Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python Name: Sanchez, Christan Ray R. 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 [31]:
```
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 [32]:
```
from statistics import mean, median, mode, variance, stdev #this is to importing a
```

MEAN

In [33]:
```
#To get the mean or average
mean = mean(salaries)
```

```
mean
```

Out[33]: 585690.0

In [34]:
```
#To find the mean without the statistical module to calculate the mean salary by di

mean_wt = sum(salaries)/len(salaries)
mean_wt
```

Out[34]: 585690.0

Median

In [37]:
```
#in order to get the median we have to first sort out the data

salaries = sorted(salaries)
median(salaries)
```

Out[37]: 589000.0

In [38]:
```
# In order to get the median without the statistical module, we can first count the
# Then we divide that number by 2 to find the middle index, and use that index to g

# First, count how many items are in the salaries list
median_number = len(salaries)

# Then, divide that number by 2 to find the middle
median_number = median_number / 2

# Get the value at the middle position (rounded down)
median_data = salaries[int(median_number)]

# Show the median value
median_data
```

Out[38]: 590000.0

Mode

In [39]:
```
#Finding the mode with statistics module
mode(salaries)
```

Out[39]: 477000.0

In [40]:
```
# To find the mode without using the statistics module, we can use the collections
from collections import Counter

# This counts how many times each number appears in the list
counter = Counter(salaries)

# This picks the value that appears the most using .most_common
mode_data = counter.most_common(1)[0][0]
```

```
# Show the result
mode_data
```

Out[40]: 477000.0

Sample Variance

In [41]:
```
# To find the sample variance with statistics module
variance(salaries, mean)
```

Out[41]: 70664054444.44444

In [42]:
```
# To find the sample variance without using the statistics module, we can calculate

summ = sum(salaries)  # Get the total sum of all salaries
numm = len(salaries)  # Get the number of salary entries
mean_number = summ / numm  # Calculate the mean (average) of the salaries
variance_data = sum((x - mean_number) ** 2 for x in salaries) / (numm - 1)  # Compu

variance_data
```

Out[42]: 70664054444.44444

Sample Standard Deviation

In [43]:
```
# To find the sample standard deviation with statistics module
stdev(salaries)
```

Out[43]: 265827.11382484

In [44]:
```
# 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[44]: 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 [46]:
```
# Write a comment per statistical function
```

In [47]:
```
from statistics import mean, stdev, quantiles
```

Range

In [49]:
```
# To find the range with statistics module
# we subtract the minimum value from the maximum value
range_data = max(salaries) - min(salaries)
range_data
```

Out[49]:
```
995000.0
```

Coefficient of variation Interquartile range

In [50]:
```
# To find the coefficient of variation and the interquartile range using the statis

# The coefficient of variation is calculated by dividing the standard deviation by
co_variation = (stdev(salaries) / mean(salaries)) * 100

# Calculate the first, second (median), and third quartiles
q1, q2, q3 = quantiles(salaries, n = 4)

# The interquartile range is the difference between the third quartile and the firs
q_range = q3 - q1

# Print the results
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 [51]:
```
# To find the quartile coefficient of dispersion using the statistics module

# Calculate the first, second (median), and third quartiles
q1, q2, q3 = quantiles(salaries, n = 4)

# The quartile coefficient of dispersion is the difference between the third and fi
# divided by the sum of the third and first quartiles
qc_dispersion = (q3 - q1) / (q3 + q1)

# Display the result
qc_dispersion
```

Out[51]:
```
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 [52]:
```
import pandas as pd

diabetes = pd.read_csv('diabetes.csv')
diabetes.head(3)
```

Out[52]:

	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

In [53]:
```
#1. Identify the column names
diabetes.columns
```

Out[53]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [54]:
```
#2. Identify the data types of the data
diabetes.dtypes
```

Out[54]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [55]:
```
#3. Display the total number of records
diabetes.shape[0]
```

Out[55]: 768

In [56]:
```
#4. Display the first 20 records
diabetes.head(20)
```

Out[56]:

	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 [58]:
```
#5. Display the last 20 records
diabetes.tail(20)
```

Out[58]:

	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 [60]:

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

diabetes = diabetes.rename(columns = {'Outcome': 'Diagnosis'})
diabetes
```

Out[60]:

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

data['Classification'] = (data['Diagnosis'] == 1).map
```

In [ ]:
```
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 outco
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 [61]:
```
# 7. Create a new column Classification that display "Diabetes" if the value of out
diabetes["Classification"] = (diabetes["Diagnosis"] == 1).map({True: "Diabetes", Fa
diabetes
```

Out[61]:

	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 [65]:
```
#8. Create a new dataframe "withDiabetes" that gathers data with diabetes

withDiabetes = diabetes[diabetes['Diagnosis'] == 1]
withDiabetes = pd.DataFrame(withDiabetes)
withDiabetes
```

Out[65]:

	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 [67]:
```
# 9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = diabetes[diabetes["Diagnosis"] == 0]
noDiabetes = pd.DataFrame(noDiabetes)
noDiabetes
```

Out[67]:

	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 [70]:
```
#10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
Pedia = diabetes[(diabetes["Age"] >= 0) & (diabetes["Age"] <= 19)]
Pedia = pd.DataFrame(Pedia)
Pedia
```

Out[70]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

In [72]:
```
# 11. Create a new dataframe "Adult" that gathers data with age greater than 19
Adult = diabetes[diabetes["Age"] >= 19]
Adult = pd.DataFrame(Adult)
Adult
```

Out[72]:

	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 [84]:
```
# 12. Use numpy to get the average age and glucose value.
import numpy as np

ave_age = np.average(diabetes["Age"])
ave_gluc = np.average(diabetes["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 [85]:
```
# 13. Use numpy to get the median age and glucose value.
sort_age = np.sort(diabetes["Age"])
sort_gluc = np.sort(diabetes["Glucose"])
med_age = np.median(sort_age)
med_gluc = np.median(sort_gluc)
```

```
print(f"Median[Age]: {med_age}")
print(f"Median[Glucose]: {med_gluc}")
```

```
Median[Age]: 29.0
Median[Glucose]: 117.0
```

In [86]:
```
# 14. Use numpy to get the middle values of glucose and age.
sort_age = np.sort(diabetes["Age"])
sort_gluc = np.sort(diabetes["Glucose"])
mid_age = np.median(sort_age)
mid_gluc = np.median(sort_gluc)
print(f"Middle[Age]: {mid_age}")
print(f"Middle[Glucose]: {mid_gluc}")
```

```
Middle[Age]: 29.0
Middle[Glucose]: 117.0
```

In [88]:
```
# 15. Use numpy to get the standard deviation of the skinthickness.

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

Out[88]:
```
15.941828626496978
```

6.4 Conclusion

In conclusion, this laboratory activity provided valuable experience in using statistical tools through Python, specifically the statistics module and numpy. It also served as a helpful refresher on the concepts we discussed during the lecture. By applying the statistics module and numpy, I learned how to perform various statistical calculations, which will be beneficial in my future studies, especially in data analysis.

Overall, I realized the importance of consistent practice in coding to improve my ability to recall and apply these commands more easily in the future.