Hands-on Activity 6.1 Introduction to Data Analysis and Tools

Name: John Rome A. Belocora

Section: CPE22S3

Performed on: 03/07/2024

Submitted on: 03/07/2024

Submitted to: Engr. Roman M. Richard

```
filepath = '/content/diabetes.csv'
```

```
import pandas as pd
import numpy as np
```

```
data = pd.read_csv(filepath)
```

```
data
```

	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

Exercise 1

Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules

```
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

Mean

```
# Sum the salaries
total = sum(salaries)
# divide the sum with the number of data list
mean = total/len(salaries)


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

```
Mean: 585690.0
```

Median

```
def Median(salaries):
  sort = sorted(salaries)
  length = len(salaries)
  # If the data list is even, then this condition will be satisfied
  if length % 2 == 0:
    first_middle = sort[length // 2 - 1]
    second_middle = sort[length // 2]
    median = (first_middle + second_middle) / 2
  # If the data list is odd, this condition will be satisfies
  else:
    median = sort[length // 2]

  return median

JomeyTest = Median(salaries)
print('Median:',JomeyTest)
```

```
Median: 589000.0
```

Mode

```
def Mode(salaries):
    # Empty list of Dictionary
    rep = {}
    # A for loop that iterates over each element in the salaries list
    for i in salaries:
        rep[i] = rep.get(i, 0) + 1

    modes = [key for key, val in rep.items() if val == max(rep.values())]

    if len(modes) == len(set(salaries)):
        print("No unique mode")
    else:
        print("Mode:", modes)

Mode(salaries)
```

```
Mode: [477000.0]
```

Sample Variance

```
def Sample_variance(salaries):
    # Calculate the sum of squared differences from the mean
    sum_squared_diff = sum((x - mean) ** 2 for x in salaries)

    # Calculate the sample variance
    sample_variance = sum_squared_diff / (length - 1)

    print("Sample Variance:", sample_variance)

Sample_variance(salaries)
```

```
Sample Variance: 70664054444.44444
```

Sample Standard Deviation

```
def Sample_std_dev(salaries):
    sum_squared_diff_2 = sum((x - mean) ** 2 for x in salaries)
    sample_variance_2 = sum_squared_diff_2 / (length - 1)

    # Calculate the sample standard deviation (square root of the sample variance)
    std_dev = sample_variance_2 ** 0.5

    print(std_dev)

Sample_std_dev(salaries)
```

```
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

Range

```
def Get_range(salaries):
  #Calculates the range by subtracting the highest value in salaries with the lowest value
  salary_range = max(salaries) - min(salaries)

  print('Range:',salary_range)

Get_range(salaries)
```

```
Range: 995000.0
```

Coefficient of variation Interquartile range

```
from statistics import *

def CVIQR(salaries):
  total = sum(salaries)
  mean = total/len(salaries)
  # CV = standard dev divided by mean
  standard_dev = stdev(salaries)
  CV = (standard_dev/mean)
  # Calculate quartiles
  q1 = median(salaries[:len(salaries) // 2])
  q3 = median(salaries[len(salaries) // 2:])

  # Handle equal quartile case for IQR
  iqr = None
  if q1 != q3:
    iqr = q3 - q1

print("Coefficient of Variation (CV):", cv)
print("Interquartile Range (IQR):", iqr)
CVIQR(salaries)
```

```
Coefficient of Variation (CV): 0.45386998894439035
Interquartile Range (IQR): 17500.0
```

Quartile coefficient of dispersion

```
def calculate_qcd(sal):
  # Calculate quartiles
  q1 = median(sal[:len(sal) // 2])
  q3 = median(sal[len(sal) // 2:])

  qcd = (q3 - q1) / (2 * median(salaries))

  return qcd
# Calculate QCD
qcd = calculate_qcd(salaries)

print("QCD:", qcd)
```

```
QCD: 0.014855687606112054
```

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.

```
filepath = '/content/diabetes.csv'
import pandas as pd
import numpy as np

data = pd.read_csv(filepath)
data
```

	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

```
#1
data.columns
```

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

```
#2
data.dtypes
```

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

```
#3
len(data)
```

```
768
```

```
#4
data[:20]
```

	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
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

```
#5
data.tail(20)
```

	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

```
#6
data.rename(columns = {'Outcome':'Diagnosis'}, inplace = True)
data
```

	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

```
#7
data['Classification'] = np.where(data['Diagnosis'] == 1, 'Diabetes', 'No Diabetes')

data
```

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

768 rows × 10 columns

```
#8
df = pd.DataFrame(data)
withDiabetes = df[df['Diagnosis'] == 1].copy()

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
df = pd.DataFrame(data)
noDiabetes = df[df['Diagnosis'] == 0].copy()

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
Pedia = df[df['Age'] <= 19].copy()
Pedia
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc

```
#11
Adult = df[df['Age'] >= 19].copy()
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	

```
#12
numpy_mean = np.mean(data['Age']), np.mean(data['Glucose'])
numpy_mean
```