hands-on-activity-6-1-2

March 10, 2024

1 Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python Name: Adornado, john louie V. Section: CPE22S3 Performed on: 03/08/2023 Submitted on: 03/10/2024 Submitted to: Engr. Roman M. Richard

#6.1 Intended Learning Outcome • Use pandas and numpy data analysis tools. • Demonstrate how to analyze data using numpy and pandas.

2 6.2 Resources

• Personal Computer • Jupyter Notebook • Internet Connection

#6.3 Supplementary Activities: ##Exercise 1 • Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

```
[ ]: import random
     random.seed(0)
     salaries = [round(random.random()*1000000, -3) for _ in range(100)]
     salaries
```

```
[ ]: [844000.0,
      758000.0,
      421000.0,
      259000.0,
      511000.0,
      405000.0,
      784000.0,
      303000.0,
      477000.0,
      583000.0,
      908000.0,
      505000.0,
      282000.0,
      756000.0,
      618000.0,
      251000.0,
      910000.0,
      983000.0,
810000.0,
902000.0,
310000.0,
730000.0,
899000.0,
684000.0,
472000.0,
101000.0,
434000.0,
611000.0,
913000.0,
967000.0,
477000.0,
865000.0,
260000.0,
805000.0,
549000.0,
14000.0,
720000.0,
399000.0,
825000.0,
668000.0,
1000.0,
494000.0,
868000.0,
244000.0,
325000.0,
870000.0,
191000.0,
568000.0,
239000.0,
968000.0,
803000.0,
448000.0,
80000.0,
320000.0,
508000.0,
933000.0,
109000.0,
551000.0,
707000.0,
547000.0,
814000.0,
540000.0,
964000.0,
603000.0,
588000.0,
445000.0,
596000.0,
385000.0,
576000.0,
290000.0,
189000.0,
187000.0,
613000.0,
657000.0,
477000.0,
90000.0,
758000.0,
877000.0,
923000.0,
842000.0,
898000.0,
923000.0,
541000.0,
391000.0,
705000.0,
276000.0,
812000.0,
849000.0,
895000.0,
590000.0,
950000.0,
580000.0,
451000.0,
660000.0,
996000.0,
917000.0,
793000.0,
82000.0,
613000.0,
486000.0]
```

#Mean

[ ]:

```
def mean(): # Define a function called mean
    total = 0
    for salary in salaries:  # Iterate
        total += salary
    salary_mean = total/len(salaries) # Calculating the mean by dividing the␣
 ↪total by the number of salaries
    print("Mean salary:", salary_mean) # Display
```

[ ]:

```
mean() # Calling the function
```

```
Mean salary: 585690.0
```

#Median

[ ]:
```python
def median():
    sort = sorted(salaries)  # Sort the salaries
    n = len(sort) # Find the total number of salaries
    m = n // 2 # Calculate the middle

    if n % 2:   # If odd, the median is the middle salary
        median = sort[m]
        print("Median salary is odd:", median)
    else:  # If even, calculate the median by averaging the two middle salaries
        median = (sort[m - 1] + sort[m]) / 2
        print("Median salary is even:", median)
```

[ ]:
```python
median() # Calling the function
```

```
Median salary is even: 589000.0
```

3 Mode

[ ]:
```python
def mode(salaries):
    salary_count = {}
    # Checking if the salary is already inside the dictionary
    # if not, set its count to 0
    # If it exists, increase its count by 1
    for num in salaries:
        salary_count[num] = salary_count.get(num, 0) + 1

    max_count = max(salary_count.values()) # Find the maximum count of among␣
  ↪all salaries
    mode_salaries = ', '.join(str(salary) for salary, count in salary_count.
  ↪items() if count == max_count) # Creating a list to store the salaries that␣
  ↪found with maximum counts
    print("The mode:", mode_salaries)
```

[ ]:
```python
mode(salaries) # Calling the function
```

```
The mode: 477000.0
```

#Sample Variance

[ ]:
```python
def sv():
    mean = sum(salaries)/len(salaries) # Calculate the mean (average) of the␣
  ↪salaries.
    sum_ = 0 # store the sum of squared differences from the mean.
    for salary in salaries:
        sum_ += (salary - mean) ** 2 # Add the squared difference between each␣
↪salary, mean to the sum.

    sample_var = sum_/(len(salaries) - 1) # Calculate the sample variance
    print("Sample Variance: ", sample_var)
```

```
[ ]: sv()
```

```
Sample Variance:  70664054444.44444
```

#Sample Standard Deviation

```
[ ]: def ssd():
         mean = sum(salaries)/len(salaries) # Calculate the mean (average) of the␣
     ↪salaries.
         sum_ = 0 # store the sum of squared differences from the mean.
         for salary in salaries:
             sum_ += (salary - mean) ** 2 # Add the squared difference between each␣
     ↪salary, mean to the sum.

         sample_var = sum_/(len(salaries) - 1)
         std_deviation = sample_var ** 0.5 # Calculate the standard deviation
         print("Sample Standard Deviation:", std_deviation)
```

```
[ ]: ssd()
```

```
Sample Standard Deviation: 265827.11382484
```

4 Code from the statistics module

```
[ ]: from statistics import mean

     mean(salaries)
```

```
[ ]: 585690.0
```

```
[ ]: from statistics import median as mediann

     mediann(salaries)
```

```
[ ]: 589000.0
```

```
[ ]: from statistics import mode as mde

     mde(salaries)
```

```
[ ]: 477000.0
```

[92]:

```
from statistics import variance

variance(salaries)
```

[92]:

```
70664054444.44444
```

[93]:

```
from statistics import stdev

stdev(salaries)
```

[93]:

```
265827.11382484
```

#Exercise 2 Using the same data, calculate the following statistics using the functions in the statistics module where appropriate: • Range • Coefficient of variation Interquartile range • Quartile coefficient of dispersion

#Range

[ ]:

```
def rangee():
    data_range = max(salaries) - min(salaries)
    print("range: ", data_range)

rangee()
```

```
range:  995000.0
```

#Coefficient of variation Interquartile range

[ ]:

```
def cv(salaries): # Function to calculate Coefficient of Variation (CV)
    return ssd(salaries) / mean(salaries)

def ir(salaries): # Function to Interquartile Range (CV)
    sorted_salaries = sorted(salaries)
    n = len(sorted_salaries)
    q1 = median(sorted_salaries[:n // 2])
    q3 = median(sorted_salaries[n // 2:])
    ir = q3 - q1
    return ir

cv_result = cv(salaries)
ir_result = ir(salaries)

print("Coefficient of Variation (CV):", cv_result)
print("Interquartile Range (IQR):", ir_result)
```

```
Coefficient of Variation (CV): 0.45386998894439035
Interquartile Range (IQR): 417500.0
```

#Quartile coefficient of dispersion

[ ]:

```
def qcd(salaries):
    return ir(salaries)/(2*median(salaries))
qcd_result = qcd(salaries)

print("Quartile coefficient of dispersion: ", qcd_result)
```

```
Quartile coefficient of dispersion:  0.35441426146010185
```

#Exercise 3: Pandas for Data Analysis • Load the diabetes.csv file. Convert the diabetes.csv into dataframe • Perform the following tasks in the diabetes dataframe: • Identify the column names • Identify the data types of the data • Display the total number of records • Display the first 20 records • Display the last 20 records • Change the Outcome column to DiagnosisIn [ ]: • Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes" • Create a new dataframe "withDiabetes" that gathers data with diabetes • Create a new dataframe "noDiabetes" thats gathers data with no diabetes • Create a new dataframe "Pedia" that gathers data with age 0 to 19 • Create a new dataframe "Adult" that gathers data with age greater than 19 • Use numpy to get the average age and glucose value. • Use numpy to get the median age and glucose value. • Use numpy to get the middle values of glucose and age. • Use numpy to get the standard deviation of the skinthickness.

[ ]:

```
import pandas
import numpy

diab_data = pandas.read_csv('diabetes.csv')
diab_data
```

[ ]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …             …             …       …    …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        …  …      …
763                     0.171   63        0
764                     0.340   27        0
```

```
765                        0.245   30        0
766                        0.349   47        1
767                        0.315   23        0

[768 rows x 9 columns]
```

#Identify the column names

```
[ ]: diab_data.columns
```

```
[ ]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
            'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
           dtype='object')
```

#Identify the data types of the data

```
[ ]: diab_data.dtypes
```

```
[ ]: Pregnancies                   int64
     Glucose                       int64
     BloodPressure                 int64
     SkinThickness                 int64
     Insulin                       int64
     BMI                         float64
     DiabetesPedigreeFunction    float64
     Age                           int64
     Outcome                       int64
     dtype: object
```

5 Display the total number of records

```
[ ]: print("total records of diabetes: ", len(diab_data))
```

```
total records of diabetes:  768
```

6 Display the first 20 records

```
[ ]: diab_data.head(20)
```

```
[ ]:     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
     0             6      148             72             35        0  33.6
     1             1       85             66             29        0  26.6
     2             8      183             64              0        0  23.3
     3             1       89             66             23       94  28.1
     4             0      137             40             35      168  43.1
     5             5      116             74              0        0  25.6
```

```
6             3       78             50             32       88  31.0
7            10      115              0              0        0  35.3
8             2      197             70             45      543  30.5
9             8      125             96              0        0   0.0
10            4      110             92              0        0  37.6
11           10      168             74              0        0  38.0
12           10      139             80              0        0  27.1
13            1      189             60             23      846  30.1
14            5      166             72             19      175  25.8
15            7      100              0              0        0  30.0
16            0      118             84             47      230  45.8
17            7      107             74              0        0  29.6
18            1      103             30             38       83  43.3
19            1      115             70             30       96  34.6

    DiabetesPedigreeFunction  Age  Outcome
0                      0.627   50        1
1                      0.351   31        0
2                      0.672   32        1
3                      0.167   21        0
4                      2.288   33        1
5                      0.201   30        0
6                      0.248   26        1
7                      0.134   29        0
8                      0.158   53        1
9                      0.232   54        1
10                     0.191   30        0
11                     0.537   34        1
12                     1.441   57        0
13                     0.398   59        1
14                     0.587   51        1
15                     0.484   32        1
16                     0.551   31        1
17                     0.254   31        1
18                     0.183   33        0
19                     0.529   32        1
```

#Display the last 20 records

```
[ ]: diab_data.tail(20)
```

```
[ ]:      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
     748            3      187             70             22      200  36.4
     749            6      162             62              0        0  24.3
     750            4      136             70              0        0  31.2
     751            1      121             78             39       74  39.0
     752            3      108             62             24        0  26.0
```

```
753            0      181             88             44      510  43.3
754            8      154             78             32        0  32.4
755            1      128             88             39      110  36.5
756            7      137             90             41        0  32.0
757            0      123             72              0        0  36.3
758            1      106             76              0        0  37.5
759            6      190             92              0        0  35.5
760            2       88             58             26       16  28.4
761            9      170             74             31        0  44.0
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
748                     0.408   36        1
749                     0.178   50        1
750                     1.182   22        1
751                     0.261   28        0
752                     0.223   25        0
753                     0.222   26        1
754                     0.443   45        1
755                     1.057   37        1
756                     0.391   39        0
757                     0.258   52        1
758                     0.197   26        0
759                     0.278   66        1
760                     0.766   22        0
761                     0.403   43        1
762                     0.142   33        0
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

7 Change the Outcome column to DiagnosisIn [ ]:

```
[ ]: diab_data.rename(columns = {'Outcome': 'Diagnosis'}, inplace = True)
     diab_data
```

```
[ ]:      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
     0              6      148             72             35        0  33.6
     1              1       85             66             29        0  26.6
```

```
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …      …            …            …      …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis
0                       0.627   50          1
1                       0.351   31          0
2                       0.672   32          1
3                       0.167   21          0
4                       2.288   33          1
..                        …  …        …
763                     0.171   63          0
764                     0.340   27          0
765                     0.245   30          0
766                     0.349   47          1
767                     0.315   23          0

[768 rows x 9 columns]
```

#Create a new column Classification that display “Diabetes” if the value of outcome is 1 , otherwise “No Diabetes”

[ ]:

```
diab_data["Classification"] = numpy.where(diab_data["Diagnosis"] == 1,␣
 ↪"Diabetes", "No Diabetes")
diab_data
```

[ ]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …      …            …            …      …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
0                       0.627   50          1       Diabetes
```

```
1                       0.351   31          0    No Diabetes
2                       0.672   32          1       Diabetes
3                       0.167   21          0    No Diabetes
4                       2.288   33          1       Diabetes
..                        …   …        …            …
763                     0.171   63          0    No Diabetes
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
766                     0.349   47          1       Diabetes
767                     0.315   23          0    No Diabetes

[768 rows x 10 columns]
```

#Create a new dataframe “withDiabetes” that gathers data with diabetes

[ ]:

```
diab_data = pandas.DataFrame(diab_data)
withDiabetes = diab_data[diab_data['Diagnosis'] == 1].copy()

withDiabetes
```

[ ]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
2              8      183             64              0        0  23.3
4              0      137             40             35      168  43.1
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
..           …      …            …            …      …   …
755            1      128             88             39      110  36.5
757            0      123             72              0        0  36.3
759            6      190             92              0        0  35.5
761            9      170             74             31        0  44.0
766            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Diagnosis Classification
0                       0.627   50          1       Diabetes
2                       0.672   32          1       Diabetes
4                       2.288   33          1       Diabetes
6                       0.248   26          1       Diabetes
8                       0.158   53          1       Diabetes
..                        …   …        …            …
755                     1.057   37          1       Diabetes
757                     0.258   52          1       Diabetes
759                     0.278   66          1       Diabetes
761                     0.403   43          1       Diabetes
766                     0.349   47          1       Diabetes

[268 rows x 10 columns]
```

8 Create a new dataframe “noDiabetes” thats gathers data with no diabetes

[ ]:

```
diab_data = pandas.DataFrame(diab_data)
Nodiab = diab_data[diab_data['Diagnosis'] == 0].copy()

Nodiab
```

[ ]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1              1       85             66             29        0  26.6
3              1       89             66             23       94  28.1
5              5      116             74              0        0  25.6
7             10      115              0              0        0  35.3
10             4      110             92              0        0  37.6
..           …       …             …             …       …     …
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
1                       0.351   31          0    No Diabetes
3                       0.167   21          0    No Diabetes
5                       0.201   30          0    No Diabetes
7                       0.134   29          0    No Diabetes
10                      0.191   30          0    No Diabetes
..                        …  …        …            …
762                     0.142   33          0    No Diabetes
763                     0.171   63          0    No Diabetes
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
767                     0.315   23          0    No Diabetes

[500 rows x 10 columns]
```

#Create a new dataframe “Pedia” that gathers data with age 0 to 19

[ ]:

```
diab_data = pandas.DataFrame(diab_data)
pedia = diab_data[diab_data['Age'] <= 19].copy()

pedia
```

[ ]:

```
Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age, Diagnosis, Classification]
Index: []
```

#Create a new dataframe “Adult” that gathers data with age greater than 19

[ ]:

```
diab_data = pandas.DataFrame(diab_data)
Adult = diab_data[diab_data['Age'] > 19].copy()

Adult
```

[ ]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …              …              …       …    …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
0                       0.627   50          1       Diabetes
1                       0.351   31          0    No Diabetes
2                       0.672   32          1       Diabetes
3                       0.167   21          0    No Diabetes
4                       2.288   33          1       Diabetes
..                        …    …         …              …
763                     0.171   63          0    No Diabetes
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
766                     0.349   47          1       Diabetes
767                     0.315   23          0    No Diabetes

[768 rows x 10 columns]
```

#Use numpy to get the average age and glucose value.

[ ]:

```
mean_age = numpy.mean(diab_data['Age'])
mean_glucose = numpy.mean(diab_data['Glucose'])


print("Average Age:", mean_age)
print("Average Glucose Value:", mean_glucose)
```

```
Average Age: 33.240885416666664
Average Glucose Value: 120.89453125
```

#Use numpy to get the median age and glucose value.

[ ]:

```
median_age = numpy.median(diab_data['Age'])
median_glucose = numpy.median(diab_data['Glucose'])


print("Median Age:", median_age)
print("Median Glucose Value:", median_glucose)
```

```
Average Age: 29.0
Average Glucose Value: 117.0
```

#Use numpy to get the middle values of glucose and age.

[ ]:

```
median_age = numpy.median(diab_data['Age'])
median_glucose = numpy.median(diab_data['Glucose'])


print("Median Age:", median_age)
print("Median Glucose Value:", median_glucose)
```

#Use numpy to get the standard deviation of the skinthickness.

[ ]:

```
skinthickness_std = numpy.std(diab_data['SkinThickness'])

print("Standard deviation of the skinthickness: ", skinthickness_std)
```

```
Standard deviation of the skinthickness:  15.941828626496939
```

#6.4 Conclusion

In this HOA, I think i got a little hang of it now at coding because we tried doing things without using special tools or bringing in extra stuff. We looked at our own code and compared it to what's already built into Python. We also practiced bringing in data from csv and using helpful tools like numpy and pandas. Using these tools made our work quicker and simpler, especially when we had clear instructions. So basically, we learned some tricks and figured out how to use built-in features more effectively.