# Manhattan Distance,

Also known as the L1 norm or taxicab distance, this metric measures the distance between two points by summing the absolute differences of their corresponding coordinates. It’s useful for measuring distance in grid-like paths (e.g., city blocks).

### Importing required Libraries

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

### 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\_8272\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\_8272\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\_8272\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\_8272\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\_8272\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 Manhattan Distance

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

### Calculate Manhattan Distance

#### For Adult Dataset

manhattan\_matrix\_adult = calculate\_manhattan\_matrix(adult\_df)  
manhattan\_matrix\_adult

0 1 2 3 4 5 6 7 8 9 ... 90 \  
0 0.0 12.0 10.0 29.0 15.0 10.0 25.0 20.0 16.0 6.0 ... 24.0   
1 12.0 0.0 20.0 17.0 25.0 20.0 15.0 8.0 26.0 10.0 ... 12.0   
2 10.0 20.0 0.0 25.0 17.0 8.0 21.0 16.0 14.0 10.0 ... 24.0   
3 29.0 17.0 25.0 0.0 38.0 33.0 10.0 13.0 39.0 23.0 ... 13.0   
4 15.0 25.0 17.0 38.0 0.0 13.0 32.0 33.0 7.0 15.0 ... 33.0   
.. ... ... ... ... ... ... ... ... ... ... ... ...   
95 22.0 32.0 14.0 39.0 11.0 16.0 35.0 30.0 10.0 22.0 ... 36.0   
96 14.0 6.0 20.0 23.0 27.0 18.0 19.0 12.0 24.0 12.0 ... 18.0   
97 13.0 23.0 5.0 30.0 18.0 7.0 26.0 21.0 13.0 13.0 ... 27.0   
98 16.0 8.0 18.0 15.0 23.0 18.0 9.0 14.0 24.0 10.0 ... 12.0   
99 18.0 28.0 8.0 33.0 13.0 14.0 29.0 24.0 10.0 18.0 ... 32.0   
  
 91 92 93 94 95 96 97 98 99   
0 14.0 23.0 19.0 9.0 22.0 14.0 13.0 16.0 18.0   
1 24.0 33.0 29.0 19.0 32.0 6.0 23.0 8.0 28.0   
2 6.0 15.0 9.0 11.0 14.0 20.0 5.0 18.0 8.0   
3 31.0 40.0 34.0 32.0 39.0 23.0 30.0 15.0 33.0   
4 17.0 8.0 8.0 8.0 11.0 27.0 18.0 23.0 13.0   
.. ... ... ... ... ... ... ... ... ...   
95 10.0 3.0 7.0 13.0 0.0 32.0 9.0 30.0 8.0   
96 24.0 33.0 29.0 21.0 32.0 0.0 23.0 10.0 28.0   
97 1.0 10.0 12.0 12.0 9.0 23.0 0.0 21.0 11.0   
98 20.0 29.0 25.0 19.0 30.0 10.0 21.0 0.0 26.0   
99 12.0 11.0 5.0 9.0 8.0 28.0 11.0 26.0 0.0   
  
[100 rows x 100 columns]

#### For Titanic Dataset

manhattan\_matrix\_titanic = calculate\_manhattan\_matrix(titanic\_df)  
manhattan\_matrix\_titanic

0 1 2 3 4 5 6 \  
0 0.0000 22.1833 37.4208 91.5833 65.7333 65.2833 47.7833   
1 22.1833 0.0000 21.2375 69.4000 49.5500 43.1000 25.6000   
2 37.4208 21.2375 0.0000 88.1625 30.3125 59.8625 42.3625   
3 91.5833 69.4000 88.1625 0.0000 65.8500 35.7000 45.8000   
4 65.7333 49.5500 30.3125 65.8500 0.0000 39.5500 39.9500   
5 65.2833 43.1000 59.8625 35.7000 39.5500 0.0000 29.5000   
6 47.7833 25.6000 42.3625 45.8000 39.9500 29.5000 0.0000   
7 212.7167 226.9000 246.1375 264.3000 276.4500 266.0000 236.5000   
8 16.4459 38.6292 31.8667 108.0292 60.1792 81.7292 64.2292   
9 37.3041 40.8792 22.1167 110.2792 44.4292 81.9792 64.4792   
10 21.1917 41.3750 40.6125 110.7750 70.9250 82.4750 64.9750   
11 71.7833 49.6000 68.3625 32.2000 46.0500 8.5000 28.0000   
12 80.6333 58.4500 75.2125 31.0500 54.9000 15.3500 32.8500   
13 207.7167 221.9000 243.1375 267.3000 271.4500 263.0000 233.5000   
14 20.1083 20.0750 17.3125 89.4750 47.6250 61.1750 43.6750   
15 70.6291 56.4458 35.2083 88.9542 23.1042 60.6542 44.8458   
16 23.9250 24.2583 43.4958 69.6583 73.8083 63.3583 33.8583   
17 25.0042 39.1875 58.4250 80.5875 88.7375 78.2875 48.7875   
18 30.2833 14.1000 7.1375 81.3000 37.4500 53.0000 35.5000   
19 191.2375 207.4208 226.6583 254.8208 256.9708 246.5208 217.0208   
20 65.7833 43.6000 62.3625 33.2000 40.0500 2.5000 29.0000   
21 24.0042 44.1875 25.4250 113.5875 55.7375 85.2875 67.7875   
22 65.0000 42.8167 61.5792 26.5833 39.2667 30.2833 19.2167   
23 21.1833 3.0000 18.2375 72.4000 48.5500 44.1000 26.6000   
24 22.9167 39.1000 58.3375 86.5000 88.6500 78.2000 48.7000   
25 49.7833 30.6000 44.3625 43.8000 24.0500 15.5000 19.0000   
26 21.6833 26.5000 47.7375 69.9000 76.0500 67.6000 38.1000   
  
 7 8 9 ... 17 18 19 20 \  
0 212.7167 16.4459 37.3041 ... 25.0042 30.2833 191.2375 65.7833   
1 226.9000 38.6292 40.8792 ... 39.1875 14.1000 207.4208 43.6000   
2 246.1375 31.8667 22.1167 ... 58.4250 7.1375 226.6583 62.3625   
3 264.3000 108.0292 110.2792 ... 80.5875 81.3000 254.8208 33.2000   
4 276.4500 60.1792 44.4292 ... 88.7375 37.4500 256.9708 40.0500   
5 266.0000 81.7292 81.9792 ... 78.2875 53.0000 246.5208 2.5000   
6 236.5000 64.2292 64.4792 ... 48.7875 35.5000 217.0208 29.0000   
7 0.0000 218.2708 248.0208 ... 187.7125 239.0000 21.4792 265.5000   
8 218.2708 0.0000 31.7500 ... 30.5583 28.7292 196.7916 82.2292   
9 248.0208 31.7500 0.0000 ... 60.3083 28.9792 226.5416 84.4792   
10 205.5250 12.7458 42.4958 ... 30.1875 33.4750 186.0458 84.9750   
11 264.5000 88.2292 90.4792 ... 76.7875 61.5000 245.0208 6.0000   
12 263.3500 97.0792 97.3292 ... 75.6375 68.3500 243.8708 14.8500   
13 5.0000 213.2708 245.0208 ... 188.7125 236.0000 18.4792 260.5000   
14 228.8250 20.5542 20.8042 ... 41.1125 10.1750 209.3458 63.6750   
15 281.3458 65.0750 33.3250 ... 93.6333 42.3458 259.8666 63.1542   
16 204.6417 40.3709 43.3791 ... 16.9292 36.3583 185.1625 62.8583   
17 187.7125 30.5583 60.3083 ... 0.0000 51.2875 174.2333 77.7875   
18 239.0000 28.7292 28.9792 ... 51.2875 0.0000 219.5208 55.5000   
19 21.4792 196.7916 226.5416 ... 174.2333 219.5208 0.0000 246.0208   
20 265.5000 82.2292 84.4792 ... 77.7875 55.5000 246.0208 0.0000   
21 220.7125 7.5583 27.3083 ... 33.0000 32.2875 201.2333 87.7875   
22 237.7167 81.4459 83.6959 ... 54.0042 54.7167 228.2375 27.7833   
23 227.9000 37.6292 37.8792 ... 40.1875 11.1000 208.4208 46.6000   
24 189.8000 28.4708 58.2208 ... 5.9125 51.2000 168.3208 77.7000   
25 255.5000 66.2292 66.4792 ... 67.7875 37.5000 236.0208 18.0000   
26 200.4000 38.1292 49.6208 ... 12.6875 40.6000 184.9208 65.1000   
  
 21 22 23 24 25 26   
0 24.0042 65.0000 21.1833 22.9167 49.7833 21.6833   
1 44.1875 42.8167 3.0000 39.1000 30.6000 26.5000   
2 25.4250 61.5792 18.2375 58.3375 44.3625 47.7375   
3 113.5875 26.5833 72.4000 86.5000 43.8000 69.9000   
4 55.7375 39.2667 48.5500 88.6500 24.0500 76.0500   
5 85.2875 30.2833 44.1000 78.2000 15.5000 67.6000   
6 67.7875 19.2167 26.6000 48.7000 19.0000 38.1000   
7 220.7125 237.7167 227.9000 189.8000 255.5000 200.4000   
8 7.5583 81.4459 37.6292 28.4708 66.2292 38.1292   
9 27.3083 83.6959 37.8792 58.2208 66.4792 49.6208   
10 15.1875 84.1917 38.3750 26.2750 66.9750 40.8750   
11 93.7875 26.7833 52.6000 76.7000 24.0000 64.1000   
12 100.6375 27.6333 59.4500 75.5500 30.8500 64.9500   
13 217.7125 240.7167 224.9000 186.8000 252.5000 197.4000   
14 24.1125 62.8917 17.0750 41.0250 45.6750 30.4250   
15 60.6333 62.3709 53.4458 91.5458 45.1542 82.9458   
16 45.9292 43.0750 25.2583 16.8417 52.8583 6.2417   
17 33.0000 54.0042 40.1875 5.9125 67.7875 12.6875   
18 32.2875 54.7167 11.1000 51.2000 37.5000 40.6000   
19 201.2333 228.2375 208.4208 168.3208 236.0208 184.9208   
20 87.7875 27.7833 46.6000 77.7000 18.0000 65.1000   
21 0.0000 87.0042 41.1875 32.9125 69.7875 43.6875   
22 87.0042 0.0000 45.8167 59.9167 19.7833 43.3167   
23 41.1875 45.8167 0.0000 40.1000 28.6000 29.5000   
24 32.9125 59.9167 40.1000 0.0000 67.7000 16.6000   
25 69.7875 19.7833 28.6000 67.7000 0.0000 57.1000   
26 43.6875 43.3167 29.5000 16.6000 57.1000 0.0000   
  
[27 rows x 27 columns]

### Explanation

Manhattan Distance Calculation: This metric is the sum of the absolute differences between corresponding coordinates of two points. It’s especially useful in grid-like environments where movement is restricted to horizontal and vertical paths.

Handling Different Data Types: Like Euclidean Distance, Manhattan Distance is more meaningful for interval and ratio-scaled data, but it can also be applied to ordinal data with caution. It’s less sensitive to outliers compared to Euclidean Distance.

### Observation and Analysis

The resulting matrices will represent the pairwise Manhattan distances between data points. A smaller value indicates that the data points are closer to each other in terms of their grid-like path, while a larger value indicates they are further apart.

Manhattan Distance is particularly useful in scenarios where the difference in individual dimensions is more meaningful than their squared differences (as in Euclidean Distance).