# Supremum Distance

also known as the Chebyshev Distance. This metric measures the maximum absolute difference between the coordinates of two points. It is particularly useful in scenarios where the most significant difference between any single dimension matters the most.

### Importing required Libraries

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

### 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\_5048\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\_5048\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\_5048\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\_5048\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\_5048\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 Supremum Distance

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

### Calculate Supremum Distance

#### For Adult Dataset

supremum\_matrix\_adult = calculate\_supremum\_matrix(adult\_df)  
supremum\_matrix\_adult

0 1 2 3 4 5 6 7 8 9 ... 90 \  
0 0.0 11.0 4.0 14.0 11.0 3.0 10.0 13.0 8.0 3.0 ... 18.0   
1 11.0 0.0 12.0 6.0 22.0 13.0 8.0 4.0 19.0 8.0 ... 7.0   
2 4.0 12.0 0.0 15.0 10.0 5.0 11.0 14.0 7.0 4.0 ... 19.0   
3 14.0 6.0 15.0 0.0 25.0 16.0 4.0 8.0 22.0 11.0 ... 5.0   
4 11.0 22.0 10.0 25.0 0.0 9.0 21.0 24.0 3.0 14.0 ... 29.0   
.. ... ... ... ... ... ... ... ... ... ... ... ...   
95 10.0 21.0 9.0 24.0 5.0 8.0 20.0 23.0 4.0 13.0 ... 28.0   
96 9.0 3.0 10.0 9.0 20.0 11.0 11.0 7.0 17.0 6.0 ... 9.0   
97 5.0 13.0 3.0 16.0 9.0 4.0 12.0 15.0 6.0 5.0 ... 20.0   
98 9.0 2.0 10.0 5.0 20.0 11.0 7.0 4.0 17.0 6.0 ... 9.0   
99 7.0 18.0 6.0 21.0 4.0 5.0 17.0 20.0 5.0 10.0 ... 25.0   
  
 91 92 93 94 95 96 97 98 99   
0 5.0 11.0 9.0 5.0 10.0 9.0 5.0 9.0 7.0   
1 13.0 22.0 20.0 16.0 21.0 3.0 13.0 2.0 18.0   
2 3.0 10.0 8.0 4.0 9.0 10.0 3.0 10.0 6.0   
3 16.0 25.0 23.0 19.0 24.0 9.0 16.0 5.0 21.0   
4 9.0 5.0 4.0 6.0 5.0 20.0 9.0 20.0 4.0   
.. ... ... ... ... ... ... ... ... ...   
95 8.0 1.0 3.0 5.0 0.0 19.0 8.0 19.0 3.0   
96 11.0 20.0 18.0 14.0 19.0 0.0 11.0 4.0 16.0   
97 1.0 9.0 7.0 5.0 8.0 11.0 0.0 11.0 5.0   
98 11.0 20.0 18.0 14.0 19.0 4.0 11.0 0.0 16.0   
99 5.0 4.0 2.0 4.0 3.0 16.0 5.0 16.0 0.0   
  
[100 rows x 100 columns]

#### For Titanic Dataset

supremum\_matrix\_titanic = calculate\_supremum\_matrix(titanic\_df)  
supremum\_matrix\_titanic

0 1 2 3 4 5 6 \  
0 0.0000 18.1833 19.4208 54.5833 44.7333 58.2833 35.7833   
1 18.1833 0.0000 19.0000 36.4000 26.5500 40.1000 17.6000   
2 19.4208 19.0000 0.0000 50.0000 25.3125 38.8625 26.0000   
3 54.5833 36.4000 50.0000 0.0000 54.0000 30.0000 24.0000   
4 44.7333 26.5500 25.3125 54.0000 0.0000 24.0000 30.0000   
5 58.2833 40.1000 38.8625 30.0000 24.0000 0.0000 22.5000   
6 35.7833 17.6000 26.0000 24.0000 30.0000 22.5000 0.0000   
7 191.7167 209.9000 211.1375 246.3000 236.4500 250.0000 227.5000   
8 11.0000 23.6292 24.8667 60.0292 50.1792 63.7292 41.2292   
9 27.0000 30.0000 11.0000 61.0000 35.4292 48.9792 37.0000   
10 12.1917 30.3750 31.6125 66.7750 56.9250 70.4750 47.9750   
11 60.7833 42.6000 41.3625 25.0000 29.0000 5.0000 25.0000   
12 63.6333 45.4500 44.2125 21.0000 33.0000 9.0000 27.8500   
13 191.7167 209.9000 211.1375 246.3000 236.4500 250.0000 227.5000   
14 10.1083 11.0000 9.3125 44.4750 34.6250 48.1750 25.6750   
15 36.6291 36.0000 17.2083 67.0000 13.0000 37.0000 43.0000   
16 15.0000 12.0000 31.0000 46.6583 36.8083 50.3583 27.8583   
17 17.0000 24.1875 33.0000 60.5875 50.7375 64.2875 41.7875   
18 19.2833 12.0000 7.0000 43.0000 25.4500 39.0000 19.0000   
19 176.2375 194.4208 195.6583 230.8208 220.9708 234.5208 212.0208   
20 58.2833 40.1000 38.8625 28.5000 25.5000 1.5000 22.5000   
21 16.0000 24.1875 25.4250 60.5875 50.7375 64.2875 41.7875   
22 45.0000 26.8167 35.0000 15.0000 39.0000 15.0000 9.2167   
23 18.1833 2.0000 17.0000 36.4000 26.5500 40.1000 17.6000   
24 14.0000 26.1000 30.0000 62.5000 52.6500 66.2000 43.7000   
25 45.2833 27.1000 25.8625 32.5000 21.5000 13.0000 9.5000   
26 16.0000 13.5000 32.0000 49.9000 40.0500 53.6000 31.1000   
  
 7 8 9 ... 17 18 19 20 \  
0 191.7167 11.0000 27.0000 ... 17.0000 19.2833 176.2375 58.2833   
1 209.9000 23.6292 30.0000 ... 24.1875 12.0000 194.4208 40.1000   
2 211.1375 24.8667 11.0000 ... 33.0000 7.0000 195.6583 38.8625   
3 246.3000 60.0292 61.0000 ... 60.5875 43.0000 230.8208 28.5000   
4 236.4500 50.1792 35.4292 ... 50.7375 25.4500 220.9708 25.5000   
5 250.0000 63.7292 48.9792 ... 64.2875 39.0000 234.5208 1.5000   
6 227.5000 41.2292 37.0000 ... 41.7875 19.0000 212.0208 22.5000   
7 0.0000 186.2708 201.0208 ... 185.7125 211.0000 15.4792 250.0000   
8 186.2708 0.0000 16.0000 ... 28.0000 24.7292 170.7916 63.7292   
9 201.0208 16.0000 0.0000 ... 44.0000 18.0000 185.5416 48.9792   
10 179.5250 6.7458 21.4958 ... 24.0000 31.4750 164.0458 70.4750   
11 252.5000 66.2292 51.4792 ... 66.7875 41.5000 237.0208 3.5000   
12 255.3500 69.0792 54.3292 ... 69.6375 44.3500 239.8708 7.5000   
13 4.0000 186.2708 201.0208 ... 185.7125 211.0000 15.4792 250.0000   
14 201.8250 15.5542 19.0000 ... 25.0000 9.1750 186.3458 48.1750   
15 228.3458 42.0750 27.3250 ... 50.0000 24.0000 212.8666 38.5000   
16 199.6417 26.0000 42.0000 ... 13.9292 24.0000 184.1625 50.3583   
17 185.7125 28.0000 44.0000 ... 0.0000 26.0000 170.2333 64.2875   
18 211.0000 24.7292 18.0000 ... 26.0000 0.0000 195.5208 39.0000   
19 15.4792 170.7916 185.5416 ... 170.2333 195.5208 0.0000 234.5208   
20 250.0000 63.7292 48.9792 ... 64.2875 39.0000 234.5208 0.0000   
21 185.7125 5.0000 15.3083 ... 33.0000 25.2875 170.2333 64.2875   
22 236.7167 50.4459 46.0000 ... 51.0042 28.0000 221.2375 13.5000   
23 209.9000 23.6292 28.0000 ... 24.1875 10.0000 194.4208 40.1000   
24 183.8000 25.0000 41.0000 ... 3.0000 27.2000 168.3208 66.2000   
25 237.0000 50.7292 35.9792 ... 51.2875 26.0000 221.5208 13.0000   
26 196.4000 27.0000 43.0000 ... 10.6875 25.0000 180.9208 53.6000   
  
 21 22 23 24 25 26   
0 16.0000 45.0000 18.1833 14.0000 45.2833 16.0000   
1 24.1875 26.8167 2.0000 26.1000 27.1000 13.5000   
2 25.4250 35.0000 17.0000 30.0000 25.8625 32.0000   
3 60.5875 15.0000 36.4000 62.5000 32.5000 49.9000   
4 50.7375 39.0000 26.5500 52.6500 21.5000 40.0500   
5 64.2875 15.0000 40.1000 66.2000 13.0000 53.6000   
6 41.7875 9.2167 17.6000 43.7000 9.5000 31.1000   
7 185.7125 236.7167 209.9000 183.8000 237.0000 196.4000   
8 5.0000 50.4459 23.6292 25.0000 50.7292 27.0000   
9 15.3083 46.0000 28.0000 41.0000 35.9792 43.0000   
10 9.0000 57.1917 30.3750 21.0000 57.4750 23.0000   
11 66.7875 15.7833 42.6000 68.7000 15.5000 56.1000   
12 69.6375 18.6333 45.4500 71.5500 18.3500 58.9500   
13 185.7125 236.7167 209.9000 183.8000 237.0000 196.4000   
14 16.1125 34.8917 9.0000 22.0000 35.1750 24.0000   
15 42.6333 52.0000 34.0000 47.0000 34.5000 49.0000   
16 31.0000 37.0750 14.0000 15.8417 37.3583 3.2417   
17 33.0000 51.0042 24.1875 3.0000 51.2875 10.6875   
18 25.2875 28.0000 10.0000 27.2000 26.0000 25.0000   
19 170.2333 221.2375 194.4208 168.3208 221.5208 180.9208   
20 64.2875 13.5000 40.1000 66.2000 13.0000 53.6000   
21 0.0000 51.0042 24.1875 30.0000 51.2875 32.0000   
22 51.0042 0.0000 26.8167 52.9167 17.5000 40.3167   
23 24.1875 26.8167 0.0000 26.1000 27.1000 15.0000   
24 30.0000 52.9167 26.1000 0.0000 53.2000 12.6000   
25 51.2875 17.5000 27.1000 53.2000 0.0000 40.6000   
26 32.0000 40.3167 15.0000 12.6000 40.6000 0.0000   
  
[27 rows x 27 columns]

### Explanation

Supremum Distance Calculation: This metric considers the largest difference between any single dimension of two data points. It can be thought of as the distance you would move in a chessboard when the king moves, as it captures the maximum deviation across all dimensions.

Handling Different Data Types: Supremum Distance works well with interval and ratio-scaled data, and can be applied to ordinal data as well. It's particularly useful when the most significant difference in any single attribute is of primary concern.

### Observation and Analysis

The resulting matrices will show the pairwise Supremum distances between data points. A smaller value indicates that the largest difference between any dimension of the data points is small, while a larger value indicates a significant difference in at least one dimension.

Supremum Distance is useful in scenarios where outliers or extreme differences in one dimension are more significant than cumulative differences across all dimensions.