CPE311 Computational Thinking with Python

Name: Ballesteros, Angelo

Section: CPE22S2

Performed on: 6/23/2024

Submitted on: 6/23/2024

Submitted to: Engr. Roman M. Richard

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

```
# data generated

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


from statistics import median
from math import isnan
from itertools import filterfalse


# mean

mean = sum(salaries) / len(salaries)
```

```
# median

sortSalaries = sorted(salaries)
n = len(sortSalaries)
if n % 2 == 0:
    median = (sortSalaries[n//2 - 1] + sortSalaries[n//2]) / 2
else:
    median = sortSalaries[n//2]


# mode

import math
from collections import Counter

counts = Counter(salaries)
mode= max(counts, key=counts.get)


# sample variance

meanSquare = [(x - mean) ** 2 for x in salaries]
sample_var = sum(meanSquare) / (len(salaries) - 1)


# sample standard deviation

sample_standardDev = math.sqrt(sample_var)


# output

print(f"Mean:", mean)
print(f"Median:", median)
print(f"Mode:", mode)
print(f"Sample Variance:", sample_var)
print(f"Sample Standard Deviation:", sample_standardDev)
```

```
Mean: 585690.0
Median: 589000.0
Mode: 477000.0
Sample Variance: 70664054444.44444
Sample Standard Deviation: 265827.11382484
```

```
# reference of output

import pandas as pd

df = pd.DataFrame(salaries)
df.describe()
```

	0
count	100.000000
mean	585690.000000
std	265827.113825
min	1000.000000
25%	403500.000000
50%	589000.000000
75%	816750.000000
max	996000.000000

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

range = max(salaries) - min(salaries)


# coefficient of variation

mean = sum(salaries) / len(salaries)
sample_var = sum((x - mean) ** 2 for x in salaries) / (len(salaries) - 1)
sample_standardDev = math.sqrt(sample_var)
COV = (sample_standardDev / mean) * 100


# interquartile range

sortSalaries = sorted(salaries)
n = len(sortSalaries)
q1 = sortSalaries[n // 4]
q3 = sortSalaries[(3 * n) // 4]
iqr = q3 - q1


qd = (q3 - q1) / (q3 + q1)


# output

print(f"Range:", range)
print(f"Coefficient of Variation:%", COV)
print(f"Interquartile Range:", iqr)
print(f"Quartile Coefficient of Dispersion:", qd)
```

```
Range: 995000.0
Coefficient of Variation:% 45.38699889443903
Interquartile Range: 420000.0
Quartile Coefficient of Dispersion: 0.34146341463414637
```

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.

```
# upload diabetes.csv

filepath = '/content/diabetes.csv'
data = pd.read_csv(filepath)

data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreel
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

Next steps: Generate code with data View recommended plots

```
# identify column names

data.columns
```

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

```
# identify data types

data.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
```

```
# display total number of records

data.shape[0]
```

```
768
```

```
# display first 20 records

data.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: Generate code with data View recommended plots

```
# display last 20 records

data.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	

```
# change outcome column to diagnosis

data = data.rename(columns={"Outcome": "Diagnosis"})


# create column Classification

data['Classification'] = data['Diagnosis'].apply(lambda x: 'Diabetes' if x == 1 else 'No Diabetes')


# create dataframe withDiabetes

withDiabetes = data[data['Diagnosis'] == 1]


# create dataframe noDiabetes

noDiabetes = data[data['Diagnosis'] == 0]


# create dataframe Pedia

Pedia = data[data['Age'] <= 19]


# create dataframe Adult

Adult = data[data['Age'] > 19]


# use numpy to get average age and glucose value

import numpy as np

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

print(f"Average age:", average_age)
print(f"Average glucose value:", average_glucose)
```

```
Average age: 33.240885416666664
Average glucose value: 120.89453125
```

```
# use numpy to get median age and glucose value

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

print(f"Median age:", median_age)
print(f"Median glucose value:", median_glucose)
```

```
Median age: 29.0
Median glucose value: 117.0
```

```
# use numpy to get middle values of glucose and age

middle_glucose = np.median(data['Glucose'])
middle_age = np.median(data['Age'])

print(f"Middle glucose value:", middle_glucose)
print(f"Middle age:", middle_age)
```

```
Middle glucose value: 117.0
Middle age: 29.0
```

```
# use numpy to get standard deviation of skin thickness

skinthickness_std = np.std(data['SkinThickness'])

print(f"Standard deviation of skinthickness:", skinthickness_std)
```

```
Standard deviation of skinthickness: 15.941828626496939
```

6.4 Conclusion

In conclusion, I have applied the intended learning outcomes for the activity. I have use the pandas and numpys for importing and analyzing the given data which is the diabetes file. I got a little problem in exercise 1 and 2 because the statistics value does not match in the statistics table of the random generated data but I have manage to fixed it. Furthermore, this activity does connect in my previous course which is MATH 019A - Engineering Data Analysis so it is easy for me in understanding the measures of central tendency but in terms of coding it in Python, it was hard so I did get some guides. Lastly, I thought it was not possible in converting the csv file into dataframe yet I learned how to upload a file in the file tab to make it in dataframe.