Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Bautista, Jhon Hendricks
Section: CPE22S3

Performed on: 04/05/2025
Submitted on: 04/05/2025
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.

In [7]:
```
import random
import statistics
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 confirm your results match up to those that are obtained when using the statistics module (where possible):

- Mean
- Median
- Mode
- Sample variance

- Sample standard deviation

(hint: check out the Counter in the collections module of the standard library at https://docs.python.org/3/library/collections.html#collections.Counter

MEAN

In [10]:
```
# WITHOUT import statistics
mean = sum(salaries)/len(salaries)

print('Mean: ', mean)
```

```
Mean:  585690.0
```

In [11]:
```
# Checking result WITH import statistics
mean = statistics.mean(salaries)

print('Mean: ', mean)
```

```
Mean:  585690.0
```

MEDIAN

In [13]:
```
# WITHOUT import statistics
# using formula ((n/2) + (n/2)+1)/2)

num50 = salaries[int(len(salaries)/2) - 1] # get the middle of the list
num51 = salaries[int((len(salaries)/2) + 1) - 1] # get the item next from the middl
median = (num50+num51)/2

print('Median: ', median)
```

```
Median:  885500.0
```

In [14]:
```
# Checking WITH import statistics

median = statistics.median(salaries)

print('Median: ', median)
```

```
Median:  589000.0
```

MODE

In [16]:
```
# WITHOUT import statistics
import collections as c

mode = c.Counter(salaries).most_common(1)[0][0]
'''Using the counter we count the instance of the items in the salaries then
the most common will give the most common instance of item. the indexing of [0][0]
display only the number or the mode.'''

print('Mode: ', mode)
```

```
Mode:  477000.0
```

In [17]:
```python
# Checking WITH import statistics

mode = statistics.mode(salaries)

print('Mode: ', mode)
```

```
Mode:  477000.0
```

Sample variance

In [19]:
```python
# WITHOUT import statistics

def getMEAN(salaries): # getting the mean
    mean = sum(salaries)/len(salaries)
    return mean

def getSampleVariance(salaries):
    if len(salaries) < 2:  # error checking for not enough items
        return None

    mean = getMEAN(salaries) # computing the mean
    squared_diff = 0
    for x in salaries:
        squared_diff += (x - mean) ** 2 # getting summation of (x - mean)^2

    return squared_diff / (len(salaries) - 1) # the variance

variance = getSampleVariance(salaries)

print('Sample Variance: ', variance)
```

```
Sample Variance:  70664054444.44444
```

In [20]:
```python
# Checking WITH statistics import

variance = statistics.variance(salaries)

print('Variance: ', variance)
```

```
Variance:  70664054444.44444
```

Sample standard deviation

In [22]:
```python
# WITHOUT import statistics

def getMEAN(salaries): # getting the mean
    mean = sum(salaries)/len(salaries)
    return mean

def getSampleStandardDeviation(salaries):
    if len(salaries) < 2:  # error checking for not enough items
        return None

    mean = getMEAN(salaries) # computing the mean
```

```
    squared_diff = 0
    for x in salaries:
        squared_diff += (x - mean) ** 2 # getting summation of (x - mean)^2

    return (squared_diff / (len(salaries) - 1))**0.5 # square root of variance is s

sd = getSampleStandardDeviation(salaries)

print('Sample Standard Deviation: ', sd)
```

```
Sample Standard Deviation:  265827.11382484
```

In [23]:
```
# WITH statistics import

sd = statistics.stdev(salaries)

print('Sample Standard Deviation: ', sd)
```

```
Sample Standard Deviation:  265827.11382484
```

In [ ]:
```

```

Exercise 2

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

In [25]:
```
import random
import statistics
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

- Range
- Coefficient of variation
- Interquartile range
- Quartile coefficient of dispersion

Range

In [28]:
```
# formula: maxVAL - minVAL

range = max(salaries) - min(salaries)
print('Range: ', range)
```

```
Range:  995000.0
```

Coefficient of variation

In [30]:
```
# using formula (sd/mean)*100
```

```
sd = statistics.stdev(salaries)
mean = statistics.mean(salaries)
cv = (sd/mean)*100

print('Coefficient of Variation: ', cv)
```

```
Coefficient of Variation:  45.38699889443903
```

Interquartile range

In [32]:
```
# using formula: IQR = Q3 - Q1

q = statistics.quantiles(salaries, n =  4, method = 'inclusive') # we get the quart
# inclusive method disregard the 0% and 100% which not important for this matter
q1 = q[0]
q2 = q[1]
q3 = q[2]
iqr = q3 -q1

print('InterQuartile Range: ', iqr)
```

```
InterQuartile Range:  413250.0
```

In [33]:
```
# formula: (Q1 Q3)/(Q1+Q3)
q1, q2, q3 = statistics.quantiles (salaries, n = 4, method = 'inclusive')
qcd = (q3 - q1)/(q1 + q3)
print('Quartile coefficient of dispersion: ', qcd)
```

```
Quartile coefficient of dispersion:  0.338660110633067
```

In [ ]:
```

```

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 Diagnosis
- 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.

In [36]:
```
# 1. Identify the column names
import pandas as pd

df = pd.read_csv('diabetes.csv') # loading the csv file

df.columns # using columns method to show the columns
```

Out[36]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [37]:
```
# 2. Identify the data types of the data

df.dtypes # dtypes method is used to shows the datatype per column
```

Out[37]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [38]:
```
# 3. Display the total number of records

df.shape[0] # using shape method then only returning number of rows since it repres
```

Out[38]:
```
768
```

In [39]:
```
# 4. Display the first 20 records

df.head(20) # using the head method then passing '20' to show first 20 items
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
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	

In [40]:
```
# 5. Display the last 20 records

df.tail(20) # tail outputs the last rows, passing '20' will result in the last 20 r
```

Out[40]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

In [41]:

```
# 6. Change the Outcome column to Diagnosis

df = df.rename(columns = {'Outcome': 'Diagnosis'}) # using the rename method and th
df.head()
```

Out[41]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [42]:
```
# 7. Create a new column Classification that display "Diabetes" if the value of out

df['Classification'] = df['Diagnosis'].apply(lambda x: 'Diabetes' if x ==1 else 'No
'''Using the apply method and lambda function, we can use it to validate the value
new column accordingly'''

df.head() # checking modificaiton
```

Out[42]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [43]:
```
# 8. Create a new dataframe "withDiabetes" that gathers data with diabetes

withDiabetes = df[df['Classification'] == 'Diabetes'] # only getting the rows that

withDiabetes.head() # checking
```

Out[43]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
2	8	183	64	0	0	23.3	0
4	0	137	40	35	168	43.1	2
6	3	78	50	32	88	31.0	0
8	2	197	70	45	543	30.5	0

In [44]:
```
# 9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

withoutDiabetes = df[df['Classification'] == 'No Diabetes']
```

```
withoutDiabetes.head() # checking
```

Out[44]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
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	

In [45]:

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

Pedia = df[(df['Age'] > 0) & (df['Age'] < 19)] # creating the new dataframe with th

Pedia.head() # checking
```

Out[45]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

In [46]:

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

Adult = df[df['Age'] > 19] # creating new dataframe with condition checking of age

Adult.head() # checking
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [47]:

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

import numpy as np # importing numpy

# use average method then dictate which column to compute for
aveAGE = np.average(df['Age'])
aveGLUCOSE = np.average(df['Glucose'])

# printing results
print('Average Age: ', aveAGE)
print('Average Glucose: ', aveGLUCOSE)
```

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

In [48]:
```python
# 13. Use numpy to get the median age and glucose value.

# use median method then dictate which column to compute for
midAGE = np.median(df['Age'])
midGLUCOSE = np.median(df['Glucose'])

# printing results
print('Median Age: ', midAGE)
print('Median Glucose: ', midGLUCOSE)
```

```
Median Age:  29.0
Median Glucose:  117.0
```

In [49]:
```python
# 14. Use numpy to get the middle values of glucose and age.

# using quantile to find the middle value.
middle_glucose = np.quantile(df['Glucose'], 0.5)
middle_age = np.quantile(df['Age'], 0.5)

print('Middle Value for Glucose: ', middle_glucose)
print('Middle Value for Age: ', middle_age)
```

```
Middle Value for Glucose:  117.0
Middle Value for Age:  29.0
```

In [50]:
```python
# 15. Use numpy to get the standard deviation of the skinthickness.

sd = np.std(df['SkinThickness'])

print('Standard Deviation of Skin Thickness: ', sd)
```

```
Standard Deviation of Skin Thickness:  15.941828626496978
```

6.4 CONCLUSION

In activity I was able to learn about the different ways of analyzing data and the different tools needed for analysis. I used different python modules such as statistics, pandas, and numpy but before that I learned how to do analysis even without these python modules. I can say that the modules is really helpful in making the work faster. This is becuase I dont have to create my own functions just to compute for a certain statistic. The statistics module is really helpful in performing different statistical analysis and sytnax is very straightforward. As for the analysis of dataframes in python pandas, I can conclude that it is a very versatile module because of its vast list of methods for analysis of a data or file. In this activity I was able to use alot of those methods for accessing and modifying dataframes, creating dataframes and as well as doing statistical analysis in pair with the numpy module.

In [ ]:
```python

```