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)
```

Out[6]:
```
list is even
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 [17]:
```
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 [18]:
```
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 [ ]:
```
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
```

1. Identify the column names

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

2. Identify the data types of the data

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

3. Display the total number of records

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

4. Display the first 20 records

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

5. Display the last 20 records

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

6. Change Outcome column to Diagnosis

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

diabetes_data
```

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

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

diabetes_data
```

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

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

print(withDiabetesDF)
```

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

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

print(NoDiabetesDF)
```

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

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

print(PediaDF)
```

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

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

print(AdultDF)
```

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

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

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

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

In [ ]:
```
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}')
```

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

In [ ]:
```
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}')
```

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

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

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.