Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Vista, Jens Liam P. 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
import pandas as pd
import numpy as np
import statistics as stat
from scipy.stats import iqr
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 Code
mean = sum(salaries)/len(salaries) # I need this
print(mean)
```

```
585690.0
```

```
# Median Code
if len(salaries) % 2 == 0:
  mid = int(len(salaries)/2)
  print((salaries[mid] + salaries[mid-1])/2)
else:
  print(salaries[len(salaries-1)/2])
```

```
885500.0
```

```
# Mode Code
meow = {}
for i in salaries:
  meow[i] = salaries.count(i)
print("My mode: ",max(meow, key=meow.get))
from scipy import stats
print("Scipy mode: ",stats.mode(salaries))
print("Scipy mode: ",*stats.mode(salaries))
```

```
My mode:  477000.0
Scipy mode:  ModeResult(mode=477000.0, count=3)
Scipy mode:  477000.0 3
```

```
def mode(salaries):
    # counter of occurrences
    counts = {}
    for salary in salaries:
        counts[salary] = counts.get(salary, 0) + 1

    # finding the max value in count
    max_count = max(counts.values())

    # find all items with max count
    modes = [salary for salary, count in counts.items() if count == max_count]

    if len(modes) == len(set(salaries)):
        return "No mode"  # no mode when all items occurred only once
    else:
        return modes  # return list of all possible modes
print('my mode: ', mode(salaries))
print('Pandas Mode: ', stats.mode(salaries)[0])
```

```
my mode:  [477000.0]
Pandas Mode:  477000.0
```

```
# Sample Variance
memo = []
for i in salaries:
  memo.append(pow((i-mean), 2))
print('my sample variance: ', sum(memo)/(len(salaries)-1))
print('Scipy Variance: ', stat.variance(salaries))
```

```
my sample variance:  70664054444.44444
Scipy Variance:  70664054444.44444
```

```
# sample standard deviation
import math
print('My standard deviation: ', math.sqrt(stat.pvariance(salaries)))
print('scipy standard deviation: ', stat.pstdev(salaries))
```

```
My standard deviation:  264494.6386980273
scipy standard deviation:  264494.6386980273
```

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
print('My range: ', max(salaries) - min(salaries))
print("Numpy range: ", np.ptp(salaries))
```

```
My range:  995000.0
Numpy range:  995000.0
```

```
# Coefficient of variation Interquartile range
iqr_value = iqr(salaries)
quartile_deviation = iqr_value / 2

print("Interquartile Range (IQR):", iqr_value)
```

```
Interquartile Range (IQR): 413250.0
```

```
# Quartile coefficient of dispersion

q1 = np.percentile(salaries, 25)
q3 = np.percentile(salaries, 75)
print('Quartile coefficient of dispersion', (q3 - q1)/(q3 + q1))
```

```
Quartile coefficient of dispersion 0.338660110633067
```

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.

```
import pandas as pd

df = pd.read_csv('/content/diabetes (1).csv')
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeI
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

```
# 1. Identify the column names
for column_name in df.columns:
    print(column_name)
```

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

```
# 2. Identify the data types of the data
df.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

total_records = df.shape[0]
print("Total number of records:", total_records)
print(df.count()) # the data was clean because all of it was 768
```

```
Total number of records: 768
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

```
# 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	DiabetesPedigreeI
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	

```
print(dict([('Outcome', 'Diagnosis')]))
```

```
{'Outcome': 'Diagnosis'}
```

```
# 6. Change the Outcome to Diagnosis
df.rename(columns = dict([('Outcome', 'Diagnosis')]), inplace = True)
df.head()
```

	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	

Next steps: View recommended plots

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

	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	

Next steps: View recommended plots

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	8	183	64	0	0	23.3	
2	0	137	40	35	168	43.1	
3	3	78	50	32	88	31.0	
4	2	197	70	45	543	30.5	

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	1	85	66	29	0	26.6	
1	1	89	66	23	94	28.1	
2	5	116	74	0	0	25.6	
3	10	115	0	0	0	35.3	
4	4	110	92	0	0	37.6	

```
min(df['Age'])
```

```
21
```

```
# 10. Create a new dataframe "Pedia" that gathers data with age 0 to 19

pedia = df[df['Age'] < 19].copy()
pedia.head() # None since the min age in the data is 21
```

	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()
```

	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	

Next steps: View recommended plots

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

```
Average Age:  33.240885416666664
Average glucose value:  120.89453125
```

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

```
Age median:  29.0
glucose value median:  117.0
```

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

```
Middle value of Age: 29.0
Middle value of Glucose: 117.0
```