Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Fernandez, Don Eleazar
Section: CPE22S3

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

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 [122…

```
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 confirm 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

In [125…

```
# Write a comment per statistical function
```

In [127…

```
from statistics import mean, median, mode, variance, stdev
```

Mean

In [130…

```
# To find the mean with statistics module
mean = mean(salaries)
mean
```

Out[130…

```
585690.0
```

In [132…

```
# To find the mean without statistics module manually by dividing the summation of
mean_data = sum(salaries)/len(salaries)
mean_data
```

Out[132…

```
585690.0
```

Median

In [135…

```
# Sort first the list in order to find the correct middle number of the list
salaries = sorted(salaries)
```

In [137…

```
# To find the median with statistics module
median(salaries)
```

Out[137…

```
589000.0
```

In [139…

```
# To find the median without statistics module manually by counting first the numbe
# and have the result of it as the parameter for the list
median_number = len(salaries)
median_number = median_number/2
median_data = salaries[int(median_number)]
median_data
```

Out[139…

```
590000.0
```

Mode

In [142…

```
# To find the mode with statistics module
mode(salaries)
```

Out[142…

```
477000.0
```

In [144…

```
# To find the mode without statistics module, by having the collections module
from collections import Counter

counter = Counter(salaries) # This count the frequency of each number in the list
mode_data = counter.most_common(1)[0][0] # The ".most_common" picks out the highest
mode_data
```

Out[144…

```
477000.0
```

Sample Variance

In [147…

```
# To find the sample variance with statistics module
variance(salaries, mean)
```

Out[147…

```
70664054444.44444
```

In [149…

```
# To find the sample variance without statistics module, by having pandas
summ = sum(salaries) # This sums up the salaries
numm = len(salaries) # This counts up the number of salaries
mean_number = summ / numm # This compute for the mean of the salaries with the summ
variance_data = sum((x - mean_number) ** 2 for x in salaries) / (numm - 1) # The ac
variance_data
```

Out[149…

```
70664054444.44444
```

Sample Standard Deviation

In [152…

```
# To find the sample standard deviation with statistics module
stdev(salaries)
```

Out[152…

```
265827.11382484
```

In [154…

```
# To find the sample standard deviation without statistics module manually with the
summ = sum(salaries) # This sums up the salaries
numm = len(salaries) # This counts up the number of salaries
mean_number = summ / numm # This compute for the mean of the salaries with the summ
deviation_number = sum((x - mean_number) ** 2 for x in salaries) / (numm - 1) # The
deviation_data = deviation_number ** (1/2) # If you square rooted the sample varian
deviation_data
```

Out[154…

```
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

In [157…

```
# Write a comment per statistical function
```

In [159…

```
from statistics import mean, stdev, quantiles
```

Range

In [162…

```
# To find the range with statistics module
# The range is found by the maximum value subtracted by the minimum value

range_data = max(salaries) - min(salaries)
range_data
```

Out[162…

```
995000.0
```

Coefficient of variation Interquartile range

In [165…

```
# To find the coefficient of variation Interquartile range with statistics module
# The coefficient variation is found by the standard deviation divided by the mean,
# The interquartile range is found by the third quartile subtracted by the first qu

co_variation = (stdev(salaries) / mean(salaries)) * 100
q1, q2, q3 = quantiles(salaries, n = 4)
q_range = q3 - q1
print(f"The Coefficient Variation is {co_variation}")
print(f"The Interquartile Range is {q_range}")
```

```
The Coefficient Variation is 45.38699889443903
The Interquartile Range is 421750.0
```

Quartile coefficient of dispersion

In [168…

```
# To find the quartile coefficient of dispersion with statistics module
# The quartile coefficient of dispersion is found by the third quartile subtracted
# divided by the result of the summation of the third quartile and the first quarti

q1, q2, q3 = quantiles(salaries, n = 4)
qc_dispersion = (q3 - q1) / (q3 + q1)
qc_dispersion
```

Out[168…

```
0.34491923941934166
```

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 display "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" thats 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.

In [171…

```
# Indicate which item you're answering with a comment
```

In [173…

```
import pandas as pd

data = pd.read_csv("diabetes.csv")
data = pd.DataFrame(data)
```

In [175…

```
# 1. Identify the column names
data.columns
```

Out[175…

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

In [177…

```
# 2. Identify the data types of the data
data.dtypes
```

Out[177…

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

In [179…

```
# 3. Display the total number of records
data.shape[0]
```

Out[179…

```
768
```

In [181…

```
# 4. Display the first 20 records
data.head(20)
```

Out[181…

	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 [183…

```
# 5. Display the last 20 records
data.tail(20)
```

Out[183…

	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 [185…

```
# 6. Change the Outcome column to Diagnosis
data = data.rename(columns = {"Outcome": "Diagnosis"})
data
```

Out[185…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

In [187…

```
# 7. Create a new column Classification that display "Diabetes" if the value of out
data["Classification"] = (data["Diagnosis"] == 1).map({True: "Diabetes", False: "No
data
```

Out[187…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

In [189…

```python
# 8. Create a new dataframe "withDiabetes" that gathers data with diabetes
withDiabetes = data[data["Diagnosis"] == 1]
withDiabetes = pd.DataFrame(withDiabetes)
withDiabetes
```

Out[189…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

In [191…

```python
# 9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = data[data["Diagnosis"] == 0]
noDiabetes = pd.DataFrame(noDiabetes)
noDiabetes
```

Out[191…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

In [193…

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

Out[193…

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

In [195…

```
# 11. Create a new dataframe "Adult" that gathers data with age greater than 19
Adult = data[data["Age"] >= 19]
Adult = pd.DataFrame(Adult)
Adult
```

Out[195…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

In [197…

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

ave_age = np.average(data["Age"])
ave_gluc = np.average(data["Glucose"])
print(f"The average Age is {ave_age}")
print(f"The average Glucose is {ave_gluc}")
```

```
The average Age is 33.240885416666664
The average Glucose is 120.89453125
```

In [199…

```
# 13. Use numpy to get the median age and glucose value.
import numpy as np

sort_age = np.sort(data["Age"])
sort_gluc = np.sort(data["Glucose"])
med_age = np.median(sort_age)
med_gluc = np.median(sort_gluc)
print(f"The median value of Age is {med_age}")
print(f"The median value of Glucose is {med_gluc}")
```

```
The median value of Age is 29.0
The median value of Glucose is 117.0
```

In [201…

```
# 14. Use numpy to get the middle values of glucose and age.
import numpy as np

sort_age = np.sort(data["Age"])
sort_gluc = np.sort(data["Glucose"])
mid_age = np.median(sort_age)
mid_gluc = np.median(sort_gluc)
```

```
print(f"The middle value of Age is {mid_age}")
print(f"The middle value of Glucose is {mid_gluc}")
```

```
The middle value of Age is 29.0
The middle value of Glucose is 117.0
```

In [203…

```
# 15. Use numpy to get the standard deviation of the skinthickness.
import numpy as np

std_dev = np.std(data["SkinThickness"])
std_dev
```

Out[203…

```
15.941828626496978
```

6.4 Conclusion

To conclude, the laboratory activity demonstrated the usage of statistical tools through python, which it can either be through different modules, such as collections and statistics. It also served as a refresher of the concepts discussed during the lecture. Moreover, I learned how to use the statistics module, which will be helpful in my future studies, especially in data analysis.

In [ ]:

```
```