Hands-on Activity 6.1 Introduction to Data Analysis

CPE311 Computational Thinking with Python

Name: Garcia, John Carlos M.

Section: CPE22S3

Performed on: 03/06/2024

Submitted on: 03/07/2024

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcome

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

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)]
```

Using the data generated above, calculate the following statistics without importing anything from the statistics module in the standard library (https://docs.python.org/3/library/statistics.html) and then comfirm your results match up to those that are obtained when using the statistics module (where possible):

- Mean
- Median
- Mode (hint: check out the Counter in the collections module of the standard library at https://docs.python.org/3/library/collections.html#collections.Counter)
- Sample variance
- Sample standard deviation

```
1 #Write a comment per statistical function
2 import random
3 random.seed(0)
4 salaries = [round(random.random()*1000000, -3) for _ in range(100)]
5 print(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, 75600
```

```
#Calculating the Mean Manually
mean = sum(salaries)/len(salaries) # The sum(salaries) adds all the values in the list altogether, whilst the len(salaries) finds the num
print("Mean: ", mean)

#Calculating the Mean using the imported Statistics Module
import statistics
mean2 = statistics.mean(salaries)
print("Mean using imported module: ", mean2)
```

```
Mean:  585690.0
Mean using imported module:  585690.0
```

```
#Median
sortedSalaries = sorted(salaries) # We first sort the salaries so that the values are in ascending order
length = len(sortedSalaries) # Finds the length of the list
if length % 2 == 0:  # This means that the values are even, therefore no middle values, thus the need for the -1 to get the 2 values that
  median = (sortedSalaries[length//2 - 1] + sortedSalaries[length//2]) / 2
else:
  median = sortedSalaries[length // 2] # If the length of the salaries is odd, then that means there is a middle number. We just have to

print("Median: ", median)

#Median using the imported Statistics Module
import statistics
median2 = statistics.median(salaries)
print("Median using the module: ", median2)
```

```
Median:  589000.0
Median using the module:  589000.0
```

```
#Mode using Counter
from collections import Counter

counter = Counter(salaries) #This finds the number of times the value appeared in the list
print(counter)

max_count = max(counter.values()) #This finds the highest frequency present in the list
mode = [k for k, v in counter.items() if v == max_count]
#The k represents the values in the salary list, whilst the v represents the frequency of the said value
#This finds the value with the highest frequency in the list, or, the mode, as the v is == to max_count

print("Mode (Most Occuring Value):", mode)
print("Number of times occured: ", max_count)

#Mode using stat module
import statistics
mode1 = statistics.mode(salaries)
print("\nMode using module: ", mode1)
```

```
Counter({477000.0: 3, 758000.0: 2, 613000.0: 2, 923000.0: 2, 844000.0: 1, 421000.0: 1, 259000.0: 1, 511000.0: 1, 405000.0: 1, 784000.0:
Mode (Most Occuring Value): [477000.0]
Number of times occured:  3

Mode using module:  477000.0
```

```
# Sample Variance
salaryLength = len(salaries)  # To find the sample variance, we must first find the length of the list
mean = sum(salaries) / salaryLength  # Then calculate the mean of the salaries
squared_diff_sum = 0  # Then Initialize a variable to accumulate the squared differences

# We then Loop through each value in the salaries list and compute the sum of squared differences from the mean
for salary in salaries:
    squared_diff_sum += (salary - mean) ** 2

# Calculate the sample variance using the formula and divide by (n-1), or in this case, the length of the list minus 1
sample_variance = squared_diff_sum / (salaryLength - 1)

# Print the calculated sample variance
print("Sample Variance: ", sample_variance)

# Sample Variance using Statistics Module
import statistics
sample_variance1 = statistics.variance(salaries)
print("Sample Variance using statistics module: ", sample_variance1)

```

```
Sample Variance:  70664054444.44444
Sample Variance using statistics module:  70664054444.44444
```

```
#Sample Standard Deviation
standard_deviation = sample_variance ** 0.5 #We use the value of the sample_variance in the previous code to save up time and space
print("Standard Deviation: ", standard_deviation)

#Standard Deviation using the module
import statistics
standard_deviation1 = statistics.stdev(salaries)
print("Standard Deviation using the module: ", standard_deviation1)
```

```
Standard Deviation:  265827.11382484
Standard Deviation using the module:  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
range = max(salaries) - min(salaries) #We get the range by subtracting the maximum value in the salaries list with the minimum value
print("Range: ", range)
```

```
Range:  995000.0
```

```
#Coefficient of Variation
import statistics

mean = statistics.mean(salaries) #To get the Coefficient of Variation, we must first find the mean
standard_deviation = statistics.stdev(salaries) #Then the standard deviation using the statistics module

coefficient_of_variation = (standard_deviation / mean) * 100 #Then use the formula to solve it

print("Coefficient of Variation: ", coefficient_of_variation)
```

```
Coefficient of Variation:  45.38699889443903
```

```
#Interquartile Range
quartile = statistics.quantiles(salaries) #This calculates the quartiles of the salaries list using the quantiles function

iqr = quartile[2] - quartile[0]
#After finding the quartiles, we subtracts the first and third quartiles to calculate the interquartile range (IQR)
#The values inside the bracket represent the quartile. 0 is first quartile whilst 2 is the third quartile
print("Quartiles:", quartile)
print("Interquartile Range: ", iqr) #822250.0 - 400500.0 = 421750
```

```
Quartiles: [400500.0, 589000.0, 822250.0]
Interquartile Range:  421750.0
```

```
#Quartile Coefficient of Dispersion
import statistics
qcd = (quartile[2] - quartile[0]) / (quartile[2] + quartile[0]) #By using the same principle as in the iqr, we just have to use the formu
rounded_qcd = round(qcd, 4) #This is just to round the output to 4 decimal places
print("Quartile Coefficient of Dispersion: ", qcd)
print("Rounded version: ", rounded_qcd)
```

```
Quartile Coefficient of Dispersion:  0.34491923941934166
Rounded version:  0.3449
```

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:

1. Identify the column names
2. Identify the data types of the data
3. Display the total number of records
4. Display the first 20 records
5. Display the last 20 records
6. Change the Outcome column to Diagnosis
7. Create a new column Classification that displays "Diabetes" if the value of outcome is 1, otherwise "No Diabetes"
8. Create a new dataframe "withDiabetes" that gathers data with diabetes
9. Create a new dataframe "noDiabetes" that gathers data with no diabetes
10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
11. Create a new dataframe "Adult" that gathers data with age greater than 19
12. Use numpy to get the average age and glucose value
13. Use numpy to get the median age and glucose value.
14. Use numpy to get the middle values of glucose and age.
15. Use numpy to get the standard deviation of the skinthickness.

```
#Indicate which item you're answering with a comment
filepath = '/content/diabetes.csv'
import pandas as pd
import numpy as np
data = pd.read_csv(filepath)
data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 9 columns

Next steps: View recommended plots

```
#1. Identify the column names
column_names = data.columns
print(column_names)
```

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

```
#2. Identify the data types of the data
data_types = data.dtypes
print(data_types)
```

```
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
total_records = data.shape #data.shape outputs two values, the first one is the rows and the second one is the columns
rows = data.shape[0]  #utilized [0] to only print the first values in data.shape, which is the rows. Same goes with the columns.
columns = data.shape[1]
print("Total: ", total_records)
print("Rows: ", rows)
print("Columns: ", columns)
```

```
Total:  (768, 9)
Rows:  768
Columns:  9
```

```
#4. Display the first 20 records
first_20 = data.head(20)
print(first_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
```

```
1 #5. Display the last 20 records
2 last_20 = data.tail(20)
3 print(last_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
```

```
1 #6. Change the Outcome column to Diagnosis
2 column_rename = data.rename(columns={'Outcome':'Diagnosis'})
3 print(column_rename)
```

```
         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]
```

```
#7. Create a new column Classification that displays "Diabetes" if the value of outcome is 1, otherwise "No Diabetes"
data['Classification'] = data['Outcome'].apply(lambda x: 'Diabetes' if x == 1
                                               else 'No Diabetes')
print(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 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]
```

```
#8. Create a new dataframe "withDiabetes" that gathers data with diabetes
withDiabetes = data[data['Outcome'] == 1]
withDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

268 rows × 10 columns

Next steps: View recommended plots

```
#9. Create a new dataframe "noDiabetes" that gathers data with no diabetes
noDiabetes = data[data['Outcome'] == 0]
noDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

500 rows × 10 columns

Next steps: View recommended plots

```
1 #10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
2 Pedia = data[(data['Age'] >= 0) & (data['Age'] <= 19)]
3 Pedia
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc

```
1 #11. Create a new dataframe "Adult" that gathers data with age greater than 19
2 Adult = data[data['Age'] > 19]
3 Adult
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 10 columns

Next steps: View recommended plots

```
1 #12. Use numpy to get the average age and glucose value
2 ave_age = np.mean(data['Age'])
3 ave_glc = np.mean(data['Glucose'])
4
5 print("Average Age:", ave_age)
6 print("Average Glucose Value:", ave_glc)
```

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

```
1 #13. Use numpy to get the median age and glucose value.
2 med_age = np.median(data['Age'])
3 med_glc = np.median(data['Glucose'])
4
```

```
print("Median Age:", med_age)
print("Median Value of Glucose:", med_glc)
```

```
Median Age: 29.0
Median Value of Glucose: 117.0
```

```
#14. Use numpy to get the middle values of glucose and age.
sorted_age = np.argsort(data['Age']) #I first sorted the list so that I can accurately find the middle values
sorted_glc = np.argsort(data['Glucose'])

mid_age = len(sorted_age) // 2 #I then find the length of the list before doing doing floor division by 2 to find the middle values of the
mid_glc = len(sorted_glc) // 2

middle_age_row = data.iloc[sorted_age[mid_age]] #I then used the .iloc function to find the values in the whole middle row of the middle a
middle_glc_row = data.iloc[sorted_glc[mid_glc]]
print("Middle row if Age is sorted:\n",middle_age_row, "\n\n")
print("Middle row if Glucose is sorted:\n",middle_glc_row)
#We can see that the result correlates with the output in task 13, where I got the median age and glucose value, if we compare these value
```

```
Middle row if Age is sorted:
 Pregnancies                 2
Glucose                   139
BloodPressure              75
SkinThickness               0
Insulin                     0
BMI                      25.6
DiabetesPedigreeFunction ...
Age                        29
Outcome                     0
Classification      No Diabetes
Name: 433, dtype: object


Middle row if Glucose is sorted:
 Pregnancies                 0
Glucose                   117
BloodPressure              80
SkinThickness              31
Insulin                    53
BMI                      45.2
DiabetesPedigreeFunction ...
Age                        24
Outcome                     0
Classification      No Diabetes
Name: 229, dtype: object
```

```
#15. Use numpy to get the standard deviation of the skinthickness.
std_deviation_skinthickness = np.std(data['SkinThickness'])
print("The standard deviation of the skin thickness is: ", std_deviation_skinthickness)
```

```
The standard deviation of the skin thickness is:  15.941828626496939
```

6.4 Conclusion

I conclude that this hands-on-activity helped me familiarize myself with the basic functions and commands that are found in pandas and numpy. By practicing various specific tasks, I got a hold of the logic and mechanism behind the functions that I used to solve those tasks. This activity helped me practice how to load data in a csv file, manipulate the rows and columns present in the data, and utilize modules and functions to help me save time in calculating and measuring the statistical values and other computations in the given problems. While still not enough, this activity helped me at least gain confidence in my ability to work with simple data structures using Python, while I explored various functions and methods present in the statistics, pandas, and numpy libraries. I will make use of this experience to move deeper forward into familiarizing myself with harder and more complex algorithms and topics present in these libraries.
