# Data Mining LAB : Experiment 3

## Submitted By:

Name: Debatreya Das

Roll No. 12212070

CS A4

Data Mining LAB

### Problem Statement

Select two publically available datasets comprising different types of attributes viz. nominal, ordinal, interval-scaled and ratio-scaled. **A dataset must comprise minimally 2 different types of attribute.**

Compute the

* proximity (similarity and/or dissimilarity) between data points using following metrics:
  + Simple Matching Coefficient,
  + Jaccard Coefficient,
  + Cosine Similarity,
  + Euclidean Distance,
  + Manhattan Distance,
  + Supremum Distance, and
  + Correlation as similarity metric.

Initially consider each attribute individually for populating corresponding proximity matrix then consider each data object as represented by a vector of mixed attribute types and compute the proximity matrix for your dataset.

Discuss your observation regarding applicability of different metric and any pattern prevailing in your data.

### Datasets used

* **ADULT**: [Adult Dataset](http://../adult/adult.data) (Links Won't Work)
* **TITANIC**: [Titanic Dataset](http://../titanic/train.csv) (Links Won’t Work)

### Code and Output

* **Jupyter Notebooks**
  + [Simple Matching Coefficient](#_yiynzfe6funi)
  + [Jaccard Coefficient](#_j42ffat3jr3r)
  + [Cosine Similarity](#_pb1kn3yy0tx0)
  + [Euclidean Distance](#_lllkrg66waju)
  + [Manhattan Distance](#_mcpl898f4hhh)
  + [Supremum Distance](#_vnwfcpq8b6tc)
  + [Correlation as similarity metric](#_3b7n1h9sppgh)

### Observation and Discussion

After computing the proximity metrics:

Simple Matching & Jaccard Coefficients are particularly useful for nominal attributes. SMC includes all matches, while Jaccard focuses only on the presence/absence match.

Euclidean & Manhattan Distances are more sensitive to differences in scale, which is why standardization might be necessary before applying them to attributes with different scales.

Cosine Similarity is useful when the magnitude of the data points is irrelevant, focusing purely on the direction of the data points.

Supremum Distance is useful when interested in the maximum difference among dimensions, often highlighting outliers.

Correlation is apt for understanding linear relationships between attributes, especially in interval or ratio-scaled data.

### Conclusion

Each proximity metric has its own strengths and is applicable in different contexts based on the type of attributes and the specific analysis goal. Mixed-type datasets, like those in the experiment, require careful selection of proximity metrics to ensure meaningful comparisons.

## Simple Matching Coefficient

# Simple Matching Coefficient (SMC)

## It is typically used for comparing binary or nominal attributes.

### Importing required Libraries

**import** pandas **as** pd  
**import** numpy **as** np  
**from** sklearn.preprocessing **import** LabelEncoder

### Load Datasets

*# Load datasets*  
adult\_df = pd.read\_csv("../adult/adult\_trim.data", header=None) *# No header*  
titanic\_df = pd.read\_csv('../titanic/train.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   
.. ... ... ...   
886 887 0 2   
887 888 1 1   
888 889 0 3   
889 890 1 1   
890 891 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   
.. ... ... ... ...   
886 Montvila, Rev. Juozas male 27.0 0   
887 Graham, Miss. Margaret Edith female 19.0 0   
888 Johnston, Miss. Catherine Helen "Carrie" female NaN 1   
889 Behr, Mr. Karl Howell male 26.0 0   
890 Dooley, Mr. Patrick male 32.0 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   
.. ... ... ... ... ...   
886 0 211536 13.0000 NaN S   
887 0 112053 30.0000 B42 S   
888 2 W./C. 6607 23.4500 NaN S   
889 0 111369 30.0000 C148 C   
890 0 370376 7.7500 NaN Q   
  
[891 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\_12436\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\_12436\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\_12436\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   
.. ... ... ...   
871 872 1 1   
872 873 0 1   
879 880 1 1   
887 888 1 1   
889 890 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   
.. ... ... ... ...   
871 Beckwith, Mrs. Richard Leonard (Sallie Monypeny) female 47.0 1   
872 Carlsson, Mr. Frans Olof male 33.0 0   
879 Potter, Mrs. Thomas Jr (Lily Alexenia Wilson) female 56.0 0   
887 Graham, Miss. Margaret Edith female 19.0 0   
889 Behr, Mr. Karl Howell male 26.0 0   
  
 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   
.. ... ... ... ... ...   
871 1 11751 52.5542 D35 S   
872 0 695 5.0000 B51 B53 B55 S   
879 1 11767 83.1583 C50 C   
887 0 112053 30.0000 B42 S   
889 0 111369 30.0000 C148 C   
  
[183 rows x 12 columns]

### 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  
.. ... ... ... ... ...  
871 47.0 female 1 52.5542 S  
872 33.0 male 1 5.0000 S  
879 56.0 female 1 83.1583 C  
887 19.0 female 1 30.0000 S  
889 26.0 male 1 30.0000 C  
  
[183 rows x 5 columns]

### 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\_12436\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\_12436\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 2  
6 54.0 1 1 51.8625 2  
10 4.0 0 3 16.7000 2  
11 58.0 0 1 26.5500 2  
.. ... ... ... ... ...  
871 47.0 0 1 52.5542 2  
872 33.0 1 1 5.0000 2  
879 56.0 0 1 83.1583 0  
887 19.0 0 1 30.0000 2  
889 26.0 1 1 30.0000 0  
  
[183 rows x 5 columns]

### 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  
}

### Code to Calculate SMC

**def** simple\_matching\_coefficient(a, b):  
 *"""Calculate the Simple Matching Coefficient between two vectors."""*  
 **try**:  
 **return** np.sum(a == b) / len(a)  
 **except** Exception **as** e:  
 **return** np.nan  
  
*# Function to create the proximity matrix*  
**def** calculate\_smc\_matrix(dataset):  
 n = len(dataset)  
 smc\_matrix = np.zeros((n, n))  
   
 **for** i **in** range(n):  
 *# print(f"{i}/{n}")*  
 **for** j **in** range(n):  
 smc\_matrix[i, j] = simple\_matching\_coefficient(dataset.iloc[i].values, dataset.iloc[j].values)  
   
 **return** pd.DataFrame(smc\_matrix)

### Calculate SMC matrices for each dataset

smc\_matrix\_adult = calculate\_smc\_matrix(adult\_df)  
smc\_matrix\_titanic = calculate\_smc\_matrix(titanic\_df)

### Print SMC matrices

#### *Adult Dataset SMC Matrix*

smc\_matrix\_adult

0 1 2 3 4 5 6 7 8 9 ... 90 91 92 93 \  
0 1.0 0.6 0.2 0.2 0.4 0.0 0.0 0.2 0.0 0.6 ... 0.2 0.0 0.0 0.0   
1 0.6 1.0 0.2 0.2 0.4 0.0 0.0 0.4 0.0 0.6 ... 0.2 0.0 0.0 0.0   
2 0.2 0.2 1.0 0.4 0.2 0.2 0.2 0.6 0.2 0.4 ... 0.4 0.2 0.2 0.6   
3 0.2 0.2 0.4 1.0 0.2 0.2 0.2 0.2 0.2 0.4 ... 0.4 0.2 0.2 0.2   
4 0.4 0.4 0.2 0.2 1.0 0.4 0.4 0.0 0.4 0.6 ... 0.2 0.4 0.6 0.4   
.. ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...   
95 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.2 0.0 0.2 ... 0.2 0.4 0.4 0.0   
96 0.2 0.4 0.2 0.2 0.0 0.0 0.0 0.4 0.0 0.2 ... 0.2 0.0 0.0 0.0   
97 0.2 0.2 0.4 0.4 0.2 0.4 0.2 0.2 0.2 0.4 ... 0.4 0.8 0.6 0.2   
98 0.0 0.0 0.2 0.2 0.4 0.4 0.4 0.0 0.4 0.2 ... 0.2 0.4 0.4 0.4   
99 0.2 0.2 0.6 0.2 0.0 0.0 0.0 0.6 0.0 0.2 ... 0.2 0.0 0.0 0.4   
  
 94 95 96 97 98 99   
0 0.6 0.2 0.2 0.2 0.0 0.2   
1 0.6 0.2 0.4 0.2 0.0 0.2   
2 0.2 0.2 0.2 0.4 0.2 0.6   
3 0.2 0.2 0.2 0.4 0.2 0.2   
4 0.4 0.0 0.0 0.2 0.4 0.0   
.. ... ... ... ... ... ...   
95 0.4 1.0 0.2 0.6 0.0 0.2   
96 0.2 0.2 1.0 0.2 0.2 0.2   
97 0.2 0.6 0.2 1.0 0.2 0.2   
98 0.0 0.0 0.2 0.2 1.0 0.0   
99 0.2 0.2 0.2 0.2 0.0 1.0   
  
[100 rows x 100 columns]

#### *Titanic Dataset SMC Matrix*

smc\_matrix\_titanic

0 1 2 3 4 5 6 7 8 9 ... 173 174 175 \  
0 1.0 0.4 0.2 0.2 0.4 0.0 0.2 0.2 0.6 0.4 ... 0.6 0.4 0.2   
1 0.4 1.0 0.4 0.4 0.6 0.2 0.4 0.4 0.4 0.2 ... 0.4 0.6 0.4   
2 0.2 0.4 1.0 0.2 0.4 0.4 0.6 0.6 0.2 0.4 ... 0.2 0.4 0.6   
3 0.2 0.4 0.2 1.0 0.4 0.2 0.2 0.2 0.2 0.0 ... 0.2 0.4 0.2   
4 0.4 0.6 0.4 0.4 1.0 0.2 0.4 0.4 0.4 0.2 ... 0.4 0.6 0.6   
.. ... ... ... ... ... ... ... ... ... ... ... ... ... ...   
178 0.4 0.6 0.4 0.4 0.6 0.2 0.4 0.4 0.4 0.2 ... 0.4 0.6 0.4   
179 0.2 0.4 0.6 0.2 0.4 0.4 0.6 0.6 0.2 0.4 ... 0.2 0.4 0.6   
180 0.6 0.4 0.2 0.2 0.4 0.0 0.2 0.2 0.6 0.4 ... 0.8 0.4 0.2   
181 0.4 0.6 0.4 0.4 0.6 0.2 0.4 0.6 0.4 0.2 ... 0.4 0.6 0.4   
182 0.4 0.2 0.4 0.0 0.2 0.2 0.4 0.4 0.4 0.6 ... 0.4 0.2 0.4   
  
 176 177 178 179 180 181 182   
0 0.4 0.2 0.4 0.2 0.6 0.4 0.4   
1 0.6 0.4 0.6 0.4 0.4 0.6 0.2   
2 0.4 0.6 0.4 0.6 0.2 0.4 0.4   
3 0.4 0.2 0.4 0.2 0.2 0.4 0.0   
4 0.6 0.4 0.6 0.4 0.4 0.6 0.2   
.. ... ... ... ... ... ... ...   
178 0.6 0.4 1.0 0.4 0.4 0.6 0.2   
179 0.4 0.6 0.4 1.0 0.2 0.4 0.4   
180 0.4 0.2 0.4 0.2 1.0 0.4 0.4   
181 0.6 0.4 0.6 0.4 0.4 1.0 0.4   
182 0.2 0.4 0.2 0.4 0.4 0.4 1.0   
  
[183 rows x 183 columns]

## Jaccard Coefficient

# Jaccad Coefficient (SMC)

## it's specifically useful for binary attributes where the focus is on the presence of attributes (ignoring the absence).

### Importing required Libraries

**import** pandas **as** pd  
**import** numpy **as** np  
**from** sklearn.preprocessing **import** LabelEncoder  
**from** sklearn.metrics **import** jaccard\_score

### 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

### 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\_16092\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\_16092\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  
}

### Computing Jaccard Coefficient

#### *The Jaccard Coefficient is calculated as the size of the intersection divided by the size of the union of the attribute sets. It's typically used for binary or nominal data.*

**def** jaccard\_coefficient(a, b):  
 *"""Calculate the Jaccard Coefficient between two vectors."""*  
 **try**:  
 *# Convert to binary format if necessary (for categorical data)*  
 **return** jaccard\_score(a, b, average='macro')  
 **except** Exception **as** e:  
 **return** np.nan  
  
*# Function to create the Jaccard proximity matrix*  
**def** calculate\_jaccard\_matrix(dataset):  
 n = len(dataset)  
 jaccard\_matrix = np.zeros((n, n))  
   
 **for** i **in** range(n):  
 *# print(f"{i}/{n}")*  
 **for** j **in** range(n):  
 jaccard\_matrix[i, j] = jaccard\_coefficient(dataset.iloc[i].values, dataset.iloc[j].values)  
   
 **return** pd.DataFrame(jaccard\_matrix)

### Calculating Matrices for the data sets

jaccard\_matrix\_adult = calculate\_jaccard\_matrix(adult\_df)  
jaccard\_matrix\_titanic = calculate\_jaccard\_matrix(titanic\_df)

### Displaying the Matrices

#### *Adult Dataset: Jaccard Matrix*

jaccard\_matrix\_adult

0 1 2 3 4 5 6 \  
0 1.000000 0.428571 0.125000 0.071429 0.250000 0.000000 0.000000   
1 0.428571 1.000000 0.125000 0.071429 0.250000 0.000000 0.000000   
2 0.125000 0.125000 1.000000 0.250000 0.125000 0.125000 0.125000   
3 0.071429 0.071429 0.250000 1.000000 0.142857 0.125000 0.125000   
4 0.250000 0.250000 0.125000 0.142857 1.000000 0.250000 0.250000   
.. ... ... ... ... ... ... ...   
95 0.111111 0.111111 0.125000 0.062500 0.000000 0.000000 0.000000   
96 0.111111 0.250000 0.125000 0.062500 0.000000 0.000000 0.000000   
97 0.111111 0.111111 0.285714 0.214286 0.111111 0.285714 0.111111   
98 0.000000 0.000000 0.125000 0.125000 0.250000 0.250000 0.285714   
99 0.071429 0.071429 0.300000 0.055556 0.000000 0.000000 0.000000   
  
 7 8 9 ... 90 91 92 93 \  
0 0.125000 0.000 0.428571 ... 0.125000 0.000000 0.000000 0.000000   
1 0.285714 0.000 0.428571 ... 0.111111 0.000000 0.000000 0.000000   
2 0.333333 0.125 0.285714 ... 0.285714 0.125000 0.125000 0.333333   
3 0.071429 0.125 0.250000 ... 0.214286 0.125000 0.125000 0.142857   
4 0.000000 0.250 0.428571 ... 0.111111 0.250000 0.428571 0.285714   
.. ... ... ... ... ... ... ... ...   
95 0.125000 0.000 0.111111 ... 0.111111 0.250000 0.250000 0.000000   
96 0.285714 0.000 0.111111 ... 0.111111 0.000000 0.000000 0.000000   
97 0.125000 0.125 0.250000 ... 0.250000 0.666667 0.428571 0.125000   
98 0.000000 0.250 0.111111 ... 0.111111 0.285714 0.285714 0.285714   
99 0.300000 0.000 0.071429 ... 0.071429 0.000000 0.000000 0.166667   
  
 94 95 96 97 98 99   
0 0.428571 0.111111 0.111111 0.111111 0.000 0.071429   
1 0.428571 0.111111 0.250000 0.111111 0.000 0.071429   
2 0.125000 0.125000 0.125000 0.285714 0.125 0.300000   
3 0.071429 0.062500 0.062500 0.214286 0.125 0.055556   
4 0.250000 0.000000 0.000000 0.111111 0.250 0.000000   
.. ... ... ... ... ... ...   
95 0.250000 1.000000 0.111111 0.428571 0.000 0.071429   
96 0.111111 0.111111 1.000000 0.111111 0.125 0.071429   
97 0.111111 0.428571 0.111111 1.000000 0.125 0.071429   
98 0.000000 0.000000 0.125000 0.125000 1.000 0.000000   
99 0.071429 0.071429 0.071429 0.071429 0.000 1.000000   
  
[100 rows x 100 columns]

#### *Titanic Dataset: Jaccard Matrix*

jaccard\_matrix\_titanic

0 1 2 3 4 5 6 7 8 9 ... 17 18 19 \  
0 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
1 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
2 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
3 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
4 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
5 NaN NaN NaN NaN NaN 1.000000 NaN 0.111111 NaN NaN ... NaN 0.111111 NaN   
6 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
7 NaN NaN NaN NaN NaN 0.111111 NaN 1.000000 NaN NaN ... NaN 0.200000 NaN   
8 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
9 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
10 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
11 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
12 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
13 NaN NaN NaN NaN NaN 0.047619 NaN 0.333333 NaN NaN ... NaN 0.111111 NaN   
14 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
15 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
16 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
17 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
18 NaN NaN NaN NaN NaN 0.111111 NaN 0.200000 NaN NaN ... NaN 1.000000 NaN   
19 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
20 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
21 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
22 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
23 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
24 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
25 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
26 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN   
  
 20 21 22 23 24 25 26   
0 NaN NaN NaN NaN NaN NaN NaN   
1 NaN NaN NaN NaN NaN NaN NaN   
2 NaN NaN NaN NaN NaN NaN NaN   
3 NaN NaN NaN NaN NaN NaN NaN   
4 NaN NaN NaN NaN NaN NaN NaN   
5 NaN NaN NaN NaN NaN NaN NaN   
6 NaN NaN NaN NaN NaN NaN NaN   
7 NaN NaN NaN NaN NaN NaN NaN   
8 NaN NaN NaN NaN NaN NaN NaN   
9 NaN NaN NaN NaN NaN NaN NaN   
10 NaN NaN NaN NaN NaN NaN NaN   
11 NaN NaN NaN NaN NaN NaN NaN   
12 NaN NaN NaN NaN NaN NaN NaN   
13 NaN NaN NaN NaN NaN NaN NaN   
14 NaN NaN NaN NaN NaN NaN NaN   
15 NaN NaN NaN NaN NaN NaN NaN   
16 NaN NaN NaN NaN NaN NaN NaN   
17 NaN NaN NaN NaN NaN NaN NaN   
18 NaN NaN NaN NaN NaN NaN NaN   
19 NaN NaN NaN NaN NaN NaN NaN   
20 NaN NaN NaN NaN NaN NaN NaN   
21 NaN NaN NaN NaN NaN NaN NaN   
22 NaN NaN NaN NaN NaN NaN NaN   
23 NaN NaN NaN NaN NaN NaN NaN   
24 NaN NaN NaN NaN NaN NaN NaN   
25 NaN NaN NaN NaN NaN NaN NaN   
26 NaN NaN NaN NaN NaN NaN NaN   
  
[27 rows x 27 columns]

### Explanation

Jaccard Coefficient Calculation: This metric focuses on the presence of attributes, making it useful for binary data. It calculates the ratio of the intersection of attributes to the union of attributes between two data points.

Handling Nominal Data: The Jaccard Coefficient is generally used for binary data, but in this case, we can apply it to the encoded nominal attributes as they are binary representations of categories.

### Observation and Analysis

The resulting matrices will have values between 0 and 1, where 1 indicates identical presence/absence of attributes (perfect similarity) and 0 indicates no similarity.

Jaccard Coefficient is particularly sensitive to the presence of attributes (1s) and ignores cases where both attributes are absent (0s), making it a useful measure for sparse datasets.

## Cosine Similarity

# Cosine Similarity

This metric measures the cosine of the angle between two non-zero vectors in a multi-dimensional space. It is particularly useful for interval and ratio-scaled attributes, where the magnitude of the vectors is not as important as their direction.

### Importing required Libraries

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

### 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\_15644\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\_15644\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\_15644\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\_15644\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\_15644\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 Cosine Similarity

**def** cosine\_similarity(a, b):  
 *"""Calculate the Cosine Similarity between two vectors."""*  
 **try**:  
 **return** 1 - cosine(a, b) *# scipy returns distance, so we subtract from 1 to get similarity*  
 **except** Exception **as** e:  
 **return** np.nan  
  
*# Function to create the Cosine Similarity matrix*  
**def** calculate\_cosine\_similarity\_matrix(dataset):  
 n = len(dataset)  
 cosine\_matrix = np.zeros((n, n))  
   
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 cosine\_matrix[i, j] = cosine\_similarity(dataset.iloc[i].values, dataset.iloc[j].values)  
   
 **return** pd.DataFrame(cosine\_matrix)

### Calculate Cosine SImilarity

#### *For Adult Dataset*

cosine\_matrix\_adult = calculate\_cosine\_similarity\_matrix(adult\_df)  
cosine\_matrix\_adult

0 1 2 3 4 5 6 \  
0 1.000000 0.996120 0.992260 0.967533 0.991062 0.993478 0.969143   
1 0.996120 1.000000 0.995356 0.985101 0.980434 0.987759 0.986363   
2 0.992260 0.995356 1.000000 0.973431 0.980291 0.991643 0.980816   
3 0.967533 0.985101 0.973431 1.000000 0.934177 0.948206 0.997445   
4 0.991062 0.980434 0.980291 0.934177 1.000000 0.996981 0.936376   
.. ... ... ... ... ... ... ...   
95 0.976034 0.968522 0.984786 0.919479 0.983085 0.991265 0.933007   
96 0.998883 0.997556 0.993527 0.971956 0.991562 0.994680 0.972707   
97 0.987173 0.985626 0.996606 0.951403 0.982858 0.993749 0.962654   
98 0.991174 0.998577 0.991648 0.990072 0.974700 0.982658 0.989681   
99 0.987921 0.989193 0.997422 0.959821 0.984550 0.994882 0.968122   
  
 7 8 9 ... 90 91 92 93 \  
0 0.987947 0.986347 0.996862 ... 0.985347 0.986888 0.975612 0.991905   
1 0.996085 0.974868 0.998307 ... 0.996532 0.985455 0.965694 0.988742   
2 0.996382 0.980890 0.995442 ... 0.991265 0.996296 0.981828 0.996523   
3 0.987336 0.923486 0.976317 ... 0.994705 0.951232 0.913116 0.953271   
4 0.965545 0.997933 0.988638 ... 0.963205 0.983161 0.983478 0.988883   
.. ... ... ... ... ... ... ... ...   
95 0.967070 0.991735 0.976569 ... 0.954591 0.994996 0.998900 0.994023   
96 0.989034 0.986548 0.999319 ... 0.989125 0.987346 0.974227 0.991879   
97 0.986846 0.987841 0.988844 ... 0.977234 0.999692 0.993583 0.998922   
98 0.994338 0.967490 0.996608 ... 0.998355 0.979346 0.955547 0.982578   
99 0.988367 0.987459 0.993277 ... 0.983371 0.997737 0.988346 0.997355   
  
 94 95 96 97 98 99   
0 0.994913 0.976034 0.998883 0.987173 0.991174 0.987921   
1 0.991649 0.968522 0.997556 0.985626 0.998577 0.989193   
2 0.990753 0.984786 0.993527 0.996606 0.991648 0.997422   
3 0.958745 0.919479 0.971956 0.951403 0.990072 0.959821   
4 0.995978 0.983085 0.991562 0.982858 0.974700 0.984550   
.. ... ... ... ... ... ...   
95 0.982860 1.000000 0.976136 0.995441 0.959385 0.992139   
96 0.997575 0.976136 1.000000 0.987524 0.994687 0.990804   
97 0.988614 0.995441 0.987524 1.000000 0.979044 0.998150   
98 0.988623 0.959385 0.994687 0.979044 1.000000 0.985352   
99 0.993014 0.992139 0.990804 0.998150 0.985352 1.000000   
  
[100 rows x 100 columns]

#### *For TItanic Dataset*

cosine\_matrix\_titanic = calculate\_cosine\_similarity\_matrix(titanic\_df)  
cosine\_matrix\_titanic

0 1 2 3 4 5 6 \  
0 1.000000 0.995550 0.950379 0.953797 0.794964 0.753466 0.983664   
1 0.995550 1.000000 0.975197 0.927993 0.847999 0.811493 0.996103   
2 0.950379 0.975197 1.000000 0.830554 0.944041 0.920198 0.990421   
3 0.953797 0.927993 0.830554 1.000000 0.609816 0.565279 0.897217   
4 0.794964 0.847999 0.944041 0.609816 1.000000 0.996740 0.889719   
5 0.753466 0.811493 0.920198 0.565279 0.996740 1.000000 0.858714   
6 0.983664 0.996103 0.990421 0.897217 0.889719 0.858714 1.000000   
7 0.914004 0.872311 0.742794 0.971280 0.480649 0.422964 0.827375   
8 0.996915 0.999761 0.971814 0.931194 0.840094 0.802464 0.994507   
9 0.949365 0.974291 0.999899 0.827611 0.945111 0.921172 0.989670   
10 0.999876 0.996019 0.951987 0.952835 0.797846 0.756918 0.984815   
11 0.741537 0.800941 0.912633 0.552546 0.995315 0.999406 0.848895   
12 0.703764 0.766632 0.887558 0.516672 0.985513 0.995827 0.818987   
13 0.920035 0.879602 0.752778 0.973587 0.493846 0.436473 0.835675   
14 0.987848 0.997989 0.987099 0.904854 0.879097 0.846107 0.999611   
15 0.809779 0.860649 0.951917 0.627451 0.999486 0.994687 0.900684   
16 0.989890 0.972398 0.897178 0.979511 0.701412 0.654025 0.949109   
17 0.974767 0.949966 0.857635 0.985645 0.640159 0.589457 0.920013   
18 0.969902 0.988348 0.997505 0.866148 0.918423 0.890551 0.997659   
19 0.923689 0.883987 0.758953 0.974731 0.501963 0.444933 0.840815   
20 0.763595 0.820538 0.925919 0.578173 0.997874 0.999505 0.866089   
21 0.992686 0.999551 0.980901 0.917484 0.861941 0.826884 0.998122   
22 0.990109 0.998842 0.983530 0.915496 0.869931 0.836496 0.998803   
23 0.992710 0.999548 0.980661 0.918877 0.861284 0.826546 0.998180   
24 0.980876 0.958475 0.872046 0.983677 0.661958 0.612193 0.930736   
25 0.894276 0.932092 0.988775 0.749304 0.981410 0.968018 0.959776   
26 0.985364 0.965439 0.883920 0.983066 0.680559 0.631707 0.939538   
  
 7 8 9 ... 17 18 19 20 \  
0 0.914004 0.996915 0.949365 ... 0.974767 0.969902 0.923689 0.763595   
1 0.872311 0.999761 0.974291 ... 0.949966 0.988348 0.883987 0.820538   
2 0.742794 0.971814 0.999899 ... 0.857635 0.997505 0.758953 0.925919   
3 0.971280 0.931194 0.827611 ... 0.985645 0.866148 0.974731 0.578173   
4 0.480649 0.840094 0.945111 ... 0.640159 0.918423 0.501963 0.997874   
5 0.422964 0.802464 0.921172 ... 0.589457 0.890551 0.444933 0.999505   
6 0.827375 0.994507 0.989670 ... 0.920013 0.997659 0.840815 0.866089   
7 1.000000 0.879364 0.740421 ... 0.981288 0.788166 0.999692 0.436753   
8 0.879364 1.000000 0.971077 ... 0.954297 0.985949 0.890788 0.811660   
9 0.740421 0.971077 1.000000 ... 0.855681 0.997135 0.756698 0.926885   
10 0.912095 0.997160 0.950829 ... 0.973872 0.971209 0.921846 0.766711   
11 0.406600 0.791583 0.913638 ... 0.574645 0.881853 0.428725 0.999417   
12 0.358592 0.756428 0.888446 ... 0.531106 0.853892 0.381081 0.994150   
13 0.999879 0.886464 0.750455 ... 0.984037 0.797324 0.999940 0.450268   
14 0.840397 0.996910 0.986426 ... 0.928908 0.995935 0.853398 0.854003   
15 0.502247 0.853312 0.953092 ... 0.658975 0.927929 0.523320 0.995829   
16 0.962005 0.975764 0.895647 ... 0.996483 0.926090 0.968415 0.665266   
17 0.981288 0.954297 0.855681 ... 1.000000 0.891790 0.985667 0.601552   
18 0.788166 0.985949 0.997135 ... 0.891790 1.000000 0.802980 0.897225   
19 0.999692 0.890788 0.756698 ... 0.985667 0.802980 1.000000 0.458551   
20 0.436753 0.811660 0.926885 ... 0.601552 0.897225 0.458551 1.000000   
21 0.858832 0.999026 0.980147 ... 0.941250 0.992170 0.871109 0.835285   
22 0.849936 0.997672 0.982543 ... 0.935341 0.993696 0.862468 0.845044   
23 0.859375 0.998942 0.979817 ... 0.941682 0.992030 0.871606 0.834844   
24 0.975362 0.962606 0.870352 ... 0.999504 0.904389 0.980484 0.624022   
25 0.636737 0.926405 0.988988 ... 0.773225 0.976086 0.655360 0.971460   
26 0.969558 0.968991 0.882138 ... 0.998476 0.914699 0.975217 0.643654   
  
 21 22 23 24 25 26   
0 0.992686 0.990109 0.992710 0.980876 0.894276 0.985364   
1 0.999551 0.998842 0.999548 0.958475 0.932092 0.965439   
2 0.980901 0.983530 0.980661 0.872046 0.988775 0.883920   
3 0.917484 0.915496 0.918877 0.983677 0.749304 0.983066   
4 0.861941 0.869931 0.861284 0.661958 0.981410 0.680559   
5 0.826884 0.836496 0.826546 0.612193 0.968018 0.631707   
6 0.998122 0.998803 0.998180 0.930736 0.959776 0.939538   
7 0.858832 0.849936 0.859375 0.975362 0.636737 0.969558   
8 0.999026 0.997672 0.998942 0.962606 0.926405 0.968991   
9 0.980147 0.982543 0.979817 0.870352 0.988988 0.882138   
10 0.993348 0.990909 0.993459 0.979977 0.896648 0.984597   
11 0.816352 0.826794 0.815936 0.597622 0.963001 0.617824   
12 0.783153 0.794927 0.783146 0.554669 0.946423 0.575226   
13 0.866438 0.857785 0.866945 0.978540 0.648211 0.973147   
14 0.999382 0.999423 0.999338 0.939137 0.952714 0.947365   
15 0.874245 0.881368 0.873584 0.680496 0.985656 0.698297   
16 0.965910 0.961020 0.966199 0.998536 0.822869 0.999363   
17 0.941250 0.935341 0.941682 0.999504 0.773225 0.998476   
18 0.992170 0.993696 0.992030 0.904389 0.976086 0.914699   
19 0.871109 0.862468 0.871606 0.980484 0.655360 0.975217   
20 0.835285 0.845044 0.834844 0.624022 0.971460 0.643654   
21 1.000000 0.999406 0.999964 0.950564 0.941469 0.957961   
22 0.999406 1.000000 0.999494 0.944830 0.947106 0.953060   
23 0.999964 0.999494 1.000000 0.950895 0.941296 0.958281   
24 0.950564 0.944830 0.950895 1.000000 0.790968 0.999515   
25 0.941469 0.947106 0.941296 0.790968 1.000000 0.805963   
26 0.957961 0.953060 0.958281 0.999515 0.805963 1.000000   
  
[27 rows x 27 columns]

### Explanation

Cosine Similarity Calculation: This metric focuses on the orientation of the data points in a multi-dimensional space. A value of 1 means that the vectors are identical, while 0 indicates orthogonal vectors (no similarity), and -1 indicates opposite vectors.

Handling Different Data Types: Cosine similarity works best with interval and ratio-scaled data, as it considers the direction (or relative distribution) rather than the magnitude of the data points.

### Observation and Analysis

The resulting matrices will show the similarity between data points based on the angle between their corresponding vectors. This is especially useful in text mining and other applications where the magnitude of features is less important than their relative importance.

Cosine Similarity is not affected by the scale of the data, making it useful when the attributes have been normalized or when dealing with sparse data.

## Euclidean Distance

# 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.

## Manhattan Distance

# 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).

## Supremum Distance

# 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.

## Correlation Matrix

# Correlation Matrix

Correlation measures the linear relationship between two variables and is particularly useful when dealing with interval and ratio-scaled data. It can indicate how similar the variations in two data points are.

### Importing required Libraries

**import** pandas **as** pd  
**import** numpy **as** np  
**from** sklearn.preprocessing **import** LabelEncoder

### 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\_8904\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\_8904\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\_8904\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\_8904\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\_8904\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 Correlation Matrix

**def** correlation\_coefficient(a, b):  
 *"""Calculate the Pearson Correlation Coefficient between two vectors."""*  
 **try**:  
 **return** np.corrcoef(a, b)[0, 1]  
 **except** Exception **as** e:  
 **return** np.nan  
  
*# Function to create the Correlation matrix*  
**def** calculate\_correlation\_matrix(dataset):  
 n = len(dataset)  
 correlation\_matrix = np.zeros((n, n))  
   
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 correlation\_matrix[i, j] = correlation\_coefficient(dataset.iloc[i].values, dataset.iloc[j].values)  
   
 **return** pd.DataFrame(correlation\_matrix)

### Calculate Correlation Matrix

#### *For Adult Dataset*

correlation\_matrix\_adult = calculate\_correlation\_matrix(adult\_df)  
correlation\_matrix\_adult

0 1 2 3 4 5 6 \  
0 1.000000 0.997350 0.987351 0.979855 0.982608 0.987214 0.979155   
1 0.997350 1.000000 0.992346 0.990383 0.971402 0.981827 0.990318   
2 0.987351 0.992346 1.000000 0.977230 0.966611 0.986108 0.986540   
3 0.979855 0.990383 0.977230 1.000000 0.931087 0.947285 0.996425   
4 0.982608 0.971402 0.966611 0.931087 1.000000 0.994572 0.930666   
.. ... ... ... ... ... ... ...   
95 0.952796 0.951934 0.977942 0.912323 0.964839 0.984216 0.931341   
96 0.998811 0.996954 0.988350 0.978175 0.986529 0.990747 0.976581   
97 0.974860 0.977875 0.995414 0.952534 0.966230 0.987763 0.967844   
98 0.996972 0.999669 0.990512 0.990242 0.972744 0.981685 0.988416   
99 0.978260 0.981818 0.995396 0.957604 0.973384 0.991406 0.968203   
  
 7 8 9 ... 90 91 92 93 \  
0 0.985819 0.973770 0.996797 ... 0.990631 0.974937 0.952244 0.984207   
1 0.993768 0.963523 0.997273 ... 0.997941 0.976220 0.947628 0.983245   
2 0.995618 0.969761 0.991901 ... 0.992350 0.993895 0.973154 0.995007   
3 0.989915 0.918105 0.979980 ... 0.995363 0.948173 0.903421 0.954681   
4 0.948588 0.995967 0.983983 ... 0.957142 0.968306 0.965757 0.978583   
.. ... ... ... ... ... ... ... ...   
95 0.955238 0.982308 0.963549 ... 0.946838 0.993654 0.997665 0.990267   
96 0.983955 0.978473 0.999202 ... 0.990899 0.976483 0.955091 0.985272   
97 0.983614 0.976983 0.981705 ... 0.976156 0.999626 0.989609 0.997909   
98 0.991396 0.963678 0.997955 ... 0.997641 0.973235 0.944175 0.980862   
99 0.982442 0.980936 0.988149 ... 0.980528 0.996103 0.984183 0.995884   
  
 94 95 96 97 98 99   
0 0.990559 0.952796 0.998811 0.974860 0.996972 0.978260   
1 0.987215 0.951934 0.996954 0.977875 0.999669 0.981818   
2 0.983535 0.977942 0.988350 0.995414 0.990512 0.995396   
3 0.959776 0.912323 0.978175 0.952534 0.990242 0.957604   
4 0.993915 0.964839 0.986529 0.966230 0.972744 0.973384   
.. ... ... ... ... ... ...   
95 0.969546 1.000000 0.958159 0.992814 0.949612 0.990722   
96 0.995602 0.958159 1.000000 0.977120 0.997514 0.983073   
97 0.978402 0.992814 0.977120 1.000000 0.975251 0.997584   
98 0.988923 0.949612 0.997514 0.975251 1.000000 0.980972   
99 0.987223 0.990722 0.983073 0.997584 0.980972 1.000000   
  
[100 rows x 100 columns]

#### *For Titanic Dataset*

correlation\_matrix\_titanic = calculate\_correlation\_matrix(titanic\_df)  
correlation\_matrix\_titanic

0 1 2 3 4 5 6 \  
0 1.000000 0.993438 0.919708 0.925325 0.674371 0.601460 0.975698   
1 0.993438 1.000000 0.958218 0.879845 0.753852 0.687969 0.994154   
2 0.919708 0.958218 1.000000 0.712509 0.909944 0.866324 0.983353   
3 0.925325 0.879845 0.712509 1.000000 0.367142 0.280890 0.824795   
4 0.674371 0.753852 0.909944 0.367142 1.000000 0.995190 0.819495   
5 0.601460 0.687969 0.866324 0.280890 0.995190 1.000000 0.761294   
6 0.975698 0.994154 0.983353 0.824795 0.819495 0.761294 1.000000   
7 0.892647 0.835749 0.644276 0.981587 0.269311 0.176781 0.772517   
8 0.995210 0.999768 0.953679 0.886621 0.743310 0.676623 0.992425   
9 0.917958 0.956767 0.999920 0.709759 0.911627 0.868651 0.982463   
10 0.999916 0.993691 0.920753 0.922606 0.676151 0.603121 0.976317   
11 0.586313 0.674354 0.855988 0.267991 0.992610 0.999181 0.748391   
12 0.514880 0.608010 0.807828 0.188809 0.977701 0.993312 0.688286   
13 0.899952 0.844715 0.656760 0.983598 0.285164 0.192984 0.782849   
14 0.981440 0.996751 0.977880 0.839658 0.803135 0.742859 0.999608   
15 0.697116 0.773668 0.922489 0.394079 0.999193 0.991746 0.837030   
16 0.984967 0.958818 0.838373 0.970090 0.536940 0.454802 0.923426   
17 0.963749 0.927261 0.781920 0.983040 0.453165 0.366543 0.882174   
18 0.952177 0.980598 0.995678 0.772886 0.867522 0.816266 0.995979   
19 0.904026 0.849637 0.663848 0.984075 0.294109 0.202236 0.788686   
20 0.620265 0.705193 0.877352 0.307199 0.996774 0.999254 0.776057   
21 0.988953 0.999264 0.967654 0.861830 0.775961 0.712499 0.997386   
22 0.985852 0.998496 0.971436 0.856703 0.787108 0.724881 0.997961   
23 0.989423 0.999370 0.966838 0.863373 0.773931 0.710243 0.997149   
24 0.972736 0.939975 0.803826 0.979446 0.484868 0.399972 0.898512   
25 0.825618 0.884093 0.980519 0.569122 0.973190 0.947340 0.928780   
26 0.978661 0.949193 0.819682 0.977002 0.508782 0.424763 0.910088   
  
 7 8 9 ... 17 18 19 20 \  
0 0.892647 0.995210 0.917958 ... 0.963749 0.952177 0.904026 0.620265   
1 0.835749 0.999768 0.956767 ... 0.927261 0.980598 0.849637 0.705193   
2 0.644276 0.953679 0.999920 ... 0.781920 0.995678 0.663848 0.877352   
3 0.981587 0.886621 0.709759 ... 0.983040 0.772886 0.984075 0.307199   
4 0.269311 0.743310 0.911627 ... 0.453165 0.867522 0.294109 0.996774   
5 0.176781 0.676623 0.868651 ... 0.366543 0.816266 0.202236 0.999254   
6 0.772517 0.992425 0.982463 ... 0.882174 0.995979 0.788686 0.776057   
7 1.000000 0.844335 0.640809 ... 0.980528 0.712522 0.999658 0.200376   
8 0.844335 1.000000 0.952328 ... 0.933093 0.977476 0.857932 0.693802   
9 0.640809 0.952328 1.000000 ... 0.779074 0.995183 0.660500 0.879412   
10 0.891599 0.995384 0.918937 ... 0.963164 0.953009 0.903014 0.621688   
11 0.158591 0.662480 0.858213 ... 0.349063 0.804574 0.184045 0.999078   
12 0.073910 0.595811 0.810849 ... 0.267674 0.750107 0.099762 0.990900   
13 0.999859 0.853044 0.653319 ... 0.983618 0.723959 0.999935 0.216593   
14 0.789992 0.995461 0.976868 ... 0.895013 0.993080 0.805590 0.758153   
15 0.299276 0.763857 0.924292 ... 0.480811 0.882669 0.323915 0.993413   
16 0.956873 0.963423 0.836050 ... 0.995215 0.885357 0.964110 0.475582   
17 0.980528 0.933093 0.779074 ... 1.000000 0.836438 0.985275 0.388607   
18 0.712522 0.977476 0.995183 ... 0.836438 1.000000 0.730437 0.829249   
19 0.999658 0.857932 0.660500 ... 0.985275 0.730437 1.000000 0.225628   
20 0.200376 0.693802 0.879412 ... 0.388607 0.829249 0.225628 1.000000   
21 0.816380 0.998669 0.966449 ... 0.913895 0.986902 0.831046 0.728610   
22 0.805580 0.997263 0.970112 ... 0.906064 0.989061 0.820552 0.741658   
23 0.818231 0.998825 0.965620 ... 0.915195 0.986379 0.832829 0.726412   
24 0.972742 0.945445 0.801248 ... 0.999278 0.855589 0.978447 0.421488   
25 0.482654 0.876786 0.981455 ... 0.645182 0.958178 0.505192 0.954333   
26 0.966053 0.953936 0.816975 ... 0.997898 0.869324 0.972372 0.446578   
  
 21 22 23 24 25 26   
0 0.988953 0.985852 0.989423 0.972736 0.825618 0.978661   
1 0.999264 0.998496 0.999370 0.939975 0.884093 0.949193   
2 0.967654 0.971436 0.966838 0.803826 0.980519 0.819682   
3 0.861830 0.856703 0.863373 0.979446 0.569122 0.977002   
4 0.775961 0.787108 0.773931 0.484868 0.973190 0.508782   
5 0.712499 0.724881 0.710243 0.399972 0.947340 0.424763   
6 0.997386 0.997961 0.997149 0.898512 0.928780 0.910088   
7 0.816380 0.805580 0.818231 0.972742 0.482654 0.966053   
8 0.998669 0.997263 0.998825 0.945445 0.876786 0.953936   
9 0.966449 0.970112 0.965620 0.801248 0.981455 0.816975   
10 0.989397 0.986127 0.989859 0.972162 0.826771 0.978138   
11 0.698693 0.712487 0.696399 0.382525 0.940667 0.408168   
12 0.634393 0.648889 0.631920 0.302573 0.907857 0.328379   
13 0.825760 0.815311 0.827568 0.976391 0.497025 0.970209   
14 0.999018 0.999097 0.998871 0.910445 0.918061 0.921328   
15 0.795318 0.805268 0.793367 0.512222 0.979875 0.535128   
16 0.948722 0.942214 0.949731 0.998156 0.716015 0.999120   
17 0.913895 0.906064 0.915195 0.999278 0.645182 0.997898   
18 0.986902 0.989061 0.986379 0.855589 0.958178 0.869324   
19 0.831046 0.820552 0.832829 0.978447 0.505192 0.972372   
20 0.728610 0.741658 0.726412 0.421488 0.954333 0.446578   
21 1.000000 0.999294 0.999995 0.927869 0.899629 0.937636   
22 0.999294 1.000000 0.999232 0.920417 0.907114 0.931298   
23 0.999995 0.999232 1.000000 0.929061 0.898224 0.938747   
24 0.927869 0.920417 0.929061 1.000000 0.672388 0.999371   
25 0.899629 0.907114 0.898224 0.672388 1.000000 0.692271   
26 0.937636 0.931298 0.938747 0.999371 0.692271 1.000000   
  
[27 rows x 27 columns]

### Explanation

Correlation Calculation: This metric measures the degree of linear relationship between two variables. A high positive value indicates that as one variable increases, the other also tends to increase, while a high negative value indicates an inverse relationship.

Handling Different Data Types: Correlation is best suited for interval and ratio-scaled data. It may not be meaningful for nominal or ordinal data unless they are encoded or transformed appropriately.

### Observation and Analysis

The resulting matrices will show the correlation coefficients between each pair of data points. A value close to 1 or −1 indicates a strong relationship, while a value close to 0 indicates little to no linear relationship.

Correlation as a Similarity Metric is useful for understanding the degree to which data points share a linear relationship, which can be especially important in fields like finance and economics.