Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Pisalbon, Ery Jay P.

section: CPE22S3

Performed and Submitted on:03/06/2003

Submitted to: Engr. Roman M. Richard

ILOs

1. use pandas and numpy data analysis tools.
2. Demonstrate how to analyze data using numpy and pandas

Resources

- Personal Computer
- Jupyter Notebook
- Internet Connection

Supplementary Activities

Exercise 1

Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

In [1]:
```
import random
import pandas as pd

random.seed(0)
"""make a list that contains 100 elements in it,
```

```
the elements were randomized and were rounded of to 3"""
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

In [2]:

```
# code for getting the table of the randomly generated numbers in the salaries list
salariestbale = pd.Series(salaries, name = 'Salaries')
salariestbale.index += 1 # the index should start at 1
print(salariestbale)
```

```
1      844000.0
2      758000.0
3      421000.0
4      259000.0
5      511000.0
         ...
96     917000.0
97     793000.0
98      82000.0
99     613000.0
100    486000.0
Name: Salaries, Length: 100, dtype: float64
```

In [3]:

```
# testing how the code works
test = random.random()*1000000
print(test)
testR = round(test,-3)
print(testR)
```

```
630147.3404114728
630000.0
```

Mean

In [4]:

```
# own code
def Mmean(data):
    sum = 0
    for i in data: # loop for adding the datas from the list passed on the funciton
        sum += i
    dmean = sum/len(data) # getting the mean of the list given by dividing it by its # of elements ( used the len() funciton)
    return dmean

print(Mmean(salaries))
```

```
585690.0
```

In [5]:
```
# code from the statistics module
from statistics import mean

mean(salaries)
```

Out[5]:
```
585690.0
```

Median

In [6]:
```
def Mmedian(data):
    data.sort() # sorts the data first
    median = (len(data)+1)/2 # gets the place value of the median
    if len(data)/2 == 1: # if statement if the number of salaries is odd (we can easily get the median)
        print('list is odd')
        Imedian = median - 1 # gets the index of the median in the list
        return data[Imedian]
    else: # else statement if the number of salaries is even
        print('list is even')
        LMedian = int(median) - 1 # getting the first number of the median (trunacating using int)
        RMedian = int(median)  # getting the second number of the median
        medianEven = (data[LMedian] + data[RMedian])/2 # gets the average number of the two number on the median
        return medianEven

Mmedian(salaries)
```

```
list is even
```

Out[6]:
```
589000.0
```

In [7]:
```
from statistics import median as mdn

mdn(salaries)
```

Out[7]:
```
589000.0
```

Mode

In [8]:
```
def mode(data):
    frequency = {}
    for num in data: # loop for tallying the frequencies in the list
        frequency[num] = frequency.get(num, 0) + 1
```

```
    max_freq = max(frequency.values())
    for val in frequency: # Loop for getting the number with the most frequence
        if frequency[val] == max_freq: # conditional statement if the frequency of the current value is the same as the max freq
            return val # return the element with the most frequencies

mode(salaries)
```

Out[8]: 477000.0

In [9]:
```
from statistics import mode as md

md(salaries)
```

Out[9]: 477000.0

Sample Variance

In [10]:
```
def variance(data):
    data.sort() #sorts the data
    sample_mean = Mmean(data) # gets the mean
    result = 0
    for i in data: # for Loop for getting the summation of square of the difference of the element and sample mean
        result += (i - sample_mean)**2
    return result / (len(data) - 1) # returns the solve sample variance

variance(salaries)
```

Out[10]: 70664054444.44444

In [11]:
```
from statistics import variance as vrnce

vrnce(salaries)
```

Out[11]: 70664054444.44444

Sample standard deviation

In [12]:
```
def SD(data):
    return variance(data)**0.5 # returns the square root of the variance
```

```
SD(salaries)
```

Out[12]:

```
265827.11382484
```

In [13]:

```
from statistics import stdev

stdev(salaries)
```

Out[13]:

```
265827.11382484
```

Exercise 2

Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

Range

In [14]:

```
data_range = max(salaries) - min(salaries)
print(data_range)
```

```
995000.0
```

Coefficient of variation interquartile range

In [15]:

```
def Cv(data):
    return SD(data) / Mmean(data)
def Iqr(data):
  # getting the quartiles
  q1 = Mmedian(data[:len(data) // 2])
  q3 = Mmedian(data[len(data) // 2:])

  # Handle equal quartile case for IQR
  iqr = 0
  if q1 != q3:
    iqr = q3 - q1

  return  iqr

# Calculate CV and IQR
iqr = Iqr(salaries)
```

```
cv = Cv(salaries)
# Print the results
print("Coefficient of Variation (CV):", cv)
print("Interquartile Range (IQR):", iqr)
```

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

Quartile coefficient of dispersion

In [16]:
```
def QCD(data):
    return Iqr(data) / (2 * Mmedian(data))

QCD(salaries)

print('QCD:', QCD(salaries))
```

```
list is even
list is even
list is even
list is even
list is even
list is even
QCD: 0.35441426146010185
```

Exercise 3

In [17]:
```
import pandas as pd
import numpy as np

diabetes_data = pd.read_csv('diabetes.csv')
diabetes_data.index += 1 # index should start in 1
diabetes_data #load the csv
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
764	10	101	76	48	180	32.9	0.171	63	0
765	2	122	70	27	0	36.8	0.340	27	0
766	5	121	72	23	112	26.2	0.245	30	0
767	1	126	60	0	0	30.1	0.349	47	1
768	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

1. Identify the column names

In [18]:
```
diabetes_data.columns
```

Out[18]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

2. Identify the data types of the data

In [19]:
```
diabetes_data.dtypes
```

Out[19]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

3. Display the total number of records

In [20]:
```
print('total number of records in diabetes.csv: ', len(diabetes_data))
```

```
total number of records in diabetes.csv:  768
```

4. Display the first 20 records

In [21]:
```
diabetes_data.head(20)
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31.0	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	1
10	8	125	96	0	0	0.0	0.232	54	1
11	4	110	92	0	0	37.6	0.191	30	0
12	10	168	74	0	0	38.0	0.537	34	1
13	10	139	80	0	0	27.1	1.441	57	0
14	1	189	60	23	846	30.1	0.398	59	1
15	5	166	72	19	175	25.8	0.587	51	1
16	7	100	0	0	0	30.0	0.484	32	1
17	0	118	84	47	230	45.8	0.551	31	1
18	7	107	74	0	0	29.6	0.254	31	1
19	1	103	30	38	83	43.3	0.183	33	0
20	1	115	70	30	96	34.6	0.529	32	1

5. Display the last 20 records

In [22]:
```
diabetes_data.tail(20)
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
749	3	187	70	22	200	36.4	0.408	36	1
750	6	162	62	0	0	24.3	0.178	50	1
751	4	136	70	0	0	31.2	1.182	22	1
752	1	121	78	39	74	39.0	0.261	28	0
753	3	108	62	24	0	26.0	0.223	25	0
754	0	181	88	44	510	43.3	0.222	26	1
755	8	154	78	32	0	32.4	0.443	45	1
756	1	128	88	39	110	36.5	1.057	37	1
757	7	137	90	41	0	32.0	0.391	39	0
758	0	123	72	0	0	36.3	0.258	52	1
759	1	106	76	0	0	37.5	0.197	26	0
760	6	190	92	0	0	35.5	0.278	66	1
761	2	88	58	26	16	28.4	0.766	22	0
762	9	170	74	31	0	44.0	0.403	43	1
763	9	89	62	0	0	22.5	0.142	33	0
764	10	101	76	48	180	32.9	0.171	63	0
765	2	122	70	27	0	36.8	0.340	27	0
766	5	121	72	23	112	26.2	0.245	30	0
767	1	126	60	0	0	30.1	0.349	47	1
768	1	93	70	31	0	30.4	0.315	23	0

6. Change Outcome column to Diagnosis

In [23]:

```
diabetes_data.rename(columns = {'Outcome':'Diagnosis'}, inplace = True) #ranaming the outcome column to diagnosis

diabetes_data
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
764	10	101	76	48	180	32.9	0.171	63	0
765	2	122	70	27	0	36.8	0.340	27	0
766	5	121	72	23	112	26.2	0.245	30	0
767	1	126	60	0	0	30.1	0.349	47	1
768	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

In [24]:

```
diabetes_data['Classification'] = np.where(diabetes_data['Diagnosis'] == 1, 'Diabetes', 'No Diabetes')

diabetes_data
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
1	6	148	72	35	0	33.6	0.627	50	1	Diabetes
2	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
3	8	183	64	0	0	23.3	0.672	32	1	Diabetes
4	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
5	0	137	40	35	168	43.1	2.288	33	1	Diabetes
...	...	...	...	...	...	...	...	...	...	...
764	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
765	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
766	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
767	1	126	60	0	0	30.1	0.349	47	1	Diabetes
768	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

768 rows × 10 columns

8. Create a new dataframe "withDiabetes" that gathers data with diabetes

In [25]:

```
withDiabetesDF = pd.DataFrame(diabetes_data[diabetes_data['Diagnosis'] == 1])

print(withDiabetesDF)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1              6      148             72             35        0  33.6
3              8      183             64              0        0  23.3
5              0      137             40             35      168  43.1
7              3       78             50             32       88  31.0
9              2      197             70             45      543  30.5
..           ...      ...            ...            ...      ...   ...
756            1      128             88             39      110  36.5
758            0      123             72              0        0  36.3
760            6      190             92              0        0  35.5
762            9      170             74             31        0  44.0
767            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Diagnosis Classification
1                       0.627   50          1       Diabetes
3                       0.672   32          1       Diabetes
5                       2.288   33          1       Diabetes
7                       0.248   26          1       Diabetes
9                       0.158   53          1       Diabetes
..                        ...  ...        ...            ...
756                     1.057   37          1       Diabetes
758                     0.258   52          1       Diabetes
760                     0.278   66          1       Diabetes
762                     0.403   43          1       Diabetes
767                     0.349   47          1       Diabetes

[268 rows x 10 columns]
```

9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

In [26]:

```
NoDiabetesDF = pd.DataFrame(diabetes_data[diabetes_data['Diagnosis'] == 0])

print(NoDiabetesDF)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
2              1       85             66             29        0  26.6
4              1       89             66             23       94  28.1
6              5      116             74              0        0  25.6
8             10      115              0              0        0  35.3
11             4      110             92              0        0  37.6
..           ...      ...            ...            ...      ...   ...
763            9       89             62              0        0  22.5
764           10      101             76             48      180  32.9
765            2      122             70             27        0  36.8
766            5      121             72             23      112  26.2
768            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
2                       0.351   31          0    No Diabetes
4                       0.167   21          0    No Diabetes
6                       0.201   30          0    No Diabetes
8                       0.134   29          0    No Diabetes
11                      0.191   30          0    No Diabetes
..                        ...  ...        ...            ...
763                     0.142   33          0    No Diabetes
764                     0.171   63          0    No Diabetes
765                     0.340   27          0    No Diabetes
766                     0.245   30          0    No Diabetes
768                     0.315   23          0    No Diabetes

[500 rows x 10 columns]
```

10. Create a new dataframe "Pedia" that gathers data with age 0 to 19

In [27]:

```
PediaDF = pd.DataFrame(diabetes_data[diabetes_data['Age'] <= 19])

print(PediaDF)
```

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

11. Create a new dataframe "Adult" that gathers data with age greater than 19

In [28]:

```
AdultDF = pd.DataFrame(diabetes_data[diabetes_data['Age'] > 19])
```

```
print(AdultDF)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1              6      148             72             35        0  33.6
2              1       85             66             29        0  26.6
3              8      183             64              0        0  23.3
4              1       89             66             23       94  28.1
5              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
764           10      101             76             48      180  32.9
765            2      122             70             27        0  36.8
766            5      121             72             23      112  26.2
767            1      126             60              0        0  30.1
768            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
1                       0.627   50          1       Diabetes
2                       0.351   31          0    No Diabetes
3                       0.672   32          1       Diabetes
4                       0.167   21          0    No Diabetes
5                       2.288   33          1       Diabetes
..                        ...  ...        ...            ...
764                     0.171   63          0    No Diabetes
765                     0.340   27          0    No Diabetes
766                     0.245   30          0    No Diabetes
767                     0.349   47          1       Diabetes
768                     0.315   23          0    No Diabetes

[768 rows x 10 columns]
```

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

In [29]:

```
glucose_ave = np.mean(diabetes_data['Glucose'])

print(f'average of glucose: {glucose_ave}')
```

```
average of glucose: 120.89453125
```

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

In [30]:

```
age_median = np.median(diabetes_data['Age'])
print(f'median of age column: {age_median}')
```

```
glucose_median = np.median(diabetes_data['Glucose'])
print(f'median of glucose column: {glucose_median}')
```

```
median of age column: 29.0
median of glucose column: 117.0
```

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

In [31]:
```
glucose_median = np.median(diabetes_data['Glucose'])
print(f'median of glucose column: {glucose_median}')

age_median = np.median(diabetes_data['Age'])
print(f'median of age column: {age_median}')
```

```
median of glucose column: 117.0
median of age column: 29.0
```

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

In [32]:
```
skinThickness_std = np.std(diabetes_data['SkinThickness'])
print(f'Standard deviation of skin thickness: {skinThickness_std}')
```

```
Standard deviation of skin thickness: 15.941828626496978
```

Conclusion

In this Hands-On activity we reviewed multiple topics (the mean,median,mode,etc.) from our past subjects/courses like math in the modern world from our first year, and Statistics and Probability from our Senior High School level, but in this activity we have used it, or applied those topics in coding and data science, in this activity I have concluded that the statistical tools that we have learned from school are all contained in the python library, those are the numPy and the statistics module,I also observed that the data set that we have used in this case the diabetes.csv file corresponds to a dictionary in python where the column headers are like the keys in the dictionary and the datas under those column are the values and lastly, I have also concluded that we can use our stats in the field of data science.