New Section

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc
```

Double-click (or enter) to edit

```
df = pd.read_csv("/content/PDS Assignment2/diabetes.csv")
```

```
# Set the seed for reproducibility
np.random.seed(526)
```

```
# Randomly select 25 observations
sample = df.sample(n=25, random_state=0)
```

```
# Compute the sample statistics
sample_mean_glucose = sample['Glucose'].mean()
sample_max_glucose = sample['Glucose'].max()

# Compute the population statistics
population_mean_glucose = df['Glucose'].mean()
population_max_glucose = df['Glucose'].max()
```

```
# Display the sample and population statistics
print(f"Sample Mean Glucose: {sample_mean_glucose}")
print(f"Sample Max Glucose: {sample_max_glucose}")
print(f"Population Mean Glucose: {population_mean_glucose}")
print(f"Population Max Glucose: {population_max_glucose}")
```

```
Sample Mean Glucose: 120.08
Sample Max Glucose: 199
Population Mean Glucose: 120.89453125
Population Max Glucose: 199
```

```
# Plot the sample and population statistics
fig, ax = plt.subplots()
ax.hist(df['Glucose'], bins=25, alpha=0.5, label='Population')
ax.hist(sample['Glucose'], bins=25, alpha=0.5, label='Sample')
ax.set_xlabel('Glucose')
ax.set_ylabel('Frequency')
ax.legend()
plt.show()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

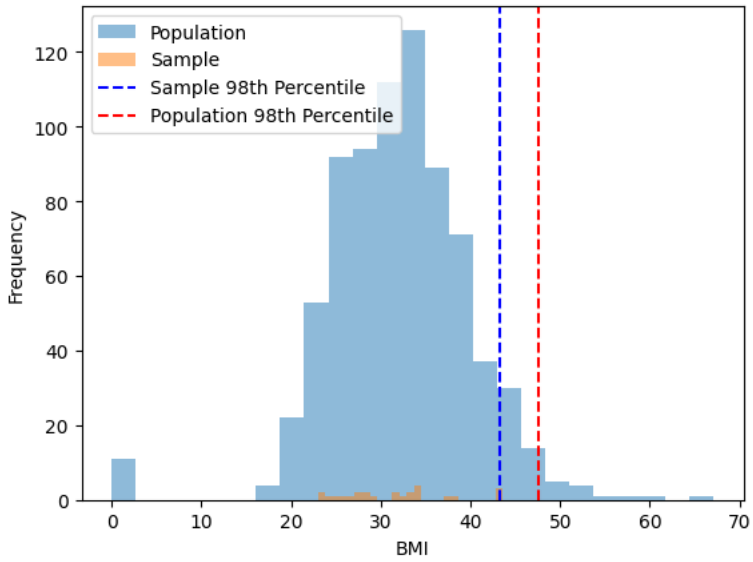
```
# Randomly select 25 observations
sample = df.sample(n=25, random_state=0)

# Compute the 98th percentile of BMI for the sample and the population
sample_percentile_bmi = np.percentile(sample['BMI'], 98)
population_percentile_bmi = np.percentile(df['BMI'], 98)

# Display the 98th percentile of BMI for the sample and the population
print(f"Sample 98th Percentile BMI: {sample_percentile_bmi}")
print(f"Population 98th Percentile BMI: {population_percentile_bmi}")
```

```
Sample 98th Percentile BMI: 43.264
Population 98th Percentile BMI: 47.52599999999996
```

```
# Plot the 98th percentile of BMI for the sample and the population
fig, ax = plt.subplots()
ax.hist(df['BMI'], bins=25, alpha=0.5, label='Population')
ax.hist(sample['BMI'], bins=25, alpha=0.5, label='Sample')
ax.axvline(x=sample_percentile_bmi, color='b', linestyle='--', label='Sample 98th Percentile')
ax.axvline(x=population_percentile_bmi, color='r', linestyle='--', label='Population 98th Percentile')
ax.set_xlabel('BMI')
ax.set_ylabel('Frequency')
ax.legend()
plt.show()
```




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
```

```
#3

# Bootstrap resampling with replacement
sample_bootstrapped = resample(df, replace=True, n_samples=500, random_state=0)
```

```
# Function to compute average mean, standard deviation, and percentile
def compute_statistics(df, percentile=98):
    average_mean = df.mean().mean()
    average_std = df.std().mean()
    average_percentile = np.percentile(df, percentile).mean()
    return average_mean, average_std, average_percentile
```

```
# Compute average mean, standard deviation, and percentile for BloodPressure in the population
average_mean_population, average_std_population, average_percentile_population = compute_statistics(df[['BloodPressure']])

# Compute average mean, standard deviation, and percentile for BloodPressure in the bootstrap samples
average_mean_samples, average_std_samples, average_percentile_samples = compute_statistics(sample_bootstrapped[['BloodPressure']])


# Display the computed statistics
print(f"Population Average Mean BloodPressure: {average_mean_population}")
print(f"Population Average Standard Deviation BloodPressure: {average_std_population}")
print(f"Population Average 98th Percentile BloodPressure: {average_percentile_population}")

print(f"Bootstrap Samples Average Mean BloodPressure: {average_mean_samples}")
print(f"Bootstrap Samples Average Standard Deviation BloodPressure: {average_std_samples}")
print(f"Bootstrap Samples Average 98th Percentile BloodPressure: {average_percentile_samples}")
```

```
Population Average Mean BloodPressure: 69.10546875
Population Average Standard Deviation BloodPressure: 19.355807170644777
Population Average 98th Percentile BloodPressure: 99.31999999999994
Bootstrap Samples Average Mean BloodPressure: 71.19
Bootstrap Samples Average Standard Deviation BloodPressure: 17.60744001516903
Bootstrap Samples Average 98th Percentile BloodPressure: 102.03999999999996
```

```
# Plot the histogram comparison of BloodPressure for the population and the bootstrap samples
fig, ax = plt.subplots()
ax.hist(df['BloodPressure'], bins=30, alpha=0.5, label='Population')
ax.hist(sample_bootstrapped['BloodPressure'], bins=30, alpha=0.5, label='Bootstrap Samples')
ax.set_xlabel('BloodPressure')
ax.set_ylabel('Frequency')
ax.legend()
plt.show()
```