Damin Zhang
Computer Science 348
1/29/2019

Submission

Problem 1

Part 1:
Code:

```
import numpy as np
import pandas as pd

d = pd.read_csv("data/census.csv")
d = pd.DataFrame(data=d)
```

Result:

```
   Age          Workclass   Education       Marital Status  \
0   39          State-gov   Bachelors        Never-married
1   50   Self-emp-not-inc   Bachelors   Married-civ-spouse
2   38            Private     HS-grad             Divorced
3   53            Private        11th   Married-civ-spouse
4   28            Private   Bachelors   Married-civ-spouse

          Occupation    Relationship  Capital-Gain  Capital-Loss  \
0       Adm-clerical   Not-in-family          2174             0
1    Exec-managerial         Husband             0             0
2  Handlers-cleaners   Not-in-family             0             0
3  Handlers-cleaners         Husband             0             0
4     Prof-specialty            Wife             0             0

   Hours-Per-Week Country-of-Origin Annual Income
0              40     United-States         <=50K
1              13     United-States         <=50K
2              40     United-States         <=50K
3              40     United-States         <=50K
4              40              Cuba         <=50K
```

Part 2:

Age: Ratio Scale. It has zero value and the difference between values is meaningful.
Workclass: Nominal. It shows the value but cannot be ranked.
Education: Ordinal. Education levels can be ranked, but not arithmetic transformation.
Marital Status: Nominal. The status value is meaningful but cannot be ranked.
Occupation: Nominal. Same as above, the value is meaningful but cannot be ranked.
Relationship: Nominal. The value is meaningful but cannot be ranked.
Capital-gain: Ratio Scale. It has zero value and the difference between values is meaningful.

Capital-loss: Ratio Scale. It has zero value and the difference between values is meaningful.

Hours-per-week: Ratio Scale. It has zero value and differences between values are comparable.

Country-of-origin: Nominal. It shows as categorical variable.

Annual Income: Ordinal. As the dataset shows, 'Annual Income' is divided into two categories: '<=50K' and '>50K'. There is rank between these two categories and the values are meaningful.

Part 3:

Integer and float show that the variable possibly has zero point and the difference between values is meaningful, so the statistical type could be either interval or ratio scale. String type variable could be either nominal or ordinal. It depends on whether the value of the variable is comparable or not. If it's comparable, then it will be ordinal, otherwise nominal.

According to the 'census.csv' file, 'Annual Income' could be easily recognized as integer or float, which means it's either interval or ratio scale. However, it's pandas datatype is string and falls into ordinal statistical type.

Part 4:

Code:

```
# --- 1 if went to college, 0 if not ---
conditions=[
        d['Education'].str.strip()=='Associates',
        d['Education'].str.strip()=='Bachelors',
        d['Education'].str.strip()=='Masters',
        d['Education'].str.strip()=='Doctorate']
choices=[1,1,1,1]
d['College-Degree']=np.select(conditions,choices,0)

# --- change object to numeric value ---
d['Annual Income'].loc[d['Annual Income'].str.strip()=='<=50K']=49000
d['Annual Income'].loc[d['Annual Income'].str.strip()=='>50K']=51000

# --- calculate total income and count ---
cincome=0
ccount=0
nincome=0
ncount=0
for index, row in d.iterrows():
if row['College-Degree']==1:
        cincome += row['Annual Income']
        ccount += 1
else:
```

```
        nincome += row['Annual Income']
        ncount += 1

# --- average num ---
c_avg=cincome/ccount
n_avg=nincome/ncount
print(c_avg > n_avg)
```

Result:

```
True
```

Since the ‘Annual Income’ only has categorical variable, I convert it into numeric variable by using 49000 for ‘<=50K’ and 51000 for ‘>50K’. The result indicates that the average income for a person with college degree is greater than the average income for a person without college degree.

Part 5:

Code:

```
# --- load new dataframe, replace '?' by NaN ---
d=pd.read_csv("data/census.csv", sep=',', na_values=[' ?'], engine='python')

# --- construct copies ---
rm_occ_copy=d.copy()
rm_cou_copy=d.copy()

# --- remove rows with NaN value ---
rm_occ_copy=rm_occ_copy[pd.notnull(rm_occ_copy['Occupation'])]
rm_cou_copy=rm_cou_copy[pd.notnull(rm_cou_copy['Country-of-Origin'])]

# --- Gain, loss and count for original dataframe ---
origin_gain=0
origin_loss=0
origin_count=0

# --- Gain, loss and count for dataframe without NaN occupation value ---
copy1_gain=0
copy1_loss=0
copy1_count=0

# --- Gain, loss and count for datafram without NaN country-of-origin value ---
copy2_gain=0
copy2_loss=0
copy2_count=0

for index, row in d.iterrows():
```

```
        origin_gain += row['Capital-Gain']
        origin_loss += row['Capital-Loss']
        origin_count += 1

    for index, row in rm_occ_copy.iterrows():
        copy1_gain += row['Capital-Gain']
        copy1_loss += row['Capital-Loss']
        copy1_count += 1

    for index, row in rm_cou_copy.iterrows():
        copy2_gain += row['Capital-Gain']
        copy2_loss += row['Capital-Loss']
        copy2_count += 1

    origin_avg_gain=origin_gain/origin_count
    origin_avg_loss=origin_loss/origin_count

    rm_occ_avg_gain=copy1_gain/copy1_count
    rm_occ_avg_loss=copy1_loss/copy1_count

    rm_cou_avg_gain=copy2_gain/copy2_count
    rm_cou_avg_loss=copy2_loss/copy2_count

    print('Average origin gain: '+str(origin_avg_gain))
    print('Average origin loss: '+str(origin_avg_loss))
    print('Average rm occupation gain: '+str(rm_occ_avg_gain))
    print('Average rm occupation loss: '+str(rm_occ_avg_loss))
    print('Average rm country gain: '+str(rm_cou_avg_gain))
    print('Average rm country loss: '+str(rm_cou_avg_loss))
```

Result:

```
Average origin gain: 1077.6488437087312
Average origin loss: 87.303829734959
Average rm occupation gain: 1106.0370792369295
Average rm occupation loss: 88.91021550882219
Average rm country gain: 1064.3606229282632
Average rm country loss: 86.73935205453749
```

The result indicates that the estimates of ‘Capital-Gain’ and ‘Capital-Loss’ increase when we omit the rows with missing ‘Occupation’ variables but decrease when we omit the rows with missing ‘Country-of-origin’ variables.

It’s reasonable to filter out rows with missing ‘Country-of-origin’ variables, since a person must be born in a country. However, removing rows with missing ‘Occupation’ doesn’t make sense since a person could be unemployed. Missing values for attributes should be taken carefully since sometimes it’s meaningful. Thus, filtering with different variables produces different results.

In this part, removing missing data is not completely appropriate. Handling of missing data should depend on the data statistical type. If categorical attribute has missing values, we could just label it as “Missing”. If numeric attribute has missing values, we should flag it as “Missing” and then set it to 0.

Problem 2

Part 1:
Code:

```
file1=np.genfromtxt("data/synthetic1.csv",delimiter=',',dtype=None)
file2=np.genfromtxt("data/synthetic2.csv",delimiter=',',dtype=None)
file3=np.genfromtxt("data/synthetic3.csv",delimiter=',',dtype=None)
file4=np.genfromtxt("data/synthetic4.csv",delimiter=',',dtype=None)
```

Part 2:
Code:

```
hist1=np.histogram(file1)
print(hist1)
hist2=np.histogram(file2)
print(hist2)
hist3=np.histogram(file3)
print(hist3)
hist4=np.histogram(file4)
print(hist4)
```

Result:

```
# --- 'synthetic1.csv' ---
(array([   14,   406,  3603, 15675, 31981, 30832, 14141,  3018,   310,
          20]), array([-4.45563583, -3.55623941, -2.65684299, -1.75744657,
 -0.85805015,
        0.04134627,  0.94074269,  1.84013911,  2.73953554,  3.63893196,
        4.53832838]))

# --- 'synthetic2.csv' ---
(array([  671,  2636,  8004, 15964, 22511, 22739, 15899,  7942,  2718,
         667]), array([-2.99441180e+00, -2.39511268e+00, -1.79581356e+00,
-1.19651444e+00,
       -5.97215319e-01,  2.08380241e-03,  6.01382923e-01,  1.20068204e+00,
        1.79998117e+00,  2.39928029e+00,  2.99857941e+00]))

# --- 'synthetic3.csv' ---
(array([36532, 37087, 18394,  6115,     0,  1482,   323,    60,     6,
           1]), array([0. , 0.8, 1.6, 2.4, 3.2, 4. , 4.8, 5.6, 6.4, 7.2,
8. ]))

# --- 'synthetic4.csv' ---
(array([ 9984,  9992,  9855, 10028,  9949, 10068, 10117, 10036, 10041,
```

```
        9930]), array([-2.99995042e+00, -2.39996251e+00, -1.79997459e+00,
-1.19998668e+00,
       -5.99998772e-01, -1.08604697e-05,  5.99977051e-01,  1.19996496e+00,
        1.79995287e+00,  2.39994078e+00,  2.99992870e+00]))
```

‘synthetic1.csv’ matches as Normal distribution. The result indicates that the array starts at 14 and increases to peaks 31981 and 30832, then decreases in the same rate to 20.

‘synthetic2.csv’ matches as Truncated Normal distribution. Different from result1, the second result describes the same shape as the one in result1, but instead of starts and ends at small numbers, it starts and ends at relative larger number 671 and 667. Thus, it’s Truncated Normal distribution.

‘synthetic3.csv’ matches as Poisson distribution. By drawing the result on a piece of paper, the shape describes the array starts at large number 36532, and then decreases rapidly to 0. Even though there is a small bounce back to 1482, but eventually the array ends at 1. Overall, the shape is similar with Poisson distribution.

‘synthetic4.csv’ matches as Uniform distribution. As the result shows, that the differences between each interval are relatively small, the values vary between 9930 and 10068. Thus, it matches with the Uniform distribution.