Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Corpuz, Micki Laurren B.

Section: CPE22S3

Performed on: 04/05/2024

Submitted on: 04/05/2024

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcomes

- Use pandas and numpy data analysis tools.
- Demonstrate how to analyze data using numpy and pandas

6.2 Resources:

- Personal Computer
- Jupyter Notebook (Colab)
- nternet Connection

6.3 Supplementary Activities:

Excercise 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)]
```

```
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,
```

> 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

> a. Finding the mean without using statistics module

```
def mean(dataset):
    return sum(salaries) / len(salaries)

mean_wo = mean(salaries)
print(f"Mean of salaries (w/o statistics module): {mean_wo}.")
```

```
Mean of salaries (w/o statistics module): 585690.0.
```

> b. Finding the mean using statistics module

```
from statistics import mean

mean_w = mean(salaries)
print(f"Mean of salaries (w/ imported module): {mean_w}.")
```

```
Mean of salaries (w/ imported module): 585690.0.
```

Median

> a. Finding the median wIthout using the statistics module

```
# Sort the list first
def median(dataset):
    x = salaries.copy()
```

```
    x.sort()

    q = len(x) // 2

    if len(x) % 2 != 0:
        return x[q]
    else:
        return (x[q - 1] + x[q]) / 2

median_wo = median(salaries)
print(f"Median of salaries (w/o statistics module): {median_wo}")
```

```
Median of salaries (w/o statistics module): 589000.0
```

b. Fiding the median using the imported module

```
from statistics import median

median_w = median(salaries)
print(f"Median of salaries (w/ statistics module): {median_w}.")
```

```
Median of salaries (w/ statistics module): 589000.0.
```

Mode

a. Finding the mode without using the statistics module

```
import collections

def mode(dataset):
  # Make a dictionary-like object to keep track of how many times each unique item appears in the dataset.
  # The .most_common() method then sorts these items by how often they appear, from most to least frequent,
  # and gives you the results as a list of tuples.
  mode = collections.Counter(dataset).most_common()

  #the indexing retrieves the most common element(mode) from the sorted list
  return mode[0][0]

mode_wo = mode(salaries)

print(f"Mode of salaries (w/o statistics module): {mode_wo}")
```

```
Mode of salaries (w/o statistics module): 477000.0
```

b. Finding the mode using the imported module

```
from statistics import mode

mode_w = mode(salaries)
print(f"Mode of salaries (w/ statistics module): {mode_w}")
```

```
Mode of salaries (w/ statistics module): 477000.0
```

Sample Variance

a. Finding the sample variance without using the statistics module.

```
def sample_variance(dataset):
    n = len(dataset)
    mean = sum(dataset) / n

    # Calculate the squared deviations
    squared_deviations = [(x - mean) ** 2 for x in dataset]

    # Sum up the squared deviations
    sum_squared_deviations = sum(squared_deviations)
```

```
    # Compute sample variance
    sample_variance = sum_squared_deviations / (n - 1)
    return sample_variance

var_wo = sample_variance(salaries)

print(f"Sample variance of salaries (w/o statistics module): {var_wo:.4f}")
```

```
Sample variance of salaries (w/o statistics module): 70664054444.4444
```

> b. Finding the sample variance using the imported *module*

```
from statistics import variance

var_w = variance(salaries)

print(f"Sample variance of salaries (w/ statistics module): {var_w}")
```

```
Sample variance of salaries (w/ statistics module): 70664054444.44444
```

Standard deviation

> a. Finding the standard deviation without using statistics module.

```
def sample_std(dataset):
  n = len(dataset)
  mean = sum(dataset)/n

  # Calculate the squared deviations
  squared_deviations = [(x - mean) ** 2 for x in dataset]

  # Sum up the squared deviations
  sum_squared_deviations = sum(squared_deviations)

  # Compute sample variance
  std = (sum_squared_deviations / (n - 1))**0.5
  return std

sd_wo = sample_std(salaries)

print(f"Standard deviation of salaries (w/o using statistics module): {sd_wo:.4f}")
```

```
Standard deviation of salaries (w/o using statistics module): 265827.1138
```

> b. Finding the standard deviation using the imported module.

```
from statistics import stdev

sd_w = stdev(salaries)

print(f"Standard deviation of salaries (w/ statistics module): {sd_w}")
```

```
Standard deviation of salaries (w/ statistics module): 265827.11382484
```

Excercise 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

```
import statistics

def range(dataset):
  # Check if the dataset contains at least two values
  if len(dataset)< 2:
    raise ValueError("The data must contain at least two values.")

  # Compute the range by subtracting the maximum and minimum value of the dataset.
  return max(dataset)- min(dataset)

range = range(salaries)

print(f" The range value is {range:.2f}")
```

```
The range value is 995000.00
```

Coefficient of variation

```
from statistics import stdev, mean

def COV(dataset):
  # Calculate the coefficient of variation
  # by dividing the standard deviation by the mean of the dataset.
  COV = stdev(dataset)/mean(dataset)

  # Convert the COV in percentage
  pCOV = COV  * 100

  calc_COV = print(f" Coeff. of variation: {COV}\n Coeff. of variation in Percentage : {pCOV:.2f}")
  return calc_COV

COV(salaries)
```

```
Coeff. of variation: 0.45386998894439035
Coeff. of variation in Percentage : 45.39
```

Interquartile range

```
from statistics import quantiles

def IQR(dataset):
  # Assign a  variable  for the quartile list.
  Qlist = quantiles(dataset)

  # Compute for Interquartile Range where IQR = Q3 -Q1
  IQR = Qlist[-1]- Qlist[0]
  return IQR

iqr = IQR(salaries)

print(f"The dataset's Interquartile Range: {iqr}")
```

```
The dataset's Interquartile Range: 421750.0
```

Quartile coefficient of dispersion

```
def calc_QCD(dataset):
  qlist = quantiles(dataset)

  # QCD =  (Q3-Q1)/(Q3+Q1) * 100
  # Use the calc_IQR for the numerator
  QCD = IQR(dataset)/(qlist[-1]+ qlist[0])

  #Convert the QCD in percentage
  pQCD = QCD * 100

  #Display the value of QCD
  calc_QCD = print(f"Quartile Coefficient of Dispersion: {QCD} \nQCD in Percentage: {pQCD:.6f} %")

  return calc_QCD

print(f"The dataset's Interquartile Range: {iqr}")
```

```
Quartile Coefficient of Dispersion: 0.34491923941934166
QCD in Percentage: 34.491924 %
```

Excercise 3

> Load the diabetes.csv file. Convert the diabetes.csv into dataframe

> Perform the following tasks in the diabetes dataframe:

```
file_path = "/content/diabetes.csv"
```

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

```
diabetes = pd.read_csv(file_path)
```

```
diabetes
```

	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. Identify the column names**

```
column_names = diabetes.columns
print("Column Names:", column_names)
```

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

2. Identify the data types of the data

```
data_types = diabetes.dtypes
print("\nData Types:\n", data_types)
```

```
Data Types:
 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

```
total_records = len(diabetes)-1 # First row is header
print("Total Number of Records:", total_records)

diabetes.shape[0]
```

```
Total Number of Records: 767
768
```

4. Display the first 20 records

```
print("First 20 Records:")
diabetes.head(20)
```

First 20 Records:

	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. Display the last 20 records

```
print("Last 20 Records:")
diabetes.tail(20)
```

Last 20 Records:

	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. Change the Outcome column to Diagnosis

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

	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. Create a new column Classification

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

	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. Create a new dataframe "withDiabetes"

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

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

268 rows × 10 columns

9. Create a new dataframe "noDiabetes"

```
no_diabetes = diabetes[diabetes['Diagnosis'] == 0]
no_diabetes = pd.DataFrame(no_diabetes)
no_diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
5	5	116	74	0	0	25.6	0.201	30	0	No Diabetes
7	10	115	0	0	0	35.3	0.134	29	0	No Diabetes
10	4	110	92	0	0	37.6	0.191	30	0	No Diabetes
...	...	...	...	...	...	...	...	...	...	...
762	9	89	62	0	0	22.5	0.142	33	0	No 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
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

500 rows × 10 columns

10. Create a new dataframe "Pedia"

```
pedia = data[data['Age'] <= 19]
```

11. Create a new dataframe "Adult"

```
adult = diabetes[diabetes['Age'] > 18]
adult = pd.DataFrame(adult)

adult
```

	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

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

```
average_age = np.mean(diabetes['Age'])
average_glucose = np.mean(diabetes['Glucose'])

print(f"Average Age: {average_age:.2f}")
print(f"Average Glucose Value: {average_glucose}")
```

```
Average Age: 33.24
Average Glucose Value: 120.89453125
```

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

```
median_age = np.median(diabetes['Age'])
median_glucose = np.median(diabetes['Glucose'])
```

```
print(f"Median Age: {median_age}")
print(f"Median Glucose Value: {median_glucose}")
```

```
Median Age: 29.0
Median Glucose Value: 117.0
```

14. Use numpy to get the middle values of glucose and age.

```
middle_glucose = np.median(np.sort(diabetes['Glucose']))
middle_age = np.median(np.sort(diabetes['Age']))

print(f"Middle Glucose Value: {middle_glucose}")
print(f"Middle Age: {middle_age}")
```

```
Middle Glucose Value: 117.0
Middle Age: 29.0
```

15. Use numpy to get the standard deviation of the skinthickness

```
std_skinthickness = np.std(diabetes['SkinThickness'])

print(f"Standard Deviation of Skin Thickness: {std_skinthickness:.1f}")


# It's best to present fewer decimal digits to aid easy understanding.
# Use one decimal place for: Means. Standard deviations (SCribbr, n.d.).
```

```
Standard Deviation of Skin Thickness: 15.9
```

Conclusion