PDS Assignment

Jithendra Pavuluri

16343746

Question - 2

A)

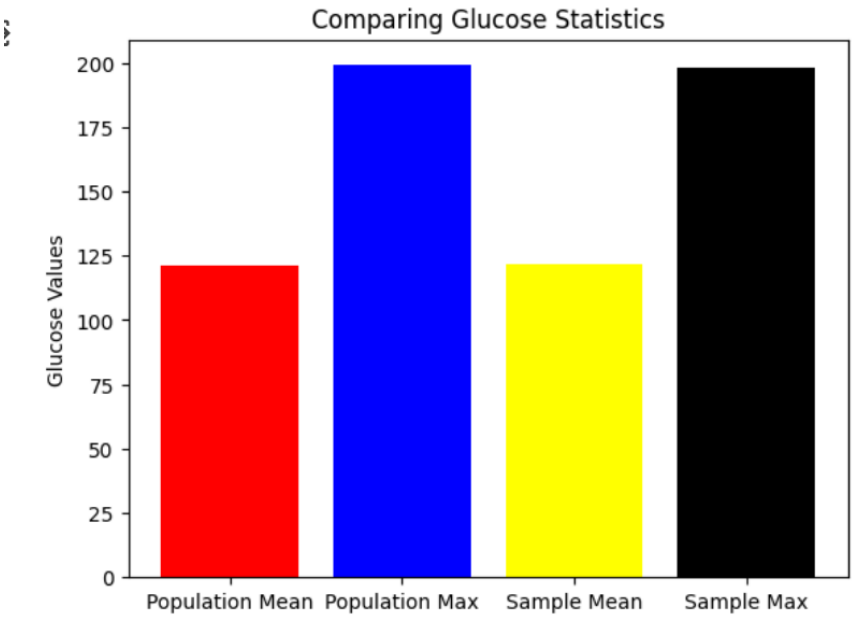
In this part of the code:

```
#A)

np.random.seed(789)
sample = data.sample(25)
population_mean_glucose = data['Glucose'].mean()
population_max_glucose = data['Glucose'].max()
sample_mean_glucose = sample['Glucose'].mean()
sample_max_glucose = sample['Glucose'].max()

#Plotting bar chart for comparison
name_label = ['Population Mean', 'Population Max', 'Sample Mean', 'Sample Max']
values = [population_mean_glucose, population_max_glucose, sample_mean_glucose, sample_max_glucose]
plt.bar(name_label, values, color=['red', 'blue', 'yellow', 'black'])
plt.title('Comparing Glucose Statistics')
plt.ylabel('Glucose Values')
plt.show()
```

- It sets a random seed to ensure reproducibility.
- It randomly selects 25 rows from the dataset as a sample.
- It calculates the mean and maximum glucose levels for both the entire population and the sample.



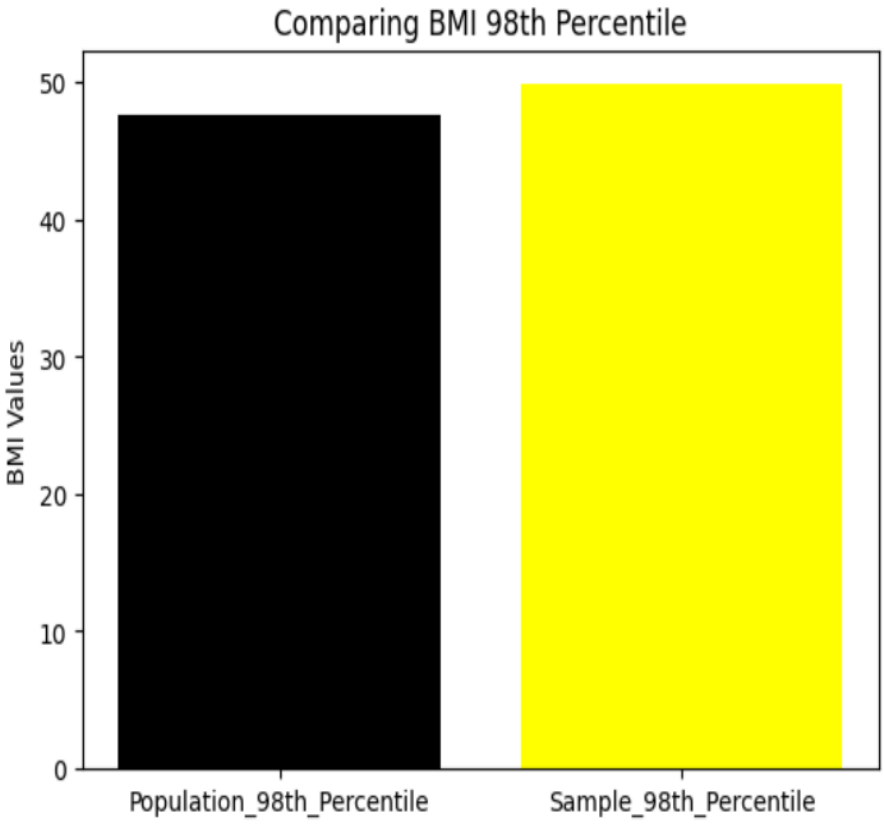
- The output is a bar chart comparing the population mean and maximum glucose levels with the sample mean and maximum glucose levels.

B)

```
#B)

population_percentage = np.percentile(data['BMI'], 98)
sample_percentage = np.percentile(sample['BMI'], 98)
label = ['Population_98th_Percentile', 'Sample_98th_Percentile']
value = [population_percentage, sample_percentage]
plt.bar(label, value, color=['black', 'yellow'])
plt.title('Comparing BMI 98th Percentile')
plt.ylabel('BMI Values')
plt.show()
```

- It calculates the 98th percentile of BMI for both the entire population and the sample.



- The output is a bar chart comparing the 98th percentile of BMI for the population and the sample.

C)

```
#C)

bootstrap_sample = 500
sample_size = 150

bootstrap_mean = np.zeros(bootstrap_sample)
bootstrap_std_deviation = np.zeros(bootstrap_sample)
bootstrap_percentile = np.zeros(bootstrap_sample)

for i in range(bootstrap_sample):
    bootstrap = data['BloodPressure'].sample(sample_size, replace=True)
    bootstrap_mean[i] = bootstrap.mean()
    bootstrap_std_deviation[i] = bootstrap.std()
    bootstrap_percentile[i] = np.percentile(bootstrap, 98)

population_mean = data['BloodPressure'].mean()
population_std_deviation = data['BloodPressure'].std()
population_percentile = np.percentile(data['BloodPressure'], 98)

mean=np.mean(bootstrap_mean)
sd=np.mean(bootstrap_std_deviation)
percentile=np.mean(bootstrap_percentile)
```

- It sets the number of bootstrap samples to 500 and the sample size for each bootstrap sample to 150.
- It performs bootstrap resampling on the blood pressure data to estimate the variability of mean, standard deviation, and 98th percentile.
- It calculates these statistics for each bootstrap sample.
- It calculates the mean, standard deviation, and 98th percentile of blood pressure for the entire population.

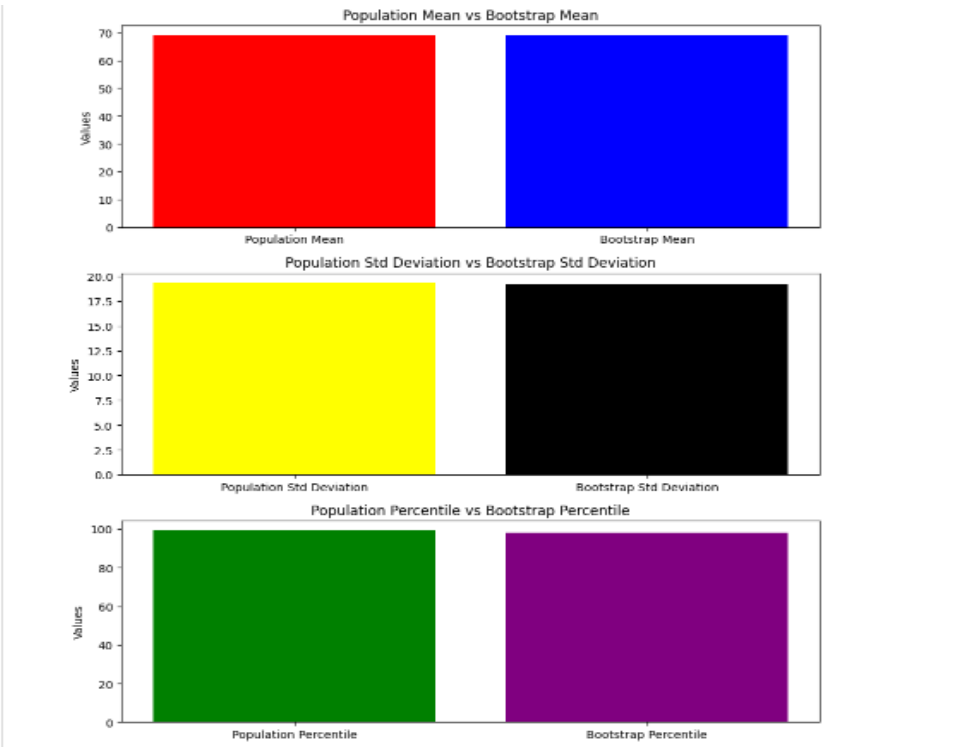
```
#plotting the graphs
fig, axs = plt.subplots(3, 1, figsize=(9, 10))

axs[0].bar(['Population Mean', 'Bootstrap Mean'], [population_mean, mean], color=['red', 'blue'])
axs[0].set_ylabel('Values')
axs[0].set_title('Population Mean vs Bootstrap Mean')

axs[1].bar(['Population Std Deviation', 'Bootstrap Std Deviation'], [population_std_deviation, sd], color=['yellow', 'black'])
axs[1].set_ylabel('Values')
axs[1].set_title('Population Std Deviation vs Bootstrap Std Deviation')

axs[2].bar(['Population Percentile', 'Bootstrap Percentile'], [population_percentile, percentile], color=['green', 'purple'])
axs[2].set_ylabel('Values')
axs[2].set_title('Population Percentile vs Bootstrap Percentile')

plt.tight_layout()
plt.show()
```



The output consists of three bar charts:

- Comparing population mean blood pressure with bootstrap mean blood pressure.
- Comparing population standard deviation of blood pressure with bootstrap standard deviation of blood pressure.
- Comparing population 98th percentile of blood pressure with bootstrap 98th percentile of blood pressure.
- Each chart visualizes the comparison between the corresponding population statistic and the statistic estimated from the bootstrap samples.

Overall, this code provides a comprehensive analysis of the dataset, comparing population statistics with statistics obtained from samples and bootstrap resampling. It helps in understanding the variability and distribution of different variables related to diabetes.