# Eucledian Distance,

This metric is the most common distance measure and is defined as the straight-line distance between two points in a multi-dimensional space. It works well with interval and ratio-scaled data.

### Importing required Libraries

import pandas as pd  
import numpy as np  
from sklearn.preprocessing import LabelEncoder  
from scipy.spatial.distance import euclidean

### Load Datasets

# Load datasets  
adult\_df = pd.read\_csv("../adult/adult\_trim.data", header=None) # No header  
titanic\_df = pd.read\_csv('../titanic/titanic\_trim.csv') # Has header  
  
# Rename columns for clarity  
adult\_df.columns = ["age", "workclass", "fnlwgt", "education", "education\_num",   
 "marital\_status", "occupation", "relationship", "race", "sex",   
 "capital\_gain", "capital\_loss", "hours\_per\_week", "native\_country", "income"]  
adult\_df.dropna(inplace=True)

adult\_df

age workclass fnlwgt education education\_num \  
0 39 State-gov 77516 Bachelors 13   
1 50 Self-emp-not-inc 83311 Bachelors 13   
2 38 Private 215646 HS-grad 9   
3 53 Private 234721 11th 7   
4 28 Private 338409 Bachelors 13   
.. ... ... ... ... ...   
95 29 Local-gov 115585 Some-college 10   
96 48 Self-emp-not-inc 191277 Doctorate 16   
97 37 Private 202683 Some-college 10   
98 48 Private 171095 Assoc-acdm 12   
99 32 Federal-gov 249409 HS-grad 9   
  
 marital\_status occupation relationship race sex \  
0 Never-married Adm-clerical Not-in-family White Male   
1 Married-civ-spouse Exec-managerial Husband White Male   
2 Divorced Handlers-cleaners Not-in-family White Male   
3 Married-civ-spouse Handlers-cleaners Husband Black Male   
4 Married-civ-spouse Prof-specialty Wife Black Female   
.. ... ... ... ... ...   
95 Never-married Handlers-cleaners Not-in-family White Male   
96 Married-civ-spouse Prof-specialty Husband White Male   
97 Married-civ-spouse Sales Husband White Male   
98 Divorced Exec-managerial Unmarried White Female   
99 Never-married Other-service Own-child Black Male   
  
 capital\_gain capital\_loss hours\_per\_week native\_country income   
0 2174 0 40 United-States <=50K   
1 0 0 13 United-States <=50K   
2 0 0 40 United-States <=50K   
3 0 0 40 United-States <=50K   
4 0 0 40 Cuba <=50K   
.. ... ... ... ... ...   
95 0 0 50 United-States <=50K   
96 0 1902 60 United-States >50K   
97 0 0 48 United-States >50K   
98 0 0 40 England <=50K   
99 0 0 40 United-States <=50K   
  
[100 rows x 15 columns]

titanic\_df

PassengerId Survived Pclass \  
0 1 0 3   
1 2 1 1   
2 3 1 3   
3 4 1 1   
4 5 0 3   
.. ... ... ...   
150 151 0 2   
151 152 1 1   
152 153 0 3   
153 154 0 3   
154 155 0 3   
  
 Name Sex Age SibSp \  
0 Braund, Mr. Owen Harris male 22.0 1   
1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1   
2 Heikkinen, Miss. Laina female 26.0 0   
3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1   
4 Allen, Mr. William Henry male 35.0 0   
.. ... ... ... ...   
150 Bateman, Rev. Robert James male 51.0 0   
151 Pears, Mrs. Thomas (Edith Wearne) female 22.0 1   
152 Meo, Mr. Alfonzo male 55.5 0   
153 van Billiard, Mr. Austin Blyler male 40.5 0   
154 Olsen, Mr. Ole Martin male NaN 0   
  
 Parch Ticket Fare Cabin Embarked   
0 0 A/5 21171 7.2500 NaN S   
1 0 PC 17599 71.2833 C85 C   
2 0 STON/O2. 3101282 7.9250 NaN S   
3 0 113803 53.1000 C123 S   
4 0 373450 8.0500 NaN S   
.. ... ... ... ... ...   
150 0 S.O.P. 1166 12.5250 NaN S   
151 0 113776 66.6000 C2 S   
152 0 A.5. 11206 8.0500 NaN S   
153 2 A/5. 851 14.5000 NaN S   
154 0 Fa 265302 7.3125 NaN S   
  
[155 rows x 12 columns]

### Select relevant columns from Adult dataset (mix of nominal and ratio-scaled)

adult\_df = adult\_df[["age", "workclass", "education", "education\_num", "sex"]]  
  
adult\_df

age workclass education education\_num sex  
0 39 State-gov Bachelors 13 Male  
1 50 Self-emp-not-inc Bachelors 13 Male  
2 38 Private HS-grad 9 Male  
3 53 Private 11th 7 Male  
4 28 Private Bachelors 13 Female  
.. ... ... ... ... ...  
95 29 Local-gov Some-college 10 Male  
96 48 Self-emp-not-inc Doctorate 16 Male  
97 37 Private Some-college 10 Male  
98 48 Private Assoc-acdm 12 Female  
99 32 Federal-gov HS-grad 9 Male  
  
[100 rows x 5 columns]

### Encode nominal attributes as integers for processing

label\_encoders = {}  
for column in adult\_df.columns:  
 if adult\_df[column].dtype == object:  
 le = LabelEncoder()  
 adult\_df[column] = le.fit\_transform(adult\_df[column])  
 label\_encoders[column] = le  
  
adult\_df

C:\Users\debat\AppData\Local\Temp\ipykernel\_11476\183426126.py:5: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 adult\_df[column] = le.fit\_transform(adult\_df[column])  
C:\Users\debat\AppData\Local\Temp\ipykernel\_11476\183426126.py:5: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 adult\_df[column] = le.fit\_transform(adult\_df[column])  
C:\Users\debat\AppData\Local\Temp\ipykernel\_11476\183426126.py:5: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 adult\_df[column] = le.fit\_transform(adult\_df[column])

age workclass education education\_num sex  
0 39 6 7 13 1  
1 50 5 7 13 1  
2 38 3 9 9 1  
3 53 3 1 7 1  
4 28 3 7 13 0  
.. ... ... ... ... ...  
95 29 2 12 10 1  
96 48 5 8 16 1  
97 37 3 12 10 1  
98 48 3 5 12 0  
99 32 1 9 9 1  
  
[100 rows x 5 columns]

### Clean and preprocess Titanic dataset

titanic\_df.dropna(inplace=True)  
titanic\_df

PassengerId Survived Pclass \  
1 2 1 1   
3 4 1 1   
6 7 0 1   
10 11 1 3   
11 12 1 1   
21 22 1 2   
23 24 1 1   
27 28 0 1   
52 53 1 1   
54 55 0 1   
62 63 0 1   
66 67 1 2   
75 76 0 3   
88 89 1 1   
92 93 0 1   
96 97 0 1   
97 98 1 1   
102 103 0 1   
110 111 0 1   
118 119 0 1   
123 124 1 2   
124 125 0 1   
136 137 1 1   
137 138 0 1   
139 140 0 1   
148 149 0 2   
151 152 1 1   
  
 Name Sex Age SibSp \  
1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1   
3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1   
6 McCarthy, Mr. Timothy J male 54.0 0   
10 Sandstrom, Miss. Marguerite Rut female 4.0 1   
11 Bonnell, Miss. Elizabeth female 58.0 0   
21 Beesley, Mr. Lawrence male 34.0 0   
23 Sloper, Mr. William Thompson male 28.0 0   
27 Fortune, Mr. Charles Alexander male 19.0 3   
52 Harper, Mrs. Henry Sleeper (Myna Haxtun) female 49.0 1   
54 Ostby, Mr. Engelhart Cornelius male 65.0 0   
62 Harris, Mr. Henry Birkhardt male 45.0 1   
66 Nye, Mrs. (Elizabeth Ramell) female 29.0 0   
75 Moen, Mr. Sigurd Hansen male 25.0 0   
88 Fortune, Miss. Mabel Helen female 23.0 3   
92 Chaffee, Mr. Herbert Fuller male 46.0 1   
96 Goldschmidt, Mr. George B male 71.0 0   
97 Greenfield, Mr. William Bertram male 23.0 0   
102 White, Mr. Richard Frasar male 21.0 0   
110 Porter, Mr. Walter Chamberlain male 47.0 0   
118 Baxter, Mr. Quigg Edmond male 24.0 0   
123 Webber, Miss. Susan female 32.5 0   
124 White, Mr. Percival Wayland male 54.0 0   
136 Newsom, Miss. Helen Monypeny female 19.0 0   
137 Futrelle, Mr. Jacques Heath male 37.0 1   
139 Giglio, Mr. Victor male 24.0 0   
148 Navratil, Mr. Michel ("Louis M Hoffman") male 36.5 0   
151 Pears, Mrs. Thomas (Edith Wearne) female 22.0 1   
  
 Parch Ticket Fare Cabin Embarked   
1 0 PC 17599 71.2833 C85 C   
3 0 113803 53.1000 C123 S   
6 0 17463 51.8625 E46 S   
10 1 PP 9549 16.7000 G6 S   
11 0 113783 26.5500 C103 S   
21 0 248698 13.0000 D56 S   
23 0 113788 35.5000 A6 S   
27 2 19950 263.0000 C23 C25 C27 S   
52 0 PC 17572 76.7292 D33 C   
54 1 113509 61.9792 B30 C   
62 0 36973 83.4750 C83 S   
66 0 C.A. 29395 10.5000 F33 S   
75 0 348123 7.6500 F G73 S   
88 2 19950 263.0000 C23 C25 C27 S   
92 0 W.E.P. 5734 61.1750 E31 S   
96 0 PC 17754 34.6542 A5 C   
97 1 PC 17759 63.3583 D10 D12 C   
102 1 35281 77.2875 D26 S   
110 0 110465 52.0000 C110 S   
118 1 PC 17558 247.5208 B58 B60 C   
123 0 27267 13.0000 E101 S   
124 1 35281 77.2875 D26 S   
136 2 11752 26.2833 D47 S   
137 0 113803 53.1000 C123 S   
139 0 PC 17593 79.2000 B86 C   
148 2 230080 26.0000 F2 S   
151 0 113776 66.6000 C2 S

### Select relevant columns from Titanic dataset (mix of nominal and ratio-scaled)

titanic\_df = titanic\_df[["Age", "Sex", "Pclass", "Fare", "Embarked"]]  
titanic\_df

Age Sex Pclass Fare Embarked  
1 38.0 female 1 71.2833 C  
3 35.0 female 1 53.1000 S  
6 54.0 male 1 51.8625 S  
10 4.0 female 3 16.7000 S  
11 58.0 female 1 26.5500 S  
21 34.0 male 2 13.0000 S  
23 28.0 male 1 35.5000 S  
27 19.0 male 1 263.0000 S  
52 49.0 female 1 76.7292 C  
54 65.0 male 1 61.9792 C  
62 45.0 male 1 83.4750 S  
66 29.0 female 2 10.5000 S  
75 25.0 male 3 7.6500 S  
88 23.0 female 1 263.0000 S  
92 46.0 male 1 61.1750 S  
96 71.0 male 1 34.6542 C  
97 23.0 male 1 63.3583 C  
102 21.0 male 1 77.2875 S  
110 47.0 male 1 52.0000 S  
118 24.0 male 1 247.5208 C  
123 32.5 female 2 13.0000 S  
124 54.0 male 1 77.2875 S  
136 19.0 female 1 26.2833 S  
137 37.0 male 1 53.1000 S  
139 24.0 male 1 79.2000 C  
148 36.5 male 2 26.0000 S  
151 22.0 female 1 66.6000 S

### Encode Nominal as Integers for processing

label\_encoders\_titanic = {}  
for column in titanic\_df.columns:  
 if titanic\_df[column].dtype == object:  
 le = LabelEncoder()  
 titanic\_df[column] = le.fit\_transform(titanic\_df[column])  
 label\_encoders[column] = le  
  
titanic\_df

C:\Users\debat\AppData\Local\Temp\ipykernel\_11476\3305425594.py:5: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 titanic\_df[column] = le.fit\_transform(titanic\_df[column])  
C:\Users\debat\AppData\Local\Temp\ipykernel\_11476\3305425594.py:5: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 titanic\_df[column] = le.fit\_transform(titanic\_df[column])

Age Sex Pclass Fare Embarked  
1 38.0 0 1 71.2833 0  
3 35.0 0 1 53.1000 1  
6 54.0 1 1 51.8625 1  
10 4.0 0 3 16.7000 1  
11 58.0 0 1 26.5500 1  
21 34.0 1 2 13.0000 1  
23 28.0 1 1 35.5000 1  
27 19.0 1 1 263.0000 1  
52 49.0 0 1 76.7292 0  
54 65.0 1 1 61.9792 0  
62 45.0 1 1 83.4750 1  
66 29.0 0 2 10.5000 1  
75 25.0 1 3 7.6500 1  
88 23.0 0 1 263.0000 1  
92 46.0 1 1 61.1750 1  
96 71.0 1 1 34.6542 0  
97 23.0 1 1 63.3583 0  
102 21.0 1 1 77.2875 1  
110 47.0 1 1 52.0000 1  
118 24.0 1 1 247.5208 0  
123 32.5 0 2 13.0000 1  
124 54.0 1 1 77.2875 1  
136 19.0 0 1 26.2833 1  
137 37.0 1 1 53.1000 1  
139 24.0 1 1 79.2000 0  
148 36.5 1 2 26.0000 1  
151 22.0 0 1 66.6000 1

### Combine the datasets into a list for further processing

# Combine the datasets into a list for further processing  
datasets = {  
 "Adult Dataset": adult\_df,  
 "Titanic Dataset": titanic\_df  
}

### Compute Eucledian Distance

def euclidean\_distance(a, b):  
 """Calculate the Euclidean Distance between two vectors."""  
 try:  
 return euclidean(a, b)  
 except Exception as e:  
 return np.nan  
  
# Function to create the Euclidean Distance matrix  
def calculate\_euclidean\_matrix(dataset):  
 n = len(dataset)  
 euclidean\_matrix = np.zeros((n, n))  
   
 for i in range(n):  
 for j in range(n):  
 euclidean\_matrix[i, j] = euclidean\_distance(dataset.iloc[i].values, dataset.iloc[j].values)  
   
 return pd.DataFrame(euclidean\_matrix)

### Calculate Eucledian Distance

#### For Adult Dataset

euclidean\_matrix\_adult = calculate\_euclidean\_matrix(adult\_df)  
euclidean\_matrix\_adult

0 1 2 3 4 5 \  
0 0.000000 11.045361 5.477226 16.643317 11.445523 4.898979   
1 11.045361 0.000000 12.961481 9.219544 22.113344 13.564660   
2 5.477226 12.961481 0.000000 17.117243 11.000000 5.291503   
3 16.643317 9.219544 17.117243 0.000000 26.419690 19.672316   
4 11.445523 22.113344 11.000000 26.419690 0.000000 9.539392   
.. ... ... ... ... ... ...   
95 12.247449 22.000000 9.591663 26.589472 6.082763 9.273618   
96 9.591663 3.741657 12.409674 12.609520 20.371549 11.575837   
97 6.855655 14.387495 3.316625 19.646883 10.770330 4.582576   
98 9.797959 3.741657 11.224972 8.185353 20.124612 12.247449   
99 9.695360 18.973666 6.324555 22.649503 6.403124 7.483315   
  
 6 7 8 9 ... 90 91 \  
0 13.527749 13.784049 9.165151 4.242641 ... 18.384776 6.928203   
1 8.888194 4.898979 19.390719 8.246211 ... 7.615773 14.422205   
2 12.767145 14.142136 8.717798 6.000000 ... 19.339080 3.464102   
3 5.477226 8.544004 24.799194 13.892444 ... 7.549834 19.672316   
4 22.671568 24.515301 4.358899 14.035669 ... 29.103264 10.723805   
.. ... ... ... ... ... ... ...   
95 22.158520 23.409400 5.099020 14.282857 ... 28.670542 8.124038   
96 11.958261 8.124038 17.378147 7.071068 ... 10.677078 13.341664   
97 15.297059 15.459625 7.549834 7.681146 ... 20.904545 1.000000   
98 7.141428 6.782330 17.832555 6.480741 ... 9.165151 13.190906   
99 18.303005 20.396078 5.656854 11.135529 ... 25.337719 6.324555   
  
 92 93 94 95 96 97 \  
0 12.845233 10.535654 6.403124 12.247449 9.591663 6.855655   
1 22.869193 20.615528 16.278821 22.000000 3.741657 14.387495   
2 10.535654 8.062258 6.082763 9.591663 12.409674 3.316625   
3 27.495454 24.454039 20.832667 26.589472 12.609520 19.646883   
4 5.830952 4.898979 6.164414 6.082763 20.371549 10.770330   
.. ... ... ... ... ... ...   
95 1.732051 3.605551 7.681146 0.000000 20.542639 8.062258   
96 21.377558 19.467922 14.662878 20.542639 0.000000 13.304135   
97 9.055385 7.745967 6.633250 8.062258 13.304135 0.000000   
98 21.283797 18.681542 14.247807 20.396078 5.477226 13.228757   
99 5.567764 3.000000 5.000000 4.472136 17.944358 6.244998   
  
 98 99   
0 9.797959 9.695360   
1 3.741657 18.973666   
2 11.224972 6.324555   
3 8.185353 22.649503   
4 20.124612 6.403124   
.. ... ...   
95 20.396078 4.472136   
96 5.477226 17.944358   
97 13.228757 6.244998   
98 0.000000 16.911535   
99 16.911535 0.000000   
  
[100 rows x 100 columns]

#### For Titanic Dataset

euclidean\_matrix\_titanic = calculate\_euclidean\_matrix(titanic\_df)  
euclidean\_matrix\_titanic

0 1 2 3 4 5 \  
0 0.000000 18.456229 25.202529 64.345448 49.010898 58.446070   
1 18.456229 0.000000 19.066500 47.853527 35.126948 40.137389   
2 25.202529 19.066500 0.000000 61.166996 25.646104 43.718347   
3 64.345448 47.853527 61.166996 0.000000 54.927429 30.260370   
4 49.010898 35.126948 25.646104 54.927429 0.000000 27.597147   
5 58.446070 40.137389 43.718347 30.260370 27.597147 0.000000   
6 37.181239 18.967340 30.720212 30.568611 31.322556 23.307724   
7 192.661083 210.511306 214.018793 246.766469 239.646829 250.451592   
8 12.274275 27.483433 25.403794 75.056678 50.989726 65.493595   
9 28.575624 31.318368 14.978238 76.007933 36.141779 57.982429   
10 14.129315 31.994384 32.868680 78.389417 58.399106 71.335304   
11 61.462261 43.032081 48.351385 25.776734 33.160255 5.678908   
12 64.993822 46.590798 52.912618 22.888917 38.094750 10.517723   
13 192.305208 210.242741 213.403477 247.039855 239.026364 250.245879   
14 12.968336 13.682311 12.276916 61.212953 36.659114 49.657131   
15 49.312179 40.475271 24.210031 69.407156 15.384345 42.894107   
16 16.994282 15.850322 33.077990 50.438051 50.811917 51.565089   
17 18.084535 27.964891 41.658500 62.967016 62.803614 65.596362   
18 21.327111 12.091733 7.001350 55.678452 27.743513 41.121770   
19 176.795521 194.736867 197.947393 231.698601 223.575702 234.738164   
20 58.559312 40.190297 44.435840 28.756564 28.893814 1.802776   
21 17.147898 30.773936 25.425000 78.586546 50.904753 67.334112   
22 48.856934 31.227158 43.362374 17.911997 39.000912 20.085967   
23 18.265607 2.236068 17.044982 49.182924 33.865949 40.224495   
24 16.114408 28.358597 40.599740 65.667724 62.689892 66.965962   
25 45.341231 27.178300 31.242902 33.834007 21.553480 13.238202   
26 16.701296 18.741665 35.244771 53.084932 53.851671 54.945063   
  
 6 7 8 9 ... 17 \  
0 37.181239 192.661083 12.274275 28.575624 ... 18.084535   
1 18.967340 210.511306 27.483433 31.318368 ... 27.964891   
2 30.720212 214.018793 25.403794 14.978238 ... 41.658500   
3 30.568611 246.766469 75.056678 76.007933 ... 62.967016   
4 31.322556 239.646829 50.989726 36.141779 ... 62.803614   
5 23.307724 250.451592 65.493595 57.982429 ... 65.596362   
6 0.000000 227.677952 46.290895 45.509867 ... 42.369743   
7 227.677952 0.000000 188.676472 206.219209 ... 185.723269   
8 46.290895 188.676472 0.000000 21.784455 ... 28.041250   
9 45.509867 206.219209 21.784455 0.000000 ... 46.597683   
10 50.897943 181.397976 7.969054 29.378043 ... 24.784777   
11 25.059928 252.701899 69.197593 62.841929 ... 67.279790   
12 28.082423 255.428312 73.170594 67.503052 ... 69.780953   
13 227.557136 4.123106 188.079268 205.366409 ... 185.725961   
14 31.356110 203.623011 15.903872 19.043286 ... 29.742439   
15 43.019942 234.193946 47.490058 27.975983 ... 65.716043   
16 28.321103 199.684272 29.253734 42.022636 ... 14.107537   
17 42.369743 185.723269 28.041250 46.597683 ... 0.000000   
18 25.164459 212.849712 24.850218 20.605447 ... 36.269238   
19 212.060887 16.297412 172.614515 190.017592 ... 170.262669   
20 22.989128 250.368229 65.845736 58.806565 ... 65.323293   
21 49.215802 188.981832 5.226060 18.877077 ... 33.000000   
22 12.920819 236.718812 58.700842 58.242573 ... 51.053192   
23 19.767650 210.670382 26.539388 29.391158 ... 29.000606   
24 43.894077 183.870715 25.141696 44.469720 ... 3.695627   
25 12.786712 237.647323 52.275250 45.921159 ... 53.587850   
26 31.689273 196.425457 28.854821 43.270681 ... 10.780661   
  
 18 19 20 21 22 23 \  
0 21.327111 176.795521 58.559312 17.147898 48.856934 18.265607   
1 12.091733 194.736867 40.190297 30.773936 31.227158 2.236068   
2 7.001350 197.947393 44.435840 25.425000 43.362374 17.044982   
3 55.678452 231.698601 28.756564 78.586546 17.911997 49.182924   
4 27.743513 223.575702 28.893814 50.904753 39.000912 33.865949   
5 41.121770 234.738164 1.802776 67.334112 20.085967 40.224495   
6 25.164459 212.060887 22.989128 49.215802 12.920819 19.767650   
7 212.849712 16.297412 250.368229 188.981832 236.718812 210.670382   
8 24.850218 172.614515 65.845736 5.226060 58.700842 26.539388   
9 20.605447 190.017592 58.806565 18.877077 58.242573 29.391158   
10 31.538478 165.387498 71.588935 10.921774 62.832241 31.410836   
11 45.257596 237.079859 4.301163 71.327205 18.711295 43.367730   
12 49.547174 239.883306 9.320542 75.461125 19.702788 47.050000   
13 212.362897 15.575803 250.182433 188.284712 236.750493 210.368748   
14 9.229335 187.642632 50.050780 17.989237 44.129703 12.091552   
15 29.628985 217.993554 44.205818 45.908586 52.688442 38.694283   
16 26.570867 184.165215 51.275807 34.000333 37.316962 17.384842   
17 36.269238 170.262669 65.323293 33.000000 51.053192 29.000606   
18 0.000000 196.871489 41.632319 26.238477 38.030891 10.060318   
19 196.871489 0.000000 234.681179 172.859412 221.298512 194.857506   
20 41.632319 234.681179 0.000000 67.802158 18.965655 40.376478   
21 26.238477 172.859412 67.802158 0.000000 61.866214 29.564086   
22 38.030891 221.298512 18.965655 61.866214 0.000000 32.313084   
23 10.060318 194.857506 40.376478 29.564086 32.313084 0.000000   
24 35.634814 168.320800 66.765934 30.077527 53.171206 29.175503   
25 28.057976 221.877702 13.638182 54.200163 17.559335 27.123053   
26 28.968258 180.937381 54.627923 33.752373 40.428162 20.205197   
  
 24 25 26   
0 16.114408 45.341231 16.701296   
1 28.358597 27.178300 18.741665   
2 40.599740 31.242902 35.244771   
3 65.667724 33.834007 53.084932   
4 62.689892 21.553480 53.851671   
5 66.965962 13.238202 54.945063   
6 43.894077 12.786712 31.689273   
7 183.870715 237.647323 196.425457   
8 25.141696 52.275250 28.854821   
9 44.469720 45.921159 43.270681   
10 21.454035 58.108740 28.544100   
11 68.903483 17.248188 56.543877   
12 71.591916 21.678849 59.068625   
13 183.808161 237.388395 196.402546   
14 28.458753 36.449014 24.625812   
15 64.755913 35.596983 58.510975   
16 15.873231 39.747863 3.675407   
17 3.695627 53.587850 10.780661   
18 35.634814 28.057976 28.968258   
19 168.320800 221.877702 180.937381   
20 66.765934 13.638182 54.627923   
21 30.077527 54.200163 33.752373   
22 53.171206 17.559335 40.428162   
23 29.175503 27.123053 20.205197   
24 0.000000 54.667083 12.835887   
25 54.667083 0.000000 43.134789   
26 12.835887 43.134789 0.000000   
  
[27 rows x 27 columns]

### Explanation

Euclidean Distance Calculation: This metric is a straightforward calculation of the straight-line distance between two points in multi-dimensional space. It's suitable for interval and ratio-scaled data.

Handling Different Data Types: While Euclidean Distance works best with interval and ratio-scaled data, it may not be meaningful for nominal or ordinal data without preprocessing or encoding.

### Observation and Analysis

The resulting matrices will represent the pairwise Euclidean distances between data points. A smaller value indicates that the data points are closer to each other, while a larger value indicates they are further apart.

Euclidean Distance is sensitive to the scale of the data, so if attributes have different units or scales, standardization or normalization is often necessary before applying this metric.