Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Dela Cruz, Gabrielle

Section: CPE22S3

Performed on: 06/20/2024

Submitted on: 06/20/2024

Submitted to: Engr. Roman M. Richard

Link to this Colab: https://colab.research.google.com/drive/1HhPabRyzIemOw_a4vwPtZGB6i3I6DeZZ?usp=sharing

6.1 Intended Learning Outcomes

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
- 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

```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]

# Mean
def getMean(data):
  mean = sum(data) / len(data)
  return mean

getMean(salaries)
```

```
585690.0
```

```
# Median
sortSalaries = sorted(salaries)

def getMedian(data):
  if len(data) % 2 == 0:    # Length of sorted list is even.
    leftMedian = data[len(data) // 2 - 1]
    rightMedian = data[len(data) // 2]
    median = (leftMedian + rightMedian) / 2
  else:                     # Length of sorted list is odd.
    median = data[len(data) // 2]
  return median

getMedian(sortSalaries)
```

```
589000.0
```

```
# Mode
from collections import Counter
modes = (Counter(salaries).most_common()) #Get modes and frequency in the list
```

```
modeCount = max(modes)

def getMode(data):
  if len(modes) > 1 and modes[0][1] == modes[1][1]:
    allModes = [mode[0] for mode in modes if mode[1] == modes[0][1]]
    return "There is more than one existing mode:", allModes, "for", modeCount, "times."
  else:
    mode_value, modeCount = modes[0]
    print(f"The mode of random list is {mode_value} for {modeCount} times.")

getMode(salaries)
```

```
The mode of random list is 477000.0 for 3 times.
```

```
# Sample Variance
# Based on the formula where the summation of ((all nth elements - mean)**2)/(given size
summation = []

def getSampleVariance(data):
  for n in data: #Using loop to get all data
    summation.append((n - getMean(data))**2) #Treats all data for the numerator
  totalSum = sum(summation)
  sampleVariance = totalSum / (len(data) - 1) #Determined numerator divided by denominato
  print("Calculated variance is equal to")
  return sampleVariance

SV = getSampleVariance(salaries)
SV
```

```
Calculated variance is equal to
70664054444.44444
```

```
# Sample Standard Deviation
# SSD is equal to the square root of Sample Variance (SV)

def getSSD(data):
  standardDeviation = (getSampleVariance(salaries) ** (1/2))
  print("Calculated standard deviation is equal to", standardDeviation)

getSSD(salaries)
```

```
Calculated variance is equal to
Calculated standard deviation is equal to 375936.3096175852
```

Exercise 2

Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

- Range

- Range
- Coefficient of variation Interquartile range
- Quartile coefficient of dispersion

```
def getRange(data):
  range = max(data) - min(data)
  print("The range is")
  return range

getRange(salaries)
```

```
The range is
995000.0
```

```
from statistics import stdev, mean
def getCOV(data):
  cov = stdev(salaries)/mean(salaries)
  percentcov = round(cov*100, 2)
  result = print("Salaries' Coefficient of Variation is ", cov, "\n COV expressed in Perc
  return result

getCOV(salaries)
```

```
Salaries' Coefficient of Variation is  0.45386998894439035
 COV expressed in Percent:  45.39
```

```
from statistics import quantiles
def getQuartiles(data):
  quartiles = quantiles(data, n=4) # n=4 for quartiles, n=10 for deciles, n=100 for perce
  return quartiles

print("The calculated quartilesa are:", getQuartiles(salaries))
```

```
The calculated quartilesa are: [400500.0, 589000.0, 822250.0]
```

```
from statistics import quantiles
def getInterQuartiles(data):
  quartile_list = getQuartiles(salaries)
  InterQuartile = quartile_list[-1] - quartile_list[0]
  result2 = print("The calculated Interquartile range is:", InterQuartile)
  return result2

print("The calculated quartilesa are:", getQuartiles(salaries))
getInterQuartiles(salaries)
```

```
The calculated quartilesa are: [400500.0, 589000.0, 822250.0]
The calculated Interquartile range is: 421750.0
```

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.

Double-click (or enter) to edit

```
import pandas as pd
import numpy as np

filepath = '/content/diabetes.csv'

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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	

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

```
# Identifying the column names
for col in data.columns:
  print(col)
```

```
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
```

```
#I dentifying data types of data
data.dtypes
```

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

```
# Displaying the total number of records in the file
len(data)
```

```
768
```

```
# Displaying the first 20 records
data.head(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	

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

```
# Displaying the last 20 records
data.tail(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Changing the 'Outcome' column to 'Diagnosis'
data.rename(columns={'Outcome': 'Diagnosis'}, inplace=True) # Uses .rename() for changing c
data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Create a new column 'Classification' that display 'Diabetes' if value = 1; else, 'No Di
data['Classification'] = np.where(data['Diagnosis'] == 1, 'Diabetes', 'No Diabetes')
data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Creating new dataframes 'withDiabetes', getting data only with diabetes
column = 'Diagnosis'
value = [1] # Records with Diagnosis = 1 where diabetes is positive

check = data[column].isin(value) # .isin() checks for values in columns
withDiabetes = data[check]
withDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Creating a new dataframe 'noDiabetes' that collects records negative of diabetes
column = 'Diagnosis'
value = [0] # Records with Diagnosis = 0 where diabetes is negative

check = data[column].isin(value) # .isin() checks for values in columns
noDiabetes = data[check]
noDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Creating a new dataframe 'Pedia' that gathers data with age 0 to 19
column = 'Age'
minimum = 0
maximum = 19

check = data[column].between(minimum, maximum) # .between() checks for values in columns (i
Pedia = data[check]
Pedia # No result exists but only the table column since the dataset has no record at age )
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF

```
# Creating a new dataframe 'Adult' that collects records with age greater than 19
column = 'Age'
minimum = 20

check = data[column].ge(minimum) # .ge() checks for values in columns (in this case, the
Adult = data[check]
Adult
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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

```
# Use numpy to get average age and glucose value
print("Average Recorded Age:", np.mean(data['Age'])) # .mean() is used for average/mean o
print("Average Recorded Glucose:", np.mean(data['Glucose']))
```

```
Average Recorded Age: 33.240885416666664
Average Recorded Glucose: 120.89453125
```

```
# Use numpy to get the median age and glucose value
print("Median Age:", np.median(data['Age'])) # .median() is used for median of data
print("Median Glucose:", np.median(data['Glucose']))
```

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

```
# Use numpy to get middle (median) values of glucose and age
print("Middle Glucose:", np.median(data['Glucose']))
print("Middle Age:", np.median(data['Age']))
```

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

```
# Use numpy to get SD of 'skinthickness' column
print("The Standard Deviation of Skin Thickness is:", np.std(data['SkinThickness']))
```

```
The Standard Deviation of Skin Thickness is: 15.941828626496939
```

6.4 Conclusion

This Hands-on activity introduced students towards the Midterm period of the course wherein concepts of data analysis tools where discussed thoroughly. In this activity, modules of numpy and pandas were used to create dataframes that visually presented data from the uploaded given datasets. For numerical computation of data in Python, "NumPy" is used whereas "Pandas" is used for the analysis and manipulation of data. Both modules are useful towards the introduction to Data Science given that they give ample leniency in the data analysis process of datasets.