Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Detchosa, Ralph Christian D.

Section: CPE22S3

Performed on: 03/07/2024

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.

In [28]:

```
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() 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)
- Sample Variance
- Sample standard deviation

Without importing libraries

In [111]:

```
# Mean
def mean(salaries):
  mean = sum(salaries)/len(salaries)
  return mean

Mean = mean(salaries)
print('Mean is: ', Mean)
```

```
Mean is:  585690.0
```

In [109]:

```
# Median
def median(salaries):

  salaries.sort()
  n = len(salaries)

  if n % 2 == 0:
    median1 = salaries[n//2]
    median2 = salaries[n//2 - 1]
    median = (median1 + median2)/2
  else:
    median = salaries[n//2]
  return median

Median = median(salaries)
print('The median is', Median)
```

```
The median is 589000.0
```

In [119]:

```
# Mode
def mode(salaries):
    counts = {}
    for salary in salaries:
        counts[salary] = counts.get(salary, 0) + 1

    max_count = max(counts.values())
    modes = [salary for salary, count in counts.items() if count == max_count]
    return modes

Mode = mode(salaries)
print('The mode is', Mode[0])
```

```
The mode is 477000.0
```

In [114]:

```
# Sample Variance
def variance(salaries):
  mean = sum(salaries) / len(salaries)
  variance = sum((i - mean) ** 2 for i in salaries) / len(salaries)
  return variance

Variance = variance(salaries)
print('The sample variance is', Variance)
```

```
The sample variance is 69957413900.0
```

In [116]:

```
# Sample Standard Deviation
def standard_deviation(salaries):
  mean = sum(salaries) / len(salaries)
  variance = sum((i - mean) ** 2 for i in salaries) / len(salaries)
  standard_dev = variance ** 0.5
  return standard_dev

Standard_Deviation = standard_deviation(salaries)
print('The standard deviation is', Standard_Deviation )
```

```
The standard deviation is 264494.6386980273
```

To check if it is correct using libraries

In [98]:

```
np.mean(salaries)
```

Out[98]:

```
585690.0
```

In [99]:

```
np.median(salaries)
```

Out[99]:

```
589000.0
```

In [126]:

```
from scipy.stats import mode
mode(salaries)
```

Out[126]:

```
ModeResult(mode=477000.0, count=3)
```

In [106]:

```
np.var(salaries)
```

Out[106]:

```
69957413900.0
```

In [107]:

```
np.std(salaries)
```

Out[107]:

```
264494.6386980273
```

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 [128]:

```
import statistics
# Range
def range(salaries):
  range = max(salaries) - min(salaries)
  return range

Range = range(salaries)
print('The range of salary is', Range)
```

```
The range of salary is 995000.0
```

In [130]:

```
# Coefficient of variation Interquartile Range
def cv(salaries):
  cv = np.std(salaries, ddof=1) / np.mean(salaries) * 100
  return cv

CVIR = cv(salaries)
print('The Coefficient of Variation Interquartile Range is', CVIR)
```

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

In [131]:

```
# Quartile Coefficient of Dispersion
def qcd(salaries):
  q1 = np.percentile(salaries, 25)
  q3 = np.percentile(salaries, 75)
  qcd = (q3 - q1) / (q3 + q1)
  return qcd

QCD = qcd(salaries)
print('The Quartile Coefficient of Dispersion of salaries is', QCD)
```

```
The Quartile Coefficient of Dispersion of salaries is 0.338660110633067
```

Exercise 3: Pandas for Data Analysis

Load the diabetes.csv file. Convert the diabetes.csv into data frame.

In [2]:

```
import numpy as np
import pandas as pd

file_path = '/content/diabetes.csv'

diabetes_data = pd.read_csv(file_path)

diabetes_data
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

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" 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 "Pedia" 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 [8]:

```
# 1.)
column_names = diabetes_data.columns
column_names
```

Out[8]:

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

In [9]:

```
# 2.)
data_types = diabetes_data.dtypes
data_types
```

Out[9]:

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

In [12]:

```
# 3.)
total_num = len(diabetes_data)
print('The total number of recors is', total_num)
```

```
The total number of recors is 768
```

In [15]:

```
# 4.)
diabetes_data[:20]
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1

In [17]:

```
# 5.)
diabetes_data[-20:]
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
748	3	187	70	22	200	36.4	0.408	36	1
749	6	162	62	0	0	24.3	0.178	50	1
750	4	136	70	0	0	31.2	1.182	22	1
751	1	121	78	39	74	39.0	0.261	28	0
752	3	108	62	24	0	26.0	0.223	25	0
753	0	181	88	44	510	43.3	0.222	26	1
754	8	154	78	32	0	32.4	0.443	45	1
755	1	128	88	39	110	36.5	1.057	37	1
756	7	137	90	41	0	32.0	0.391	39	0
757	0	123	72	0	0	36.3	0.258	52	1
758	1	106	76	0	0	37.5	0.197	26	0
759	6	190	92	0	0	35.5	0.278	66	1
760	2	88	58	26	16	28.4	0.766	22	0
761	9	170	74	31	0	44.0	0.403	43	1
762	9	89	62	0	0	22.5	0.142	33	0
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

In [20]:

```
# 6.)
```

```
diabetes_data.rename(columns = {'Outcome':'Diagnosis'}, inplace = True)
diabetes_data
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [48]:

```
# 7.)
def check_outcome(row):
  if row['Diagnosis'] == 1:
    return 'Diabetes'
  elif row['Diagnosis'] == 0:
    return 'No Diabetes'

diabetes_data['Classification'] = diabetes_data.apply(check_outcome, axis=1)
diabetes_data
```

Out[48]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
...	...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

768 rows × 10 columns

In [54]:

```
# 8.)
withDiabetes = diabetes_data[diabetes_data['Diagnosis'] == 1].copy()
withDiabetes
```

Out[54]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
6	3	78	50	32	88	31.0	0.248	26	1	Diabe
8	2	197	70	45	543	30.5	0.158	53	1	Diabe
...	...	...	...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	1.057	37	1	Diabe
757	0	123	72	0	0	36.3	0.258	52	1	Diabe
759	6	190	92	0	0	35.5	0.278	66	1	Diabe
761	9	170	74	31	0	44.0	0.403	43	1	Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe

268 rows × 10 columns

In [55]:

```
# 9.)
noDiabetes = diabetes_data[diabetes_data['Diagnosis'] == 0].copy()
noDiabetes
```

Out[55]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
5	5	116	74	0	0	25.6	0.201	30	0	No Diabe
7	10	115	0	0	0	35.3	0.134	29	0	No Diabe
10	4	110	92	0	0	37.6	0.191	30	0	No Diabe
...	...	...	...	...	...	...	...	...	...	
762	9	89	62	0	0	22.5	0.142	33	0	No Diabe
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

500 rows × 10 columns

In [57]:

```
# 10.)
Pedia = diabetes_data[diabetes_data['Age'].between(0, 9)].copy()
Pedia
```

Out[57]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

In [59]:

```
# 11.)
```

```
Adult = diabetes_data[diabetes_data['Age'] > 9].copy()
Adult
```

Out[59]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
...	...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

768 rows × 10 columns

In [79]:

```
# 12.)
print('The average of age is', np.mean(diabetes_data['Age']), 'and the average of glucose
is', np.mean(diabetes_data['Glucose']))
```

```
The average of age is 33.240885416666664 and the average of glucose is 120.89453125
```

In [80]:

```
# 13.)
print('The average of age is', np.median(diabetes_data['Age']), 'and the average of gluco
se is', np.median(diabetes_data['Glucose']))
```

```
The average of age is 29.0 and the average of glucose is 117.0
```

In [93]:

```
# 14.)
print('The middle number of glucose is',np.median(np.sort(diabetes_data['Glucose'])), 'am
d the middle number of age is', np.median(np.sort(diabetes_data['Age'])))
```

```
The middle number of glucose is 117.0 amd the middle number of age is 29.0
```

In [82]:

```
# 15.)
print('The average of age is', np.std(diabetes_data['SkinThickness']))
```

```
The average of age is 15.941828626496939
```

6.4 Conclusion

To conlude this Hands-on Activity, imported libraries such as pandas and numpy are very important in solving data analysis tasks for there are functions that would instantly solve for a certain values compared to the other approach where you will manually code the solution. Another benefits to it is that it is just very simple and easy to understand functions to call and the guides for the parameters are accesible publicly online

In [ ]: