PDS ASSIGNMENT-3

VISHAL REDDY BOMMA

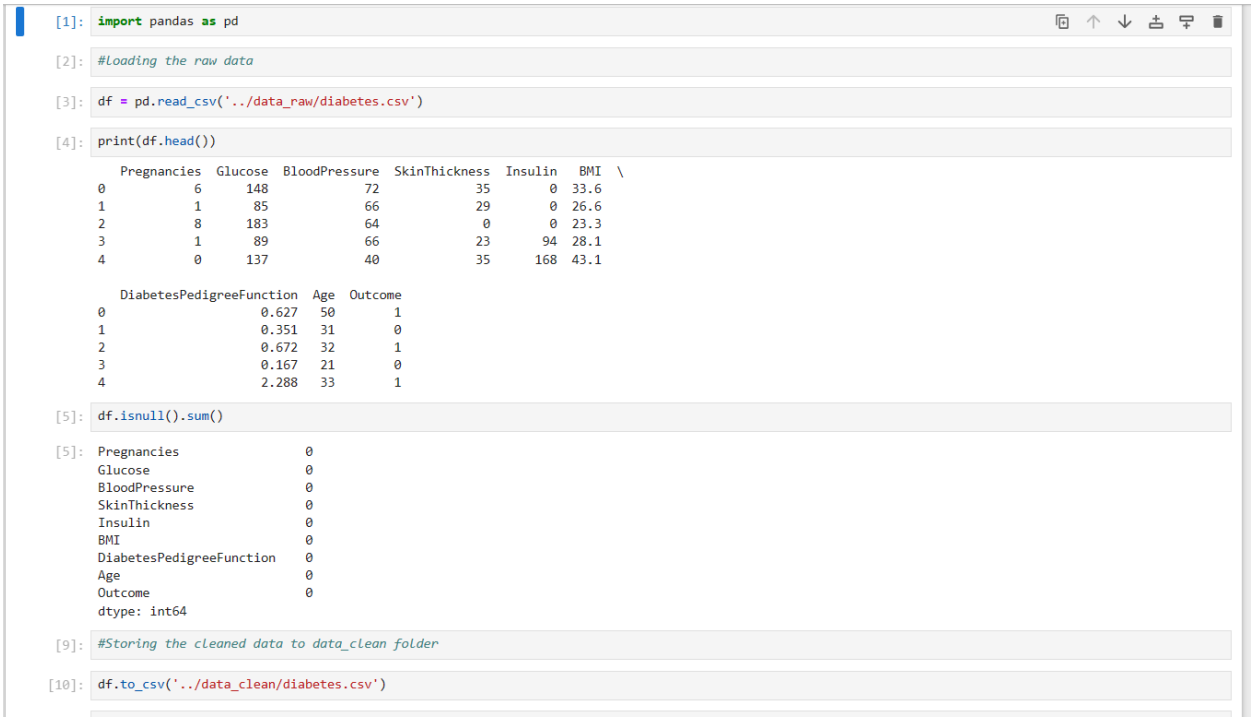
16340457

STEP1: DATA COLLECTION Reading the data from the diabetes.csv

https://app.box.com/s/7qv44umhw0vnzgmoe9krfkfkv5kf2atv

STEP2: DATA CLEANING

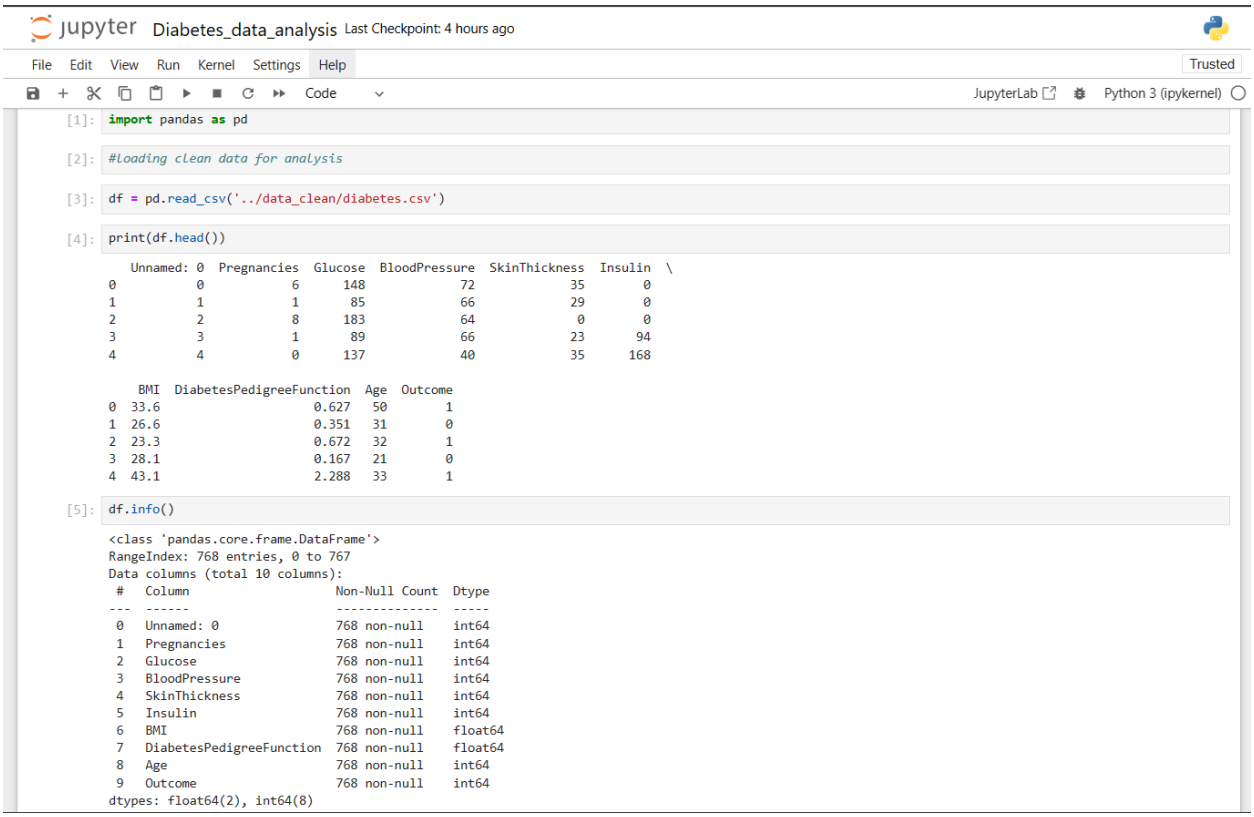
- Loading the given raw data in a dataframe using pandas.
- Checking if there are any null values present.
- if there are null values , we remove them and store the data as clean data.



```
[1]: import pandas as pd

[2]: #Loading the raw data

[3]: df = pd.read_csv('../data_raw/diabetes.csv')

[4]: print(df.head())

   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

[5]: df.isnull().sum()

[5]: Pregnancies                 0
     Glucose                     0
     BloodPressure               0
     SkinThickness               0
     Insulin                     0
     BMI                         0
     DiabetesPedigreeFunction    0
     Age                         0
     Outcome                     0
     dtype: int64

[9]: #Storing the cleaned data to data_clean folder

[10]: df.to_csv('../data_clean/diabetes.csv')
```

STEP3: DATA ANALYSIS

- Loading the cleaned data into a dataframe



TASK A:

a) set a seed (to ensure work reproducibility) and take a random sample of 25 observations and find the mean Glucose and highest Glucose values of this sample and compare these statistics with the population statistics of the same variable. You should use charts for this comparison.
(5 points)

- Importing the libraries matplotlib, seaborn, numpy
- generating sample from population with size 25(setting the seed as id num last 3 digits which is 457 using randam_state)
- then we find the mean and highest values of the sample.

```
[6]: #importing the libraries
     import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np

[7]: #generating sample from population with size 25( setting the seed as id num last 3 digits using randam_state)

[8]: sample_population= df.sample(n= 25, random_state= 457)

[9]: sample_population.head()
```

[9]:

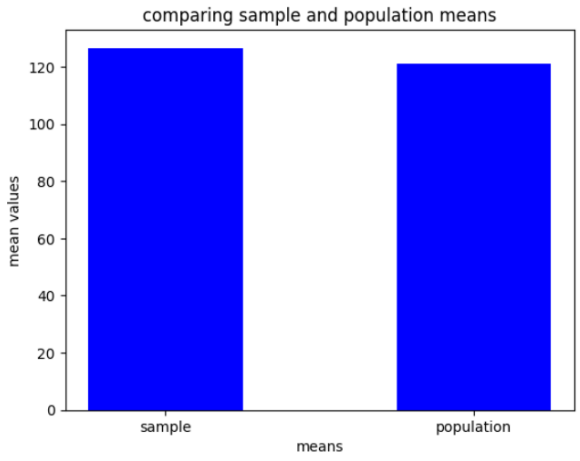
	Unnamed: 0	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
125	125	1	88	30	42	99	55.0	0.496	26	1
328	328	2	102	86	36	120	45.5	0.127	23	1
759	759	6	190	92	0	0	35.5	0.278	66	1
186	186	8	181	68	36	495	30.1	0.615	60	1
104	104	2	85	65	0	0	39.6	0.930	27	0

```
[10]: #finding mean for Glucose column sample

[11]: sample_mean=sample_population["Glucose"].mean()

[12]: #finding max value for Glucose column sample data

[13]: sample_max_value=sample_population["Glucose"].max()

[14]: #finding mean for Glucose column for whole population

[15]: population_mean=df["Glucose"].mean()

[16]: #finding max value for Gl;ucose column for whole population
```

```
[15]: population_mean=df["Glucose"].mean()

[16]: #finding max value for Gl;ucose column for whole population

[17]: population_max_value=df["Glucose"].max()

[18]: # creating list for sample and population mean for visualization

[19]: keys=["sample","population"]

[20]: values=[sample_mean,population_mean]

[21]: print(values)
      [126.52, 120.89453125]
```

- visualizing the sample and population means

```
[23]: plt.bar(keys, values, color ='blue',
              width = 0.5)

      plt.xlabel("means")
      plt.ylabel("mean values")
      plt.title("comparing sample and population means")


      plt.savefig("../results/sample_population_mean_comparison.png")

      plt.show()
```

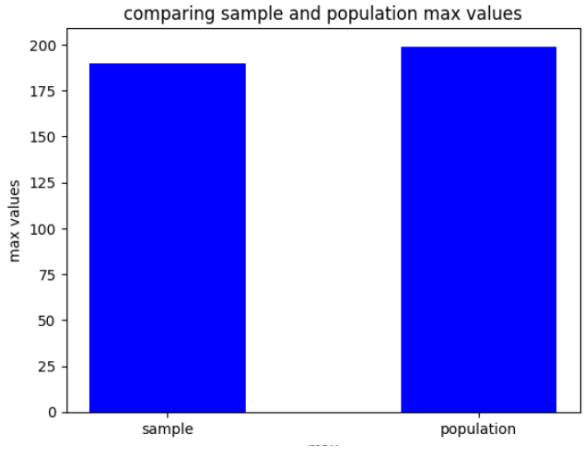


- Generating lists for max glucose values for sample and population data set.

```
[24]: #generating lists for max glucose values for sample and population data set

[25]: keys=["sample","population"]

[60]: values=[sample_max_value,population_max_value]
      print(values)

      [190, 199]
```

- visualizing the sample and population max value

```
[27]: #visualizing the sample and population max value

[28]: plt.bar(keys, values, color ='blue',
              width = 0.5)

       plt.xlabel("max")
       plt.ylabel("max values")
       plt.title("comparing sample and population max values")


       plt.savefig("../results/sample_population_max.png")

       plt.show()
```



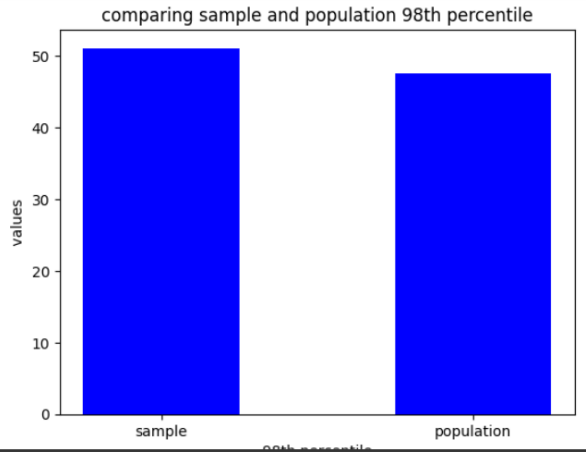
TaskB

Find the 98th percentile of BMI of your sample and the population and compare the results using charts. (5 points)

- finding the 98th percentile for BMI for sample and population
- generating lists to visualize 98th percentile of BMI.
- Visualizing the 98th percentile of BMI for sample and population data sets.

```
[29]: # finding the 98th percentile for BMI for sample and population

[30]: sample_98=sample_population.BMI.quantile(0.98)  # 98th percentile

[31]: population_98=df.BMI.quantile(0.98)

[32]: #generating lists to visualize 98 th percentile of BMI

[33]: keys=["sample","population"]

      values=[sample_98,population_98]
```

```
[34]: # visualizing the 98th percentile of BMI for sample and population data sets

[35]: plt.bar(keys, values, color ='blue',
              width = 0.5)

      plt.xlabel("98th percentile")
      plt.ylabel(" values ")
      plt.title("comparing sample and population 98th percentile")


      plt.savefig("../results/sample_population_98th percentile.png")

      plt.show()
```



TaskC:

Using bootstrap (replace= True), create 500 samples (of 150 observation each) from the population and find the average mean, standard deviation and percentile for BloodPressure and compare this with these statistics from the population for the same variable. Again, you should create charts for this comparison. Report on your findings. (10 points)

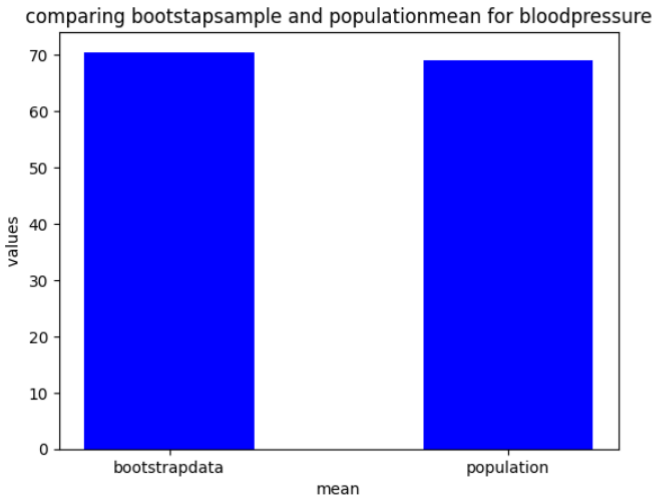
- Creating 500 sample using bootstrap

```
[36]: # Creating 500 sample using bootstrap

[37]: bootstrap_dataframe= df.sample(n=500,replace=True ,random_state= 457)
      print(bootstrap_dataframe)
```

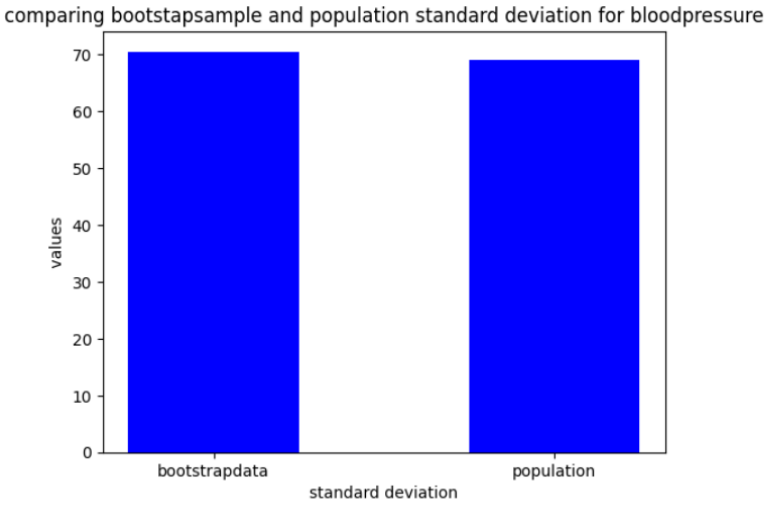
```
     Unnamed: 0  Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  \
354         354            3       90             78              0        0
147         147            2      106             64             35      119
218         218            5       85             74             22        0
436         436           12      140             85             33        0
640         640            0      102             86             17      105
..          ...          ...      ...            ...            ...      ...
664         664            6      115             60             39        0
332         332            1      180              0              0        0
345         345            8      126             88             36      108
588         588            3      176             86             27      156
410         410            6      102             90             39        0

      BMI  DiabetesPedigreeFunction  Age  Outcome
354  42.7                     0.559   21        0
147  30.5                     1.400   34        0
218  29.0                     1.224   32        1
436  37.4                     0.244   41        0
640  29.3                     0.695   27        0
..    ...                       ...  ...      ...
664  33.7                     0.245   40        1
332  43.3                     0.282   41        1
345  38.5                     0.349   49        0
588  33.3                     1.154   52        1
410  35.7                     0.674   28        0

[500 rows x 10 columns]
```

- Finding mean for bootstrap sample and population data set
- Generating lists for mean values of blood pressure for bootstrap data and population
- Visualizing the means of blood pressure for bootstrap sample and population

```
[38]: # finding mean for bootsraop sample and population data set

[39]: bootstrap_mean=bootstrap_dataframe["BloodPressure"].mean()
      population_mean=df["BloodPressure"].mean()

[40]: #generating lists for mean values of blood presuure for bootstrap data and population

[41]: keys=["bootstrapdata","population"]
      values=[bootstrap_mean,population_mean]

[42]: #visualizing the means of blood pressure for bootstrap sample and population

[43]: plt.bar(keys, values, color ='blue',
              width = 0.5)

      plt.xlabel("mean")
      plt.ylabel(" values ")
      plt.title("comparing bootstapsample and populationmean for bloodpressure")


      plt.savefig("../results/botstrapSample_population_mean.png")

      plt.show()
```




```
[44]: #from the figure we can see that mean for bootstrap sample and population data set are almost equal

[45]: # finding the standard deviation fr bootsrap sample and population

[46]: bootstrap_std=bootstrap_dataframe["BloodPressure"].std()
      population_std=df["BloodPressure"].std()

[47]: #creating lists for standard deviation of bloodpresuiure for bootstrap sample and population data set

[48]: #visualizing standard deviation of blood pressure for bootstrap data nad population data set

[49]: plt.bar(keys, values, color ='blue',
              width = 0.5)

      plt.xlabel("standard deviation")
      plt.ylabel(" values ")
      plt.title("comparing bootstapsample and population standard deviation for bloodpressure")


      plt.savefig("../results/botstrapSample_population_standarddeviation.png")

      plt.show()
```



[50]:
```
#from the figure  we can see that standard deviation fotr bootstrap sample and populatrion dataset are almost equal
```

[51]:
```
# finding the 98th percentile for Bloodpressuire for bootstrap sample and population
```

[52]:
```
Bootstrap_sample_98=bootstrap_dataframe.BloodPressure.quantile(0.98)  # 98th percent
```

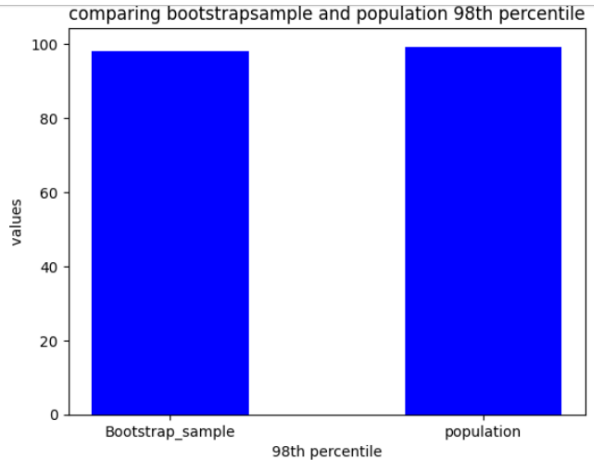
[53]:
```
population_98=df.BloodPressure.quantile(0.98)
```

[54]:
```
#generating lists to visualize 98 th percentile of Bloodpressure
```

[55]:
```
keys=["Bootstrap_sample","population"]

values=[Bootstrap_sample_98,population_98]
```

[56]:
```
# visualizing the 98th percentile of Bloodpressure for bootstrapsample and population data sets
```

[58]:
```
plt.bar(keys, values, color ='blue',
        width = 0.5)

plt.xlabel("98th percentile")
plt.ylabel(" values ")
plt.title("comparing bootstrapsample and population 98th percentile")


plt.savefig("../results/bootstrap_sample_population_98th percentile.png")

plt.show()
```



[59]:
```
# from the above figure we can say that the bootstrap sample and population 98th percentile values are almost equal
```

[ ]:
```

```