Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: CATULAY, WESLIE JEE

Section: CPE22S2

Performed on: 6/20/2024

Submitted on: 6/22/2024

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcome

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

```
# Getting the Mean

def mean(salaries):
  return sum(salaries)/len(salaries)
print(f"This is the Mean: {mean(salaries)}")
```

```
This is the Mean: 585690.0
```

```
# Getting the Median

def median(salaries):
  SalarSort = sorted(salaries)
  Med = len(SalarSort) // 2

  if len(SalarSort) % 2 == 0:
     Median = (SalarSort[Med - 1] + SalarSort[Med]) / 2

  else:
      Median = SalarSort[Med]
  return Median

print(f"This the Median: {median(salaries)}")
```

```
This the Median: 589000.0
```

```
#Getting the Mode

def mode(salaries):

  ModeNum = {}
  for i in salaries:

    ModeNum[i] = ModeNum.get(i, 0) + 1

  maxVal = max(ModeNum.values())

  Mode_Items = [ i for i, count in ModeNum.items() if count == maxVal ]

  Frequency = maxVal

  return Mode_Items, Frequency

print(f"This is the Frequency: {mode(salaries)}")
```

```
This is the Frequency: ([477000.0], 3)
```

```
# Getting the Sample Variance

def sampleVar(salaries):
  var = len(salaries)

  Mean = mean(salaries)

  sample_var = sum((i - Mean)**2 for i in salaries)/(var - 1)

  return sample_var

print(f"This is the Sample Variance: {sampleVar(salaries)}")
```

```
This is the Sample Variance: 70664054444.44444
```

```
# Getting the Standard Deviation

def stanDev(salaries):

  Dev = sampleVar(salaries)

  SD = Dev ** 0.5

  return SD

print(f"This is the Standard Deviation: {stanDev(salaries)}")
```

```
This is the 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

```
# Getting the Range

def Range(salaries):

  haba = max(salaries) - min(salaries)

  return haba

print(f"This is the Range: {Range(salaries)}")
```

```
This is the Range: 995000.0
```

```
# Getting the Coeffiecient of variation Interquartile range

def Coef_IR(salaries):
  M = mean(salaries)
  SDV = stanDev(salaries)

  coef = SDV/M

  import statistics as st
  AQ1 = st.quantiles(salaries, n=4)[0]
  AQ2 = st.quantiles(salaries, n=4)[2]

  IR = AQ2 - AQ1

  return coef,IR


print(f"This is the  answer for both Coefficient of variation & Interquartile range: {Coef_I
```

```
This is the  answer for both Coefficient of variation & Interquartile range: (0.4538699
```

```
# Getting  Quartile coeffiecient of dispersion

def QCD(salaries):

  import statistics as st
  AQ1 = st.quantiles(salaries, n=4)[0]
  AQ2 = st.quantiles(salaries, n=4)[2]

  IR = AQ2 - AQ1
  Med = median(salaries)

  QuCD = IR/Med

  return QuCD
QuCoefDis = QCD(salaries)
print(f"This is the Quartile coefficient of dispersion: {QCD(salaries)}")
```

```
This is the Quartile coefficient of dispersion: 0.716044142614601
```

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.

```
# Data Analysis

import pandas as pd
import numpy as np
filepath ='/content/diabetes.csv'
DA = pd.read_csv(filepath)
```

```
DA
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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 DA View recommended plots

```
# Identifying Columns

columns = DA.columns
print(f"This the Columns names::::: {columns}")
```

```
This the Columns names::::: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThick
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
# Identifying data types fo data

dataTypes = DA.dtypes
print(dataTypes)
```

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

```
# Total Records

TotalRecords = DA.shape[0]
print(f"This is the Total Records: {TotalRecords}")
```

```
This is the Total Records: 768
```

```
# First 20 Records

DA[: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	

```
# Last 20 records

DA[749:]
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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 the Outcome column to Diagnosis
DA.rename(columns={'Outcome':'Diagnosis'}, inplace=True)

DA
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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 DA View recommended plots

```
# Create a new column Classification that display "Diabetes" if the value of outcome is 1 ,

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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

Next steps: Generate code with DA View recommended plots

```
# Create a new dataframe "withDiabetes" that gathers data with diabetes

withDiabetes = DA[DA['Classification']=='Diabetes']
withDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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

Next steps: Generate code with withDiabetes View recommended plots

```
# Create a new dataframe "noDiabetes" thats gathers data with no diabetes

noDiabetes = DA[DA['Classification']=='No Diabetes']
noDiabetes
```

Next Generate code with noDiabetes View recommended plots

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
1	1	85	66	29	0	26.6	

```
#  Create a new dataframe "Pedia" that gathers data with age 0 to 19
Pedia = DA[DA['Age'] <= 19 ]
Pedia
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc