# 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]