Data Description

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

class: <=50K, >50K

In [10]:
```
import pandas as pd
import numpy as np
```

Importing the dataframe

In [11]:
```
df_adult = pd.read_csv("/Users/haileythanki/Downloads/adult.csv", index_col=Fals
df_adult
```

Out[11]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White

32561 rows × 15 columns

Displaying summary of the different columns of the dataframe

In [12]:
```
df_adult.describe(include="all")
```

Out[12]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation
count	32561.000000	30725	3.256100e+04	32561	32561.000000	32561	30718
unique	NaN	8	NaN	16	NaN	7	14
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty
freq	NaN	22696	NaN	10501	NaN	14976	4140
mean	38.581647	NaN	1.897784e+05	NaN	10.080679	NaN	NaN

	age	workclass	fnlwgt	education	education-num	marital-status	occupation
std	13.640433	NaN	1.055500e+05	NaN	2.572720	NaN	NaN
min	17.000000	NaN	1.228500e+04	NaN	1.000000	NaN	NaN
25%	28.000000	NaN	1.178270e+05	NaN	9.000000	NaN	NaN
50%	37.000000	NaN	1.783560e+05	NaN	10.000000	NaN	NaN
75%	48.000000	NaN	2.370510e+05	NaN	12.000000	NaN	NaN
max	90.000000	NaN	1.484705e+06	NaN	16.000000	NaN	NaN

Displaying unique values for all non-numeric columns in the adult dataframe

In [13]:
```
non_num_cols = df_adult.select_dtypes(include=['object']).columns.tolist()
for col in non_num_cols:
    print (col, ":", df_adult[col].unique())
    print ("\n")
```

```
workclass : ['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
nan
 'Self-emp-inc' 'Without-pay' 'Never-worked']


education : ['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-
acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']


marital-status : ['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spous
e-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']


occupation : ['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-special
ty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' nan
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']


relationship : ['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-
relative']


race : ['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']


sex : ['Male' 'Female']


native-country : ['United-States' 'Cuba' 'Jamaica' 'India' nan 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']
```

```
class : ['<=50K' '>50K']
```

In [14]:
```
df_adult.isna().sum()
```

Out[14]:
```
age                 0
workclass        1836
fnlwgt              0
education           0
education-num       0
marital-status      0
occupation       1843
relationship        0
race                0
sex                 0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    583
class               0
dtype: int64
```

Question 1:

For each attribute design and program a similarity metric. This is a function that takes in two values attribute values and returns a result in the interval [0,1]. "0" means ' not similar at all, "1" means 'similar'.

Nominal attributes in the adult dataframe

Nominal means “relating to names.” The values of a nominal attribute are symbols or names of things. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as categorical. The values do not have any meaningful order.

The adult dataframe has the following nominal attributes:

- workclass
- marital-status
- occupation
- relationship
- race
- native-country

Similarity metric for nominal and binary attributes:

$s_{ij}^{(f)} = 1$ if $x_{if} = x_{jf}$; otherwise, $s_{ij}^{(f)} = 0$.

The function for calculating the similarity metric of nominal attributes is as follows:

In [20]:
```
def func_sim_nominal(attr):
```

```
    print("Enter", attr, "of person 1: ")
    val1 = input()
    print("Enter", attr, "of person 2: ")
    val2 = input()
    if(val1==val2):
        sim = 1
    else:
        sim = 0
    return (sim)
```

Binary attributes in the adult dataframe

A binary attribute is a nominal attribute with only two categories or states: 0 or 1.

The adult dataframe has the following binary attributes:

- sex
- class

Similarity metric for nominal and binary attributes:

$s_{ij}^{(f)} = 1$ if $x_{if} = x_{jf}$; otherwise, $s_{ij}^{(f)} = 0$.

The function for calculating the similarity metric of binary attributes is as follows:

In [19]:
```
def func_sim_binary(attr):
    print("Enter", attr, "of person 1: ")
    val1 = input()
    print("Enter", attr, "of person 2: ")
    val2 = input()
    if(val1==val2):
        sim = 1
    else:
        sim = 0
    return (sim)
```

Ordinal attributes in the adult dataframe

An ordinal attribute is an attribute with possible values that have a meaningful order or ranking among them, but the magnitude between successive values is not known.

The adult dataframe has the following ordinal attributes:

- education

an ordered factor with levels:

Preschool < 1st-4th < 5th-6th < 7th-8th < 9th < 10th < 11th < 12th < HS-grad < Prof-school < Assoc-acdm < Assoc-voc < Some-college < Bachelors < Masters < Doctorate.

levels = {'Preschool':1, '1st-4th':2, '5th-6th':3, '7th-8th':4, '9th':5, '10th':6, '11th':7, '12th':8, 'HS-grad':9, 'Prof-school':10, 'Assoc-acdm':11, 'Assoc-voc':12, 'Some-college':13, 'Bachelors':14,

'Masters':15, 'Doctorate':16}

Similarity metric for ordinal attributes:

Compute the ranks r_{if} and $z_{if} = \frac{r_{if}-1}{M_f-1}$, and treat z_{if} as numeric.

The function for calculating the similarity metric of ordinal attributes is as follows:

In [17]:
```
def func_sim_ordinal(attr):
    print("Enter education level of person 1: ")
    edlevel1 = (input())
    print("Enter education level of person 2: ")
    edlevel2 = (input())
    df_adult = pd.read_csv("/Users/haileythanki/Downloads/adult.csv", index_col=
    education_replacements = {'Preschool':1, '1st-4th':2, '5th-6th':3, '7th-8th'
    'Assoc-voc':12, 'Some-college':13, 'Bachelors':14,  'Masters':15, 'Doctorate
    df_adult = df_adult.replace(education_replacements)
    max_h_x_hf = max(df_adult['education'])
    min_h_x_hf = min(df_adult['education'])
    sim = 1 - ((abs(education_replacements.get(edlevel1)-education_replacements.
    return (sim)
```

Numeric attributes in the adult dataframe

A numeric attribute is quantitative; that is, it is a measurable quantity, represented in integer or real values.

The adult dataframe has the following numeric attributes:

- age
- fnlwgt
- education-num
- capital-gain
- capital-loss
- hours-per-week

Similarity metric for numeric attributes:

$s_{ij}^{(f)} = 1 - \frac{|x_{if}-x_{jf}|}{max_h x_{hf}-min_h x_{hf}}$, where h runs over all non-missing objects for attribute f.

The function for calculating the similarity metric of numeric attributes is as follows:

In [18]:
```
def func_sim_numeric(attr):
    print("Enter", attr, "of person 1: ")
    val1 = int(input())
    print("Enter", attr, "of person 2: ")
    val2 = int(input())
    max_h_x_hf = max(df_adult[attr])
    min_h_x_hf = min(df_adult[attr])
    sim = 1 - ((abs(val1-val2))/(max_h_x_hf - min_h_x_hf))
    return (sim)
```

Sub-section 1: Calculating similarity metric for age attribute:

In [21]:
```
attr = "age"
sim_metric_age = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the age attribute is
```

```
Enter age of person 1:
28
Enter age of person 2:
53
The similarity metric of the two values entered for the age attribute is  0.6575
342465753424
```

Sub-section 2: Calculating similarity metric for workclass attribute:

In [22]:
```
attr = "workclass"
sim_metric_workclass = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the", attr, "attribut
```

```
Enter workclass of person 1:
State-gov
Enter workclass of person 2:
Never-worked
The similarity metric of the two values entered for the workclass attribute is
0
```

Sub-section 3: Calculating similarity metric for fnlwgt attribute:

In [23]:
```
attr = "fnlwgt"
sim_metric_fnlwgt = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the fnlwgt is ", sim_
```

```
Enter fnlwgt of person 1:
77516
Enter fnlwgt of person 2:
257302
The similarity metric of the two values entered for the fnlwgt is  0.87789761073
60672
```

Sub-section 4: Calculating similarity metric for education attribute:

In [24]:
```
attr = 'education'
sim_metric_edlevel = func_sim_ordinal(attr)
print("The similarity metric of the two values entered for the education attribu
```

```
Enter education level of person 1:
Preschool
Enter education level of person 2:
Masters
The similarity metric of the two values entered for the education attribute is
0.06666666666666665
```

Sub-section 5: Calculating similarity metric for education-num attribute:

In [25]:
```
attr = 'education-num'
sim_metric_ednum = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the education-num att
```

```
Enter education-num of person 1:
12
Enter education-num of person 2:
7
The similarity metric of the two values entered for the education-num attribute
is  0.6666666666666667
```

Sub-section 6: Calculating similarity metric for marital-status attribute:

In [26]:
```
attr = 'marital-status'
sim_metric_marstat = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the marital status at
```

```
Enter marital-status of person 1:
Never-married
Enter marital-status of person 2:
Never-married
The similarity metric of the two values entered for the marital status attribute
is  1
```

Sub-section 7: Calculating similarity metric for occupation attribute:

In [27]:
```
attr = 'occupation'
sim_metric_occ = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the marital status at
```

```
Enter occupation of person 1:
Transport-moving
Enter occupation of person 2:
Sales
The similarity metric of the two values entered for the marital status attribute
is  0
```

Sub-section 8: Calculating similarity metric for relationship attribute:

In [28]:
```
attr = 'relationship'
sim_metric_rel = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the relationship attr
```

```
Enter relationship of person 1:
Own-child
Enter relationship of person 2:
Wife
The similarity metric of the two values entered for the relationship attribute i
s  0
```

Sub-section 9: Calculating similarity metric for race attribute:

In [29]:
```
attr = 'race'
sim_metric_race = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the race attribute is
```

```
Enter race of person 1:
White
Enter race of person 2:
Black
The similarity metric of the two values entered for the race attribute is  0
```

Sub-section 10: Calculating similarity metric for sex attribute:

In [30]:
```
attr = 'sex'
sim_metric_sex = func_sim_binary(attr)
print("The similarity metric of the two values entered for the sex attribute is
```

```
Enter sex of person 1:
Male
Enter sex of person 2:
Female
The similarity metric of the two values entered for the sex attribute is  0
```

Sub-section 11: Calculating similarity metric for capital-gain attribute:

In [31]:
```
attr = 'capital-gain'
sim_metric_capgain = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the capital gain attr
```

```
Enter capital-gain of person 1:
2174
Enter capital-gain of person 2:
15024
The similarity metric of the two values entered for the capital gain attribute i
s  0.8714987149871498
```

Sub-section 12: Calculating similarity metric for capital-loss attribute:

In [32]:
```
attr = 'capital-loss'
sim_metric_caploss = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the capital loss attr
```

```
Enter capital-loss of person 1:
4356
Enter capital-loss of person 2:
0
The similarity metric of the two values entered for the capital loss attribute i
s  0.0
```

Sub-section 13: Calculating similarity metric for hours-per-week attribute:

In [33]:
```
attr = 'hours-per-week'
sim_metric_hrsprwk = func_sim_numeric(attr)
print("The similarity metric of the two values entered for the hours per week at
```

```
Enter hours-per-week of person 1:
13
Enter hours-per-week of person 2:
40
The similarity metric of the two values entered for the hours per week attribute
is  0.7244897959183674
```

Sub-section 14: Calculating similarity metric for native-country attribute:

In [34]:
```
attr = 'native-country'
sim_metric_natcountry = func_sim_nominal(attr)
print("The similarity metric of the two values entered for the native country at
```

```
Enter native-country of person 1:
Cuba
Enter native-country of person 2:
Cuba
The similarity metric of the two values entered for the native country attribute
is  1
```

Question 2:

Suppose that the data set contains p attributes of mixed type. The dissimilarity d(i, j) between objects i and j is defined as

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

where the indicator $\delta_{ij}^{(f)} = 0$ if either:

1. x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute f for object i or object j)
2. $x_{if} = x_{jf} = 0$ and attribute f is asymmetric binary

otherwise, $\delta_{ij}^{(f)} = 1$.

Sub-section 15: Calculating dissimilarity metric of two records:

In [35]:
```
def func_dissim_nominal_binary(attr, rec1, rec2):

    val1 = df_adult[attr][rec1]
    val2 = df_adult[attr][rec2]

    if(val1 == np.nan or val1 == np.nan):
        delta_ij = 0
    else:
        delta_ij = 1

    if(val1 == val2):
        d_ij = 0
    else:
        d_ij = 1

    return (delta_ij*d_ij, delta_ij)

def func_dissim_ordinal(attr, rec1, rec2):

    val1 = df_adult[attr][rec1]
    val2 = df_adult[attr][rec2]

    if(val1 == np.nan or val1 == np.nan):
```

```
        delta_ij = 0
    else:
        delta_ij = 1

    df_adult = pd.read_csv("/Users/haileythanki/Downloads/adult.csv", index_col=
    education_replacements = {'Preschool':1, '1st-4th':2, '5th-6th':3, '7th-8th'
    'Assoc-voc':12, 'Some-college':13, 'Bachelors':14,  'Masters':15, 'Doctorate
    df_adult = df_adult.replace(education_replacements)

    max_h_x_hf = max(df_adult['education'])
    min_h_x_hf = min(df_adult['education'])

    d_ij = ((abs(education_replacements.get(edlevel1)-education_replacements.get
    return (delta_ij*d_ij, delta_ij)


def func_dissim_numeric(attr, rec1, rec2):

    val1 = df_adult[attr][rec1]
    val2 = df_adult[attr][rec2]

    if(val1 == np.nan or val1 == np.nan):
        delta_ij = 0
    else:
        delta_ij = 1

    max_h_x_hf = max(df_adult[attr])
    min_h_x_hf = min(df_adult[attr])

    d_ij = ((abs(val1-val2))/(max_h_x_hf - min_h_x_hf))

    return (delta_ij*d_ij, delta_ij)
```

In [38]:

```
def sim_metric_calculator(rec1, rec2):

    delta_d = 0
    delta = 0

    age_delta_d, age_delta = func_dissim_numeric('age', rec1, rec2)

    workclass_delta_d, workclass_delta = func_dissim_nominal_binary('workclass',

    fnlwgt_delta_d, fnlwgt_delta = func_dissim_numeric('fnlwgt', rec1, rec2)

    education_delta_d, education_delta = func_dissim_nominal_binary('education',

    ednum_delta_d, ednum_delta = func_dissim_numeric('education-num', rec1, rec2

    marstat_delta_d, marstat_delta = func_dissim_nominal_binary('marital-status'

    occupation_delta_d, occupation_delta = func_dissim_nominal_binary('occupatio

    relationship_delta_d, relationship_delta = func_dissim_nominal_binary('relat

    race_delta_d, race_delta = func_dissim_nominal_binary('race', rec1, rec2)

    sex_delta_d, sex_delta = func_dissim_nominal_binary('sex', rec1, rec2)

    capgain_delta_d, capgain_delta = func_dissim_numeric('capital-gain', rec1, r
```

```
    caploss_delta_d, caploss_delta = func_dissim_numeric('capital-loss', rec1, r

    hrsperwk_delta_d, hrsperwk_delta = func_dissim_numeric('hours-per-week', rec

    natcountry_delta_d, natcountry_delta = func_dissim_nominal_binary('native-co


    delta_d = age_delta_d + workclass_delta_d + fnlwgt_delta_d + education_delta
    + occupation_delta_d + relationship_delta_d + race_delta_d + sex_delta_d + c
    + hrsperwk_delta_d + natcountry_delta_d

    delta = age_delta + workclass_delta + fnlwgt_delta + education_delta + ednum
    + occupation_delta + relationship_delta + race_delta + sex_delta + capgain_d
    + natcountry_delta

    print("The similarity metric of the 2 records is: ",1 - delta_d/delta)
```

In [39]:

```
rec1 = int(input("Enter index of record #1: ", ))
rec2 = int(input("Enter index of record #2: ", ))

sim_metric_calculator(rec1, rec2)
```

```
Enter index of record #1: 3
Enter index of record #2: 9
The similarity metric of the 2 records is:  0.7330322975272776
```

In [ ]: