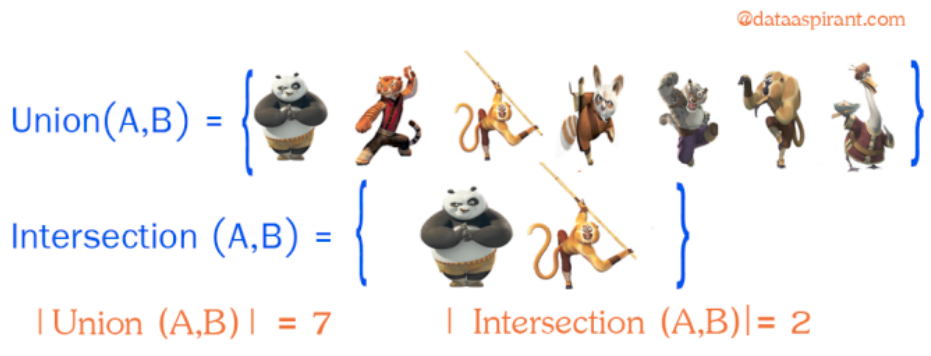
Cardinality of A (denoted by **|A|)** counts how many elements are in A.

Intersection:

Intersection between two sets A and B is denoted **A ∩ B** and reveals all items which are in both sets A,B.

Union:

Union between two sets A and B is denoted **A ∪ B** and reveals all items which are in either set.



Now going back to Jaccard similarity. The Jaccard similarity

Login

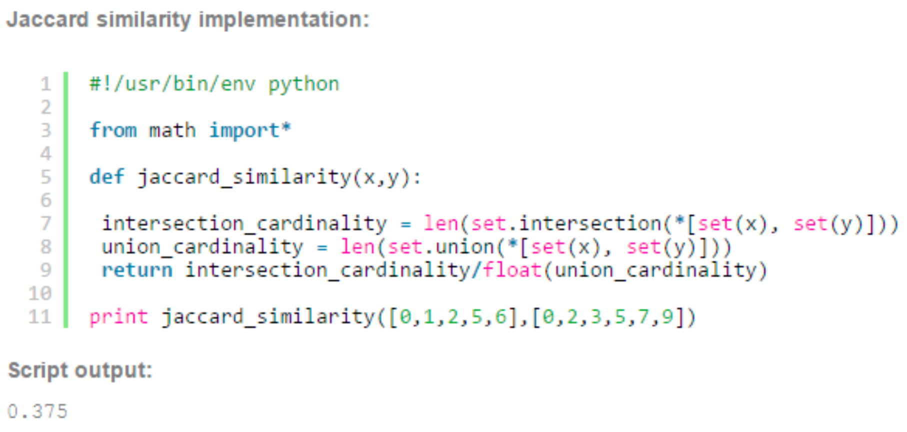
the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets. Suppose you want to find jaccard similarity between two sets A and B, it is the ration of cardinality of A ∩ B and A ∪ B.

$$\text{Jaccard Similarity } J(A,B) = |\text{Intersection }(A,B)| \,/\, |\text{Union }(A,B)|$$

$$= 2 \,/\, 7$$

$$= 0.286$$

@dataaspirant.com



Jaccard similarity implementation:

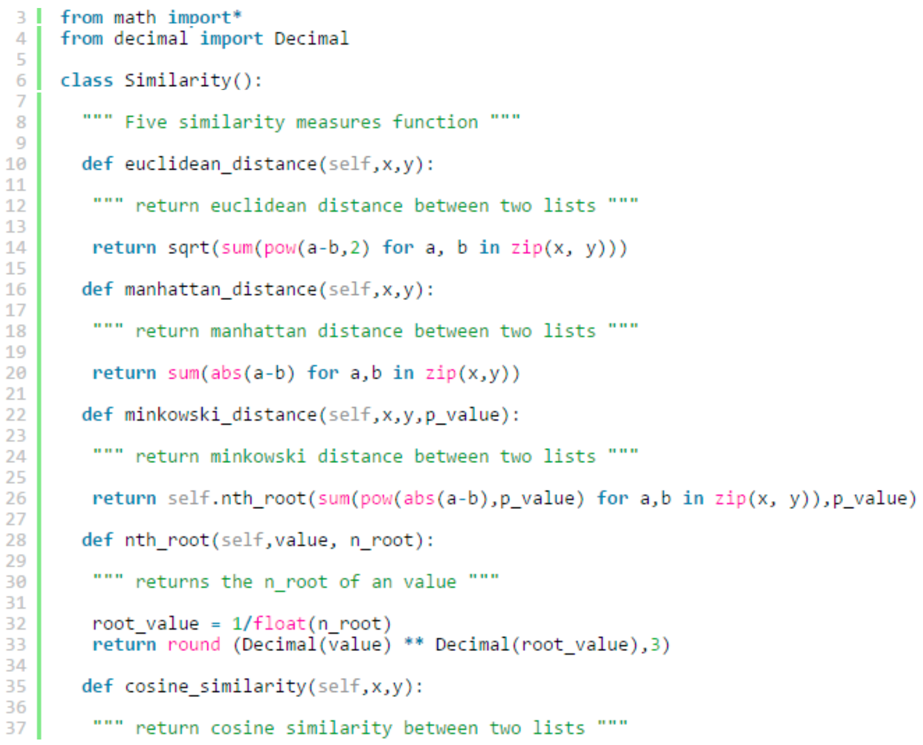
```
#!/usr/bin/env python

from math import*

def jaccard_similarity(x,y):

 intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
 union_cardinality = len(set.union(*[set(x), set(y)]))
 return intersection_cardinality/float(union_cardinality)

print jaccard_similarity([0,1,2,5,6],[0,2,3,5,7,9])
```

Script output:

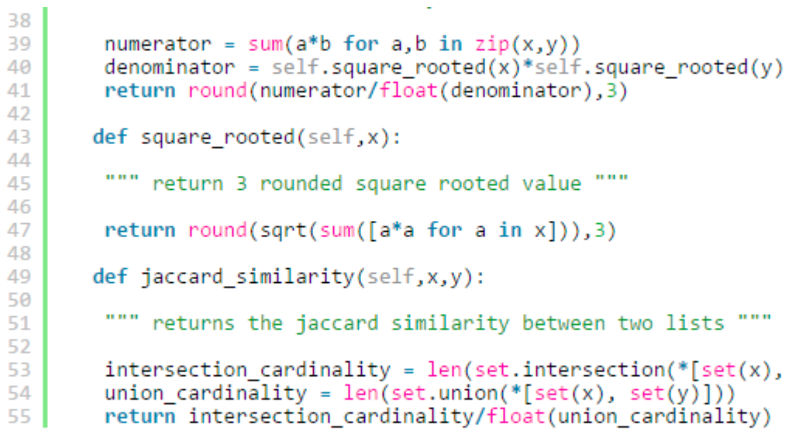
```
0.375
```

Implementation of all 5 similarity measure into one Similarity class:

Login

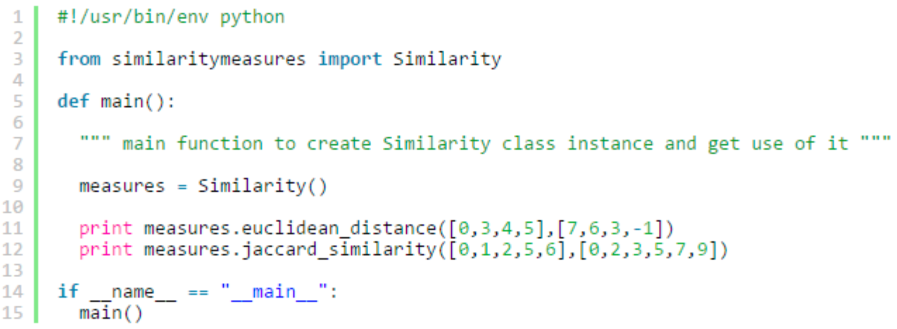


```
from math import*
from decimal import Decimal

class Similarity():

  """ Five similarity measures function """

  def euclidean_distance(self,x,y):

   """ return euclidean distance between two lists """

   return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))

  def manhattan_distance(self,x,y):

   """ return manhattan distance between two lists """

   return sum(abs(a-b) for a,b in zip(x,y))

  def minkowski_distance(self,x,y,p_value):

   """ return minkowski distance between two lists """

   return self.nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_value)

  def nth_root(self,value, n_root):

   """ returns the n_root of an value """

   root_value = 1/float(n_root)
   return round (Decimal(value) ** Decimal(root_value),3)

  def cosine_similarity(self,x,y):

   """ return cosine similarity between two lists """
```



```

   numerator = sum(a*b for a,b in zip(x,y))
   denominator = self.square_rooted(x)*self.square_rooted(y)
   return round(numerator/float(denominator),3)

  def square_rooted(self,x):

   """ return 3 rounded square rooted value """

   return round(sqrt(sum([a*a for a in x])),3)

  def jaccard_similarity(self,x,y):

   """ returns the jaccard similarity between two lists """

   intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
   union_cardinality = len(set.union(*[set(x), set(y)]))
   return intersection_cardinality/float(union_cardinality)
```

Using similarity class:



```
#!/usr/bin/env python

from similaritymeasures import Similarity

def main():

  """ main function to create Similarity class instance and get use of it """

  measures = Similarity()

  print measures.euclidean_distance([0,3,4,5],[7,6,3,-1])
  print measures.jaccard_similarity([0,1,2,5,6],[0,2,3,5,7,9])

if __name__ == "__main__":
  main()
```

(https://github.com/saimadhu-polamuri/DataAspirant_codes/tree/master/Similarity_measures), and on the Dataaspirant blog. (http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/)

Login

Photo credit: t3rmin4t0r (https://www.flickr.com/photos/t3rmin4t0r/2679917066/) / Foter (http://foter.com/) / CC BY (http://creativecommons.org/licenses/by/2.0/)

MORE FROM BIG DATA MADE SIMPLE

Big Data Posing Big Challenge for Pentagon (http://bigdata-madesimple.com/big-data-posing-big-challenge-for-pentagon/)

The Pentagon is projected to boost spending on so-called big data programs in the coming years, despite the…

In **Crime / Law (http://bigdata-madesimple.com/category/sectors/crime-law/)**

30 most influential papers in the world of big data (http://bigdata-madesimple.com/30-most-influential-papers-in-the-world-of-big-data/)

Here is a list of some of the most influer papers in the world of big data. We've…

In **Resources (http://bigdata-madesimple.com/category/resources/)** , Baiju NT (http://bigdata-madesimple.cor/contributers/baiju-nt/) on Jun 25

()

Login

(http://pages.sisense.com/dashboard-software-trial-download.html?source=big_data&cmp=footer)

OUR PARTNERS

(http://asiafintechbusiness.com/)

(http://events.insurancenexus.com/usa/?utm_source=BDMS&utm_medium=Listing&utm_campaign=4805)

(http://www.oreilly.com/pub/cpc/33439)

(http://ibexindia.com/)

About us (http://bigdata-madesimple.com/about-us)
Newsletter (http://www.bigdata-madesimple.com/newsletter/)
Archives (http://bigdata-madesimple.com/archives)
RSS feed (http://bigdata-madesimple.com/rss-feed)
Contributors (http://bigdata-madesimple.com/contributers)
Contact Us

Subscribe to our Newsletter

Your Email Addr **Subscribe**

Powered by

(http://www.crayondata.com/)