Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Borja, Angelo Louis C.
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.

```
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

```
# Calculating the mean without using a library
def mean(numbers):
  total, mean_val = 0, 0
  length = len(numbers)
  for number in numbers:
    total += number
  mean_val = total/length
  return mean_val
print("Mean no library:", mean(salaries))

# Using a library
import statistics as stats
print("Mean using a library:", stats.mean(salaries))
```

```
Mean no library: 585690.0
Mean using a library: 585690.0
```

- Median

```
# Calculating the median without using a library
def median(numbers):
  length = len(numbers)
  sorted_nums = sorted(numbers)
  mid_num = (sorted_nums[(length//2 -1)] + sorted_nums[(length//2)])/2
  return mid_num
print("Median no library:",median(salaries))

# Using a library
print("Median using a library:", stats.median(salaries))
```

```
Median no library: 589000.0
Median using a library: 589000.0
```

- 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)

```
# Calculating the mode without using a library
from collections import Counter
def mode(numbers):
  counter = Counter(numbers)
  highest = 0
  for key in counter:
    if counter[key]>highest:
      highest = counter[key]
      high_key = key
  return(high_key)
print("Mode no library:", mode(salaries))

# Using a library
print("Mode using a library:", stats.mode(salaries))
```

```
Mode no library: 477000.0
Mode using a library: 477000.0
```

- Sample variance

```
# Calculating the sample variance without using a library
def sam_var(numbers):
  mean_val = mean(numbers)
  length = len(numbers)
  var_val = 0
  for num in numbers:
    var_val += (num - mean_val)**2
  var_val /=length-1
  return var_val
print("Sample variance no library:", sam_var(salaries))

# Using a library
print("Sample variance using a library:", stats.variance(salaries))
```

```
Sample variance no library: 70664054444.44444
Sample variance using a library: 70664054444.44444
```

- Sample standard deviation

```
# Calculating the sample standard deviation without using a library
import math
def sta_dev(numbers):
  mean_val = mean(numbers)
  dev_val = 0
  length = len(numbers)
  for num in numbers:
    dev_val += (num-mean_val)**2
  sam_sta_dev = math.sqrt(dev_val/(length-1))
  return sam_sta_dev

print("Sample standard deviation no library:", sta_dev(salaries))

# Using a library
print("Sample standard deviation using a library:", stats.stdev(salaries))
```

```
Sample standard deviation no library: 265827.11382484
Sample standard deviation using a library: 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
sorted_list = sorted(salaries)
print("Range of list salaries:", sorted_list[-1] - sorted_list[0])

#Coefficient of variation
print("Coefficient of variation of salaries:", stats.stdev(salaries)/stats.mean(salaries)* 100)

#Interquartile range
Q1 = stats.median(sorted_list[0:(len(sorted_list)//2)])
Q3 = stats.median(sorted_list[(len(sorted_list)//2):])
print("Interquartile range of salaries is:", Q3-Q1)

#Quartile coefficient of dispersion
print("Quartile coefficient of dispersion of salaries is:", (Q3 - Q1) / (Q3 + Q1))
```

```
Range of list salaries: 995000.0
Coefficient of variation of salaries: 45.38699889443903
Interquartile range of salaries is: 417500.0
Quartile coefficient of dispersion of salaries is: 0.3417928776094965
```

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.

```
# Loading the csv file from google drive
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/diabetes.csv')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
#1.identify the column names
df.columns
```

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

```
#2.Identify the data types of the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
```

```
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
#3.Display the total number of records
print(df.shape[0])
```

```
768
```

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

	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	
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	

Next steps: View recommended plots

```
#5.Display the last 20 records
df.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
df.rename(columns={'Outcome':'Diagnosis'}, inplace = True)
df.head(3)
```

	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	

Next steps: View recommended plots

```
#7.Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No
df['Classification'] = np.where(df['Diagnosis'] == 1, 'Diabetes', 'No Diabetes')
df.head(3)
```

	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	

Next steps: View recommended plots

```
#8.Create a new dataframe "withDiabetes" that gathers data with diabetes
withDiabetes = df[df['Classification'] == 'Diabetes'].copy()
withDiabetes.head(3)
```

	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	

Next steps: View recommended plots

```
#9.Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = df[df['Classification'] == 'No Diabetes'].copy()
noDiabetes.head(3)
```

	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	

Next steps: View recommended plots

```
#10.Create a new dataframe "Pedia" that gathers data with age 0 to 19
Pedia = df[(df['Age'] >= 0) & (df['Age'] <=19)].copy()
Pedia.head(3)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc

```
#11.Create a new dataframe "Adult" that gathers data with age greater than 19
Adult = df[df['Age'] > 19].copy()
Adult.head(3)
```

	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	

Next steps: View recommended plots

```
#12.Use numpy to get the average age and glucose value.
print("Average age:", np.mean(df['Age']))
print("Average glucose value:", np.mean(df['Glucose']))
```

```
Average age: 33.240885416666664
```