Hands - On - Activity 6.1 Introduction to Data Analysis and Tools

- **Name:** Calingo, Christian Lei S.
- **Section:** CPE22S3
- **Performed on:** 03 - 06 - 2024
- **Submitted on:** 03 - 06 - 2024
- **Submitted to:** Engr. Roman 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

```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
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,
 756000.0,
 618000.0,
 251000.0,
 910000.0,
 983000.0,
 810000.0,
 902000.0,
 310000.0,
 730000.0,
 899000.0,
 684000.0,
 472000.0,
 101000.0,
 434000.0,
 611000.0,
 913000.0,
 967000.0,
 477000.0,
 865000.0,
 260000.0,
 805000.0,
 549000.0,
 14000.0,
 720000.0,
 399000.0,
 825000.0,
 668000.0,
 1000.0,
 494000.0,
 868000.0,
 244000.0,
 325000.0,
 870000.0,
 191000.0,
 568000.0,
 239000.0,
 968000.0,
 803000.0,
 448000.0,
 80000.0,
 320000.0,
 508000.0,
 933000.0,
 109000.0,
 551000.0
```

MEAN

```
length = len(salaries) # the length or the total number of data in the data set
sum = 0
for x in salaries: #summation of the data
  sum +=x

mean = sum/length #formula for the mean

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

```
    Mean:  585690.0
```

MEDIAN

```
salaries.sort()# sorting the list
midpoint = int((length/2)) # formula for midpoint

if midpoint % 2 == 1: #checking if even or odd
  print("Median: ", salaries[midpoint+1])
else:
  pass
  print("Median", (salaries[midpoint-1]+salaries[midpoint])/2)
```

```
    Median 589000.0
```

MODE

```
salaries.sort()
temp=[]

i=0
while i<len(salaries):
  temp.append(salaries.count(salaries[i]))
  i+=1

dic = dict(zip(salaries, temp))
mode = {k for (k,v)in dic.items() if v== max(temp)} #checking for the most abundant value

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

```
    Mode: {477000.0}
```

SAMPLE VARIANCE

```
summation = 0
for x in salaries:
  summation += (x - mean)**2 #formula for summation

samp_var = summation/(length-1) #formula for samp_var

print(samp_var)
```

```
70664054444.44444
```

SAMPLE STANDARD DEVIATION

```
standard_dev = (samp_var)**0.5 #square rooting the samp_var for std
print("Standard Deviation:",standard_dev)
```

```
Standard Deviation: 265827.11382484
```

EXERCISE 2

MEAN

```
from statistics import mean #importing the mean function

def mean_cal(data):
  mean_ans = mean(data) #getting the mean of the data

  return mean_ans

print("Mean:",mean_cal(salaries))
```

```
Mean: 585690.0
```

MEDIAN

```
from statistics import median #importing the median function

def median_cal(data):
  median_ans = median(data) #getting the median of the data

  return median_ans

print("Median:",median_cal(salaries))
```

```
Median: 589000.0
```

MODE

```
from statistics import mode #importing the modefunction

def mode_cal(data):
  mode_ans = mode(data) # getting the mode of the data

  return mode_ans

print("Mode:",mode_cal(salaries))
```

```
Mode: 477000.0
```

SAMPLE VARIANCE

```
from statistics import variance # importing the variance function

def samp_var(data):
  samp_var2 = variance(data) # getting the variance of the data

  return samp_var2

print("Sample Variance:",samp_var(salaries))
```

```
Sample Variance: 70664054444.44444
```

STANDARD DEVIATION

```
from statistics import stdev #importing the std function

def standard_dev_cal(data):
  st_dev = stdev(data) #getting the standard variation of the data

  return st_dev

print("Standard Deviation:",standard_dev_cal(salaries))
```

```
Standard Deviation: 265827.11382484
```

RANGE

```
def range_cal(data):
  range = max(data)-min(data) #getting the max and min value of the data and subtracting it
  return range

print("Range:",range_cal(salaries))
```

```
Range: 995000.0
```

Coefficient of Variation Interquartile range

```
from statistics import stdev
from statistics import mean

def coef_var_function(data):
  std = stdev(data)
  mean_res = mean(data)
  coef_var = (std/mean_res) # formula for the coefficient variation
  return coef_var

def quartile_range_function(size):
  low_q = (size+1)*(0.25) #formula for Q1
  high_q = (size+1)*(0.75) # formula for Q3
  iqr = high_q - low_q #IQR formula
  return iqr

print("Coefficient of Variation:",coef_var_function(salaries))
print("Interquartile Range:",quartile_range_function(length))
```

```
Coefficient of Variation: 0.45386998894439035
Interquartile Range: 50.5
```

```
dispersion = (max(salaries) - min(salaries))/max(salaries) + min(salaries) #dispersion formula
print("Coefficient of Dispersion:",dispersion)
```

```
Coefficient of Dispersion: 1000.9989959839357
```

EXERCISE 3

```
filepath = '/content/diabetes (1).csv' #importing the diabetes csv
import pandas as pd
import numpy as np
```

```
data = pd.read_csv(filepath) #passing the data of the csv in data variable
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

1. Identify the Column Names

```
data.columns
```

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

```
col_num = 1
for x in data.columns:
  print(f"{col_num}.",x)
  col_num+=1
```

```
1. Pregnancies
2. Glucose
3. BloodPressure
4. SkinThickness
5. Insulin
6. BMI
7. DiabetesPedigreeFunction
8. Age
9. Outcome
```

2.Identify the data types of the data

```
data.dtypes
```

```
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

```
print("Number of records:",len(data))
```

```
Number of records: 768
```

4.Display the first 20 records

```
data.head(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
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	

5.Display the last 20 records

```
data.tail(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

6.Change the Outcome column to Diagnosis

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

7.Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

```
data['Classification'] = np.where(data['Diagnosis'] == 1, 'Diabetes', 'No Diabetes') #creating
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 × 10 columns

8.Create a new dataframe "withDiabetes" that gathers data with diabetes

```
diabetes_dataframe6 = pd.DataFrame(data) #creating a dataframe
withDiabetes = diabetes_dataframe6[diabetes_dataframe6['Diagnosis'] == 1].copy() #accessing the diagnosis column and checking if the value i
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

9.Create a new dataframe "noDiabetes" thats gathers data with no diabetes

```
diabetes_dataframe7 = pd.DataFrame(data) #creating a dataframe
noDiabetes = diabetes_dataframe7[diabetes_dataframe7['Diagnosis'] == 0].copy() #accessing the diagnosis column and checking if the value is
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

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

```
diabetes_dataframe8 = pd.DataFrame(data) #creating a dataframe
Pedia = diabetes_dataframe8[diabetes_dataframe8['Age'] <= 19].copy() #operator for age lessthan 19
Pedia
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc

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

```
diabetes_dataframe9 = pd.DataFrame(data) #creating a dataframe
Adult = diabetes_dataframe9[diabetes_dataframe9['Age'] > 19].copy() #operator for age greater than 19
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

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

```
nump_mean = np.mean(data['Age']),np.mean(data['Glucose']) #accessing the age and glucose column and getting their average
for x in nump_mean:
  print(x)
```

```
33.240885416666664
120.89453125
```

13.Use numpy to get the median age and glucose value.

```
nump_median = np.median(data['Age']), np.median(data['Glucose']) #accessing the age and glucose column and getting their median
for x in nump_median:
  print(x)
```