Beilei Guo, Liwei Jiang

Prof. Thaddeus Pawlicki

CSC240

6th February 2021

Homework 2

In this assignment, we use pandas as a helper for primary data processing.

```
31 import pandas as pd
```

```
32 # read in original data as a dataframe
   df = pd.read_csv('adult.data', header=None, skipinitialspace=True)
   df.head()
```

32

	0	1	2	3	4	5	6	7	8	9
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female

```
33 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       32561 non-null  int64
 1   1       32561 non-null  object
 2   2       32561 non-null  int64
 3   3       32561 non-null  object
 4   4       32561 non-null  int64
 5   5       32561 non-null  object
 6   6       32561 non-null  object
 7   7       32561 non-null  object
 8   8       32561 non-null  object
 9   9       32561 non-null  object
 10  10      32561 non-null  int64
 11  11      32561 non-null  int64
 12  12      32561 non-null  int64
 13  13      32561 non-null  object
 14  14      32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

For the similarity for each attribute, we first classified them into different types. for input x, y of attribue f,

- If x, y are numeric, our solution is to calculate the distance between x and y first, then divide it by the range of f. The attributes are applied to this solution include age, fnlwgt, education num, capital gain, capital loss, and hours per week.

```
37 max_sample = df.max()
   min_sample = df.min()
   max_mins = [(max_sample[i], min_sample[i]) for i in range(len(max_sample))]
   max_mins
```

```
37 [(90, 17),
    ('Without-pay', '?'),
    (1484705, 12285),
    ('Some-college', '10th'),
    (16, 1),
    ('Widowed', 'Divorced'),
    ('Transport-moving', '?'),
    ('Wife', 'Husband'),
    ('White', 'Amer-Indian-Eskimo'),
    ('Male', 'Female'),
    (99999, 0),
    (4356, 0),
    (99, 1),
    ('Yugoslavia', '?'),
    ('>50K', '<=50K')]
```

```
38 #Continuous Data
   def continuous_two_values(x,y, max_min):
       if x == '?' or x == '?':
           return 0

       values_max = max_min[0]
       values_min = max_min[1]

       return 1-abs(x-y)/(values_max-values_min)
```

```
39 continuous_two_values(215646, 83311, max_mins[2])
```

```
39 0.9101241493595578
```

- If x, y are nominal or binary, our solution is to test whether x is the same as y. If they are the same, return 1, else return 0. The attributes are applied to this solution include a binary attribute - sex, and 8 nominal attributes - workclass, education, marital status, occupation, relationship, race, native country.

```
40 #Binary data
   def binary_two_values(x,y):
       if x == y and (x != '?' or x != '?'):
           return 1
       else:
           return 0
```

```
34 #Function for nominal data
   #For attributes:
   def nominal_two_values(x,y):
       if x == y and (x != '?' or x != '?'):
           return 1
       else:
           return 0

35 nominal_two_values('Private','Self-emp-not-inc')
35 0

36 nominal_two_values('Private','Private')
36 1
```

- If x, y are ordinal, our solution is first to transform them into numeric data and treat them the same way as numeric data. Since the data set does not include such type, it is not used.

```
41 #Ordinal data
   def ordinal_two_values(x,y,values):
       if x == '?' or x == '?':
           return 0

       m = len(values)-1
       r_x = values.index(x)
       r_y = values.index(y)
       n_x = float(r_x-1)/(m-1)
       n_y = float(r_y-1)/(m-1)

       return 1-abs(n_x-n_y)/m
```

Whenever any of the functions mentioned previously meets a missing input, it would instantly return 0.

The overall data set's similarity contains 14 attributes of mixed types - 6 numeric attributes, 8 nominal attributes, and 1 binary attribute. So, we followed the approach in 2.4.6 of Han's book. We first marked each column with its attributes type. The overall similarity

function would select different functions to apply to each attribute based on those marks. Since the book mentioned that when any attribute is missing, its indicator would be 0. We followed this instruction, but since this data set does not include asymmetric binary attributes, we didn't write additional procedures for it.

```
42 def mixed_attributes_two_values(x, y, data_frame=df, ranges=max_mins):
       distance = 0.0
       delta = 0
       types = ['continuous','nominal','continuous','nominal','continuous',
                'nominal','nominal','nominal','nominal','binary','continuous',
                'continuous','continuous','nominal']
       for i in range(14):
           xi, yi = x[i], y[i]
           if xi!='?' or yi!='?':
               t = types[i]
               if t == 'continuous':
                   distance += continuous_two_values(xi, yi, ranges[i])
               if t == 'nominal':
                   distance += nominal_two_values(xi, yi)
               if t == 'binary':
                   distance += binary_two_values(xi, yi)
               if t == 'ordinal':
                   distance += ordinal_two_values(xi, yi, ranges[i])
               delta += 1
       return distance/delta

43 mixed_attributes_two_values(df.iloc[2],df.iloc[3])

43 0.6891594536564108
```