Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Dela Cruz, Eugene

Section: CPE22S3

Performed on: 03/07/2024

Submitted on: 03/07/2024

Submitted to: Engr. Roman M. 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

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(salaries) / len(salaries) #divide salaries from len number of salaries
print("Mean: ", mean)
```

```
    Mean:  585690.0
```

```
sorted_salaries = sorted(salaries)
n = len(sorted_salaries)

if n % 2 == 0: #calculate for median
    median = (sorted_salaries[n//2 - 1] + sorted_salaries[n//2]) / 2 # if n is odd, the middle of sorted_salaries is the median
else:
    median = sorted_salaries[n//2] # if n is even, the median is the two middle value of sorted_salaries

print("Median:", median)
```

```
    Median: 589000.0
```

```
from collections import Counter
mode = Counter(salaries)

print("Mode and its count: ", mode.most_common(1))
```

```
    Mode:  [(477000.0, 3)]
```

```
sum_of_squares = sum((x - mean) ** 2 for x in salaries) #calculate the sum of squares of difference from the mean

sample_variance = sum_of_squares / (len(salaries) - 1) # get the sample variance by dividing the sum of squares to the number of salaries minus 1

print("Sample variance: ", sample_variance)
```

```
    Sample variance:  70664054444.44444
```

```
sample_standard = sample_variance**0.5

print("Sample standard deviation: ", sample_standard)
```

```
    Sample standard deviation:  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

```
import statistics

range = max(salaries) - min(salaries)
def calculate_cv_iqr(salaries):

  cv = None
  if mean != 0:

    cv = sample_standard / mean # calculate standard deviation using the population standard deviation
  q1 = statistics.median(sorted_salaries[:len(sorted_salaries) // 2]) # function to calculate quartile
  q3 = statistics.median(sorted_salaries[len(sorted_salaries) // 2:]) # function to calculate quartile

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

  return cv, iqr

cv, iqr = calculate_cv_iqr(salaries) # calculate for CV and IQR

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

```
Range:  995000.0
Coefficient of Variation (CV): 0.45386998894439035
Interquartile Range (IQR): 417500.0
```

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.

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

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

	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

Next steps: View recommended plots

```
# 1 Identify the column names
diabetes.columns
```

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

```
# 2 Identify the data types of the data
diabetes.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
num_record = len(diabetes)
print("Total number of records: ", num_record)
```

```
Total number of records:  768
```

```
# 4 Display the first 20 records
diabetes[0:20]
```

	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 Display the last 20 records
diabetes[-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 Change 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

Next steps: View recommended plots

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

Next steps: View recommended plots

```
# 8 Create a new dataframe "withDiabetes" that gathers data with diabetes
newDiabetesdf = pd.DataFrame(diabetes, columns=['withDiabetes', 'noDiabetes', 'Pedia', 'Adult'])
newDiabetesdf
```

	withDiabetes	noDiabetes	Pedia	Adult
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN
...	...	...	...	...
763	NaN	NaN	NaN	NaN
764	NaN	NaN	NaN	NaN
765	NaN	NaN	NaN	NaN
766	NaN	NaN	NaN	NaN
767	NaN	NaN	NaN	NaN

768 rows × 4 columns

Next steps: View recommended plots

```
# 9 Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetesdf = pd.DataFrame(diabetes)
noDiabetesdf = diabetes[diabetes['Diagnosis'] == 0].copy()
noDiabetesdf
```

	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

Next View recommended plots

```
#10 Create a new dataframe "Pedia" that gathers data with age 0 to 19
pediadf = pd.DataFrame(diabetes)
pediadf = diabetes[diabetes['Age'] <= 19].copy()
pediadf
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

```
#11 Create a new dataframe "Adult" that gathers data with age greater than 19
adultdf = pd.DataFrame(diabetes)
adultdf = diabetes[diabetes['Age']> 19].copy()
adultdf
```

	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

Next steps: View recommended plots

```
# 12 Use numpy to get the average age and glucose value.
ave_ageandglu = np.mean(diabetes['Age']), np.mean(diabetes['Glucose'])
ave_ageandglu
```

```
(33.240885416666664, 120.89453125)
```

```
#13 Use numpy to get the median age and glucose value.
median_ageandglu = np.median(diabetes['Age']), np.median(diabetes['Glucose'])
median_ageandglu
```

```
(29.0, 117.0)
```

```
#14 Use numpy to get the middle values of glucose and age.
midval_gluandage = np.median(median_ageandglu)
midval_gluandage
```

```
73.0
```

```
#15 Use numpy to get the standard deviation of the skinthickness.
stdev_skinthick = np.std(diabetes['SkinThickness'])
stdev_skinthick
```

```
15.941828626496939
```

6.4 Conclusion

I therefore conclude that I've learned how to handle data more effectively. NumPy helped me with math, while Pandas made organizing data easy. Together, they've been like my helpful buddies, guiding me through sorting, filtering, and grouping data. By using these tools, I've become better at understanding and making sense of my data. I've also applied these tools to statistics such as calculating mean, median, mode, standard deviation in which is easier to apply because of keywords compared to using it without Data Analysis and tools.