Open in Colab

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
```

```
dataset = Path("__file__").parents[0].joinpath('datasets/Body_Measurements_original.csv')
df = pd.read_csv(dataset)
df
```

| | Gender | HeadCircumference | ShoulderWidth | ChestWidth | Belly | Waist | Hips | ArmLength | Sl |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 40.0 | 18.0 | 20.0 | 18.0 | 14.0 | 22.0 | 22.0 | |
| **1** | 1.0 | 19.0 | 22.0 | 17.0 | 18.0 | 21.0 | 25.0 | 28.0 | |
| **2** | 2.0 | 21.0 | 18.0 | 16.0 | 14.0 | 10.0 | 15.0 | 21.0 | |
| **3** | 1.0 | 20.0 | 20.0 | 18.0 | 11.0 | 19.0 | 14.0 | 24.0 | |
| **4** | 2.0 | 16.0 | 14.0 | 18.0 | 13.0 | 11.0 | 30.0 | 25.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **994** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **995** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **996** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **997** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **998** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

999 rows × 13 columns

```
df.columns
```

```
Index(['Gender', 'HeadCircumference', 'ShoulderWidth', 'ChestWidth', 'Belly',
       'Waist', 'Hips', 'ArmLength', 'ShoulderToWaist', 'WaistToKnee',
       'LegLength', 'TotalHeight', 'Size'],
      dtype='object')
```

```
df
```

| | Gender | HeadCircumference | ShoulderWidth | ChestWidth | Belly | Waist | Hips | ArmLength | Sl |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 40.0 | 18.0 | 20.0 | 18.0 | 14.0 | 22.0 | 22.0 | |
| **1** | 1.0 | 19.0 | 22.0 | 17.0 | 18.0 | 21.0 | 25.0 | 28.0 | |
| **2** | 2.0 | 21.0 | 18.0 | 16.0 | 14.0 | 10.0 | 15.0 | 21.0 | |
| **3** | 1.0 | 20.0 | 20.0 | 18.0 | 11.0 | 19.0 | 14.0 | 24.0 | |
| **4** | 2.0 | 16.0 | 14.0 | 18.0 | 13.0 | 11.0 | 30.0 | 25.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **994** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **995** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **996** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **997** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **998** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

999 rows × 13 columns

```
df = df.iloc[0:399, :]
df
```

| | Gender | HeadCircumference | ShoulderWidth | ChestWidth | Belly | Waist | Hips | ArmLength | Sl |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 40.0 | 18.0 | 20.0 | 18.0 | 14.0 | 22.0 | 22.0 | |
| **1** | 1.0 | 19.0 | 22.0 | 17.0 | 18.0 | 21.0 | 25.0 | 28.0 | |
| **2** | 2.0 | 21.0 | 18.0 | 16.0 | 14.0 | 10.0 | 15.0 | 21.0 | |
| **3** | 1.0 | 20.0 | 20.0 | 18.0 | 11.0 | 19.0 | 14.0 | 24.0 | |
| **4** | 2.0 | 16.0 | 14.0 | 18.0 | 13.0 | 11.0 | 30.0 | 25.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **394** | 1.0 | 18.0 | 18.0 | 9.0 | 19.0 | 12.0 | 9.0 | 14.0 | |
| **395** | 1.0 | 20.0 | 12.0 | 9.0 | 10.0 | 23.0 | 10.0 | 12.0 | |
| **396** | 1.0 | 21.0 | 13.0 | 11.0 | 10.0 | 21.0 | 10.0 | 13.0 | |
| **397** | 1.0 | 20.0 | 17.0 | 11.0 | 22.0 | 22.0 | 22.0 | 17.0 | |
| **398** | 1.0 | 20.0 | 9.0 | 9.0 | 20.0 | 20.0 | 10.0 | 14.0 | |

399 rows × 13 columns

```
df.describe()
```

| | Gender | HeadCircumference | ShoulderWidth | ChestWidth | Belly | Waist | |
|---|---|---|---|---|---|---|---|
| **count** | 399.000000 | 399.000000 | 399.000000 | 399.000000 | 399.000000 | 399.000000 | 399 |
| **mean** | 1.350877 | 20.829574 | 16.117794 | 16.761905 | 19.150376 | 20.974937 | 21 |
| **std** | 0.477844 | 4.923641 | 5.400415 | 6.051367 | 13.291379 | 10.104200 | 9 |
| **min** | 1.000000 | 9.000000 | 5.000000 | 6.000000 | 5.000000 | 6.000000 | 7 |
| **25%** | 1.000000 | 19.000000 | 14.000000 | 12.000000 | 12.000000 | 14.000000 | 15 |
| **50%** | 1.000000 | 21.000000 | 17.000000 | 16.000000 | 17.000000 | 20.000000 | 20 |
| **75%** | 2.000000 | 23.000000 | 19.000000 | 20.000000 | 23.000000 | 25.000000 | 26 |
| **max** | 2.000000 | 80.000000 | 87.000000 | 38.000000 | 213.000000 | 91.000000 | 63 |

```
df.shape
```

```
(399, 13)
```

```
df.info()
```

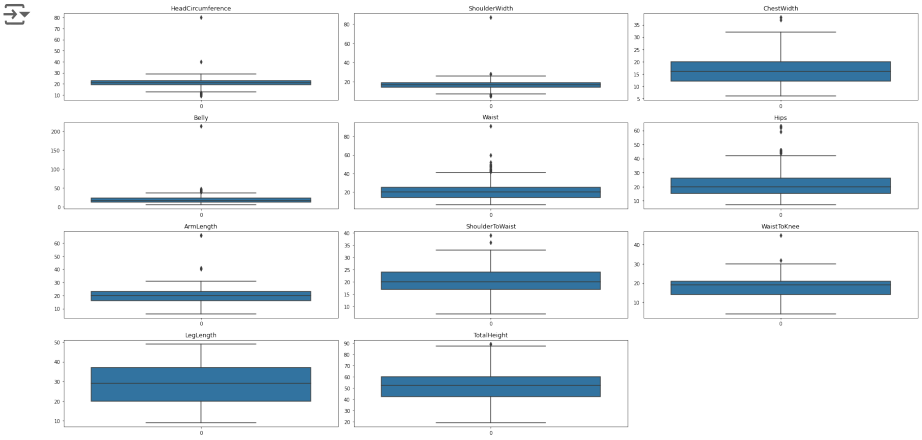
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 399 entries, 0 to 398
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             399 non-null    float64
 1   HeadCircumference  399 non-null    float64
 2   ShoulderWidth      399 non-null    float64
 3   ChestWidth         399 non-null    float64
 4   Belly              399 non-null    float64
 5   Waist              399 non-null    float64
 6   Hips               399 non-null    float64
 7   ArmLength          399 non-null    float64
 8   ShoulderToWaist    399 non-null    float64
 9   WaistToKnee        399 non-null    float64
 10  LegLength          399 non-null    float64
 11  TotalHeight        399 non-null    float64
 12  Size               399 non-null    object
dtypes: float64(12), object(1)
memory usage: 40.6+ KB
```

```
for i in df.columns :
  print(f'{i} => {df[i].unique()}\n-------------***-----------------\n')
```

```
Gender => [1. 2.]
-------------***-----------------

HeadCircumference => [40. 19. 21. 20. 16. 17. 25. 18. 15. 23. 24. 80. 28. 29. 14. 22. 27. 26.
 12. 13. 11. 10.  9.]
-------------***-----------------

ShoulderWidth => [18. 22. 20. 14. 19. 17. 15. 16. 28. 21.  9. 13.  7. 11. 10.  8. 12. 23.
  6.  5. 87. 26.]
-------------***-----------------

ChestWidth => [20. 17. 16. 18. 19. 28. 21. 22. 14. 15. 10.  8. 13. 11.  7. 12.  9. 23.
  6. 26. 27. 31. 37. 38. 32. 29. 25. 30.]
-------------***-----------------

Belly => [ 18.  14.  11.  13.  17.  15.  12.  16.   9.  19.  10.   8.   6.  22.
  21.   7.  20. 213.  23.  24.   5.  29.  30.  34.  37.  27.  28.  25.
  32.  36.  26.  44.  42.  33.  46.  41.  45.  47.  40.  43.  35.  31.]
```

```
-------------***-----------------

Waist => [14. 21. 10. 19. 11. 16. 12. 23. 22. 91. 18. 20. 24. 25. 17. 15. 13.  9.
  7.  8. 50. 49. 36. 26. 60. 31. 29. 52. 40. 27.  6. 30. 32. 37. 41. 28.
 33. 34. 42. 38. 48. 47. 45. 39. 43. 44. 35.]
-------------***-----------------

Hips => [22. 25. 15. 14. 30. 18. 28. 27. 17. 21. 19. 23. 24. 20. 35. 16. 10.  9.
 11. 13.  8. 12. 36. 44.  7. 34. 26. 37. 38. 32. 29. 31. 42. 46. 39. 63.
 45. 62. 41. 40. 59.]
-------------***-----------------

ArmLength => [22. 28. 21. 24. 25. 20. 23. 19. 15. 16. 17. 18. 11. 14. 10.  7. 26.  9.
 13. 12. 40. 41. 66.  8. 30. 27.  6. 29. 31.]
-------------***-----------------

ShoulderToWaist => [25. 23. 18. 21. 22. 24. 19. 26. 17. 20. 27. 28. 16. 15.  8. 11. 39.  9.
 10. 13. 12. 36. 33. 14.  7. 29. 30. 31. 32.]
-------------***-----------------

WaistToKnee => [25. 14. 20. 32. 21. 19. 18. 17. 23. 16. 11. 12. 13. 22.  8.  7. 24. 26.
  9. 15. 27. 30. 29.  6. 10. 45. 28.  4.]
-------------***-----------------

LegLength => [22. 20. 18. 21. 13. 19. 17. 24. 15.  9. 30. 31. 28. 39. 27. 23. 14. 38.
 37. 46. 34. 26. 41. 33. 42. 36. 16. 29. 49. 40. 32. 45. 25. 44. 12. 43.
 11. 35.]
-------------***-----------------

TotalHeight => [52. 56. 53. 45. 47. 60. 49. 58. 40. 55. 50. 59. 51. 57. 48. 42. 44. 54.
 31. 30. 33. 39. 25. 72. 34. 38. 43. 23. 62. 24. 61. 37. 75. 73. 46. 41.
 63. 64. 69. 67. 68. 66. 32. 35. 20. 21. 36. 70. 22. 80. 79. 71. 19. 27.
 74. 85. 86. 89. 78. 82. 87. 65.]
-------------***-----------------

Size => ['L' 'M' 'S' 'XS']
-------------***-----------------
```

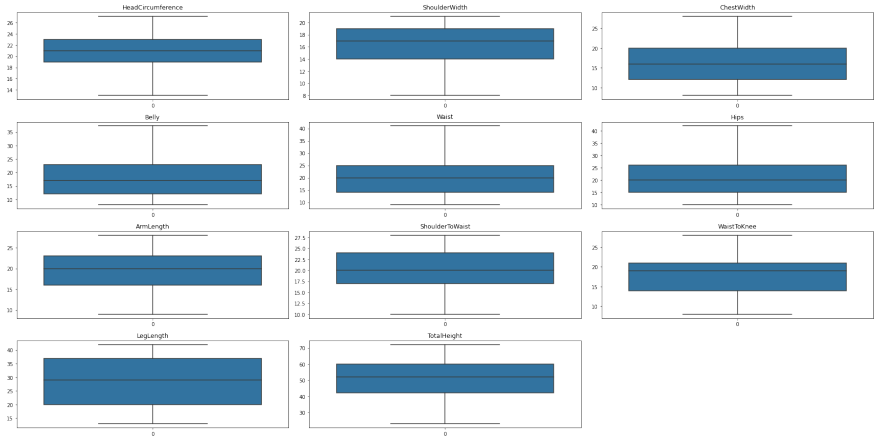
```
# Selecting only continuous variables
continuous_vars = ['HeadCircumference', 'ShoulderWidth', 'ChestWidth', 'Belly',
       'Waist', 'Hips', 'ArmLength', 'ShoulderToWaist', 'WaistToKnee',
       'LegLength', 'TotalHeight']

# Plot boxplots for continuous variables to detect outliers
plt.figure(figsize=(24, 12))
for i, var in enumerate(continuous_vars):
    plt.subplot(4, 3, i+1)
    sns.boxplot(data=df[var])
    plt.title(var)
plt.tight_layout()
plt.show()
```



```
#Remove or clip outliers between 5th and 95th percentiles
for var in continuous_vars:
    lower_bound = np.percentile(df[var], 5)
    upper_bound = np.percentile(df[var], 95)
    df[var] = np.clip(df[var], lower_bound, upper_bound)
```

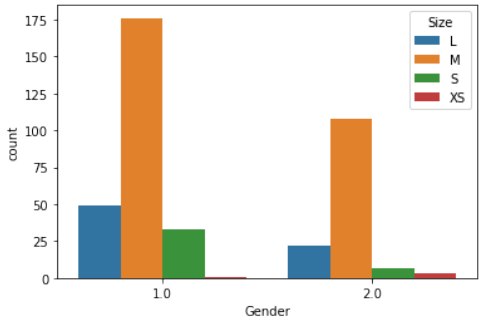
```
<ipython-input-11-072ddf0e7370>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[var] = np.clip(df[var], lower_bound, upper_bound)
```

```
# Selecting only continuous variables
continuous_vars = ['HeadCircumference', 'ShoulderWidth', 'ChestWidth', 'Belly',
       'Waist', 'Hips', 'ArmLength', 'ShoulderToWaist', 'WaistToKnee',
       'LegLength', 'TotalHeight']

# Plot boxplots for continuous variables to detect outliers
plt.figure(figsize=(24, 12))
for i, var in enumerate(continuous_vars):
    plt.subplot(4, 3, i+1)
    sns.boxplot(data=df[var])
    plt.title(var)
plt.tight_layout()
plt.show()
```




```
sns.countplot(data=df, x = 'Gender', hue = 'Size')
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
data = df
X = data.drop(columns=['Size'])  # Features
y = data['Size']  # Target
print(X, y)
```

```
     Gender  HeadCircumference  ShoulderWidth  ChestWidth  Belly  Waist  Hips  \
0       1.0               27.1           18.0        20.0   18.0   14.0  22.0
1       1.0               19.0           21.0        17.0   18.0   21.0  25.0
2       2.0               21.0           18.0        16.0   14.0   10.0  15.0
3       1.0               20.0           20.0        18.0   11.0   19.0  14.0
4       2.0               16.0           14.0        18.0   13.0   11.0  30.0
..      ...                ...            ...         ...    ...    ...   ...
394     1.0               18.0           18.0         9.0   19.0   12.0  10.0
395     1.0               20.0           12.0         9.0   10.0   23.0  10.0
396     1.0               21.0           13.0        11.0   10.0   21.0  10.0
397     1.0               20.0           17.0        11.0   22.0   22.0  22.0
398     1.0               20.0            9.0         9.0   20.0   20.0  10.0

     ArmLength  ShoulderToWaist  WaistToKnee  LegLength  TotalHeight
0         22.0             25.0         25.0       22.0         52.0
1         28.0             23.0         25.0       20.0         56.0
2         21.0             18.0         14.0       18.0         53.0
3         24.0             21.0         20.0       21.0         45.0
4         25.0             22.0         28.0       13.0         47.0
..         ...              ...          ...        ...          ...
394       14.0             11.0         13.0       21.0         42.0
395       12.0             17.0         12.0       22.0         45.0
396       13.0             17.0         12.0       22.0         45.0
397       17.0             12.0         12.0       22.0         40.0
398       14.0             11.0          9.0       17.0         37.0

[399 rows x 12 columns] 0      L
1      M
2      L
3      M
4      M
      ..
394    L
395    M
396    M
397    S
398    M
Name: Size, Length: 399, dtype: object
```

```
le = LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])  # Label encoding for 'Gender'
```

```
#Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train, X_test, y_train, y_test)
```

```
     Gender  HeadCircumference  ShoulderWidth  ChestWidth  Belly  Waist  Hips  \
3         0               20.0           20.0        18.0   11.0   19.0  14.0
18        0               20.0           16.0        28.0   18.0   17.0  23.0
377       1               19.0            9.0        16.0   20.0   18.0  21.0
248       1               13.0           21.0        15.0   13.0   12.0  14.0
177       1               21.0           14.0        15.0   21.0   14.0  12.0
..      ...                ...            ...         ...    ...    ...   ...
71        0               15.0            8.0         8.0    8.0    9.0  10.0
106       0               23.0           16.0        11.0   10.0   14.0  14.0
270       0               27.1           20.0        28.0   30.0   32.0  38.0
348       0               22.0           14.0        16.0   34.0   20.0  38.0
102       1               20.0           19.0        13.0   12.0   16.0  17.0

     ArmLength  ShoulderToWaist  WaistToKnee  LegLength  TotalHeight
3         24.0             21.0         20.0       21.0         45.0
18        16.0             27.0         23.0       13.0         48.0
377       19.0             11.0          9.0       16.0         35.0
248        9.0             10.0          8.0       13.0         23.0
177       12.0             16.0         11.0       25.0         30.0
..         ...              ...          ...        ...          ...
71        10.0             11.0          8.0       17.0         23.0
106       25.0             25.0         20.0       39.0         53.0
270       28.0             28.0         27.0       41.0         72.1
348       24.0             19.0         19.0       35.0         64.0
102        9.0             27.0         19.0       39.0         43.0

[319 rows x 12 columns]      Gender  HeadCircumference  ShoulderWidth  ChestWidth  Belly  Waist  Hips  \
198       0               21.0           15.0        13.0   11.0    9.0  13.0
349       1               21.0           18.0        19.0   34.0   37.0  39.0
33        0               24.0           19.0        16.0   15.0   19.0  21.0
208       0               27.0           17.0        20.0   18.0   21.0  22.0
93        0               16.0           14.0        12.0   10.0   13.0  19.0
..      ...                ...            ...         ...    ...    ...   ...
249       1               13.0            8.0        16.0   13.0   12.0  15.0
```

```
225       0               22.0           19.0        15.0   14.0   14.0  15.0
368       0               22.0           17.0        16.0   35.0   36.0  37.0
175       1               22.0           17.0        20.0   16.0   20.0  20.0
285       0               24.0           15.0        20.0   22.0   21.0  20.0

     ArmLength  ShoulderToWaist  WaistToKnee  LegLength  TotalHeight
198       21.0             17.0         20.0       42.0         42.0
349       22.0             17.0         15.0       33.0         59.0
33        17.0             24.0         12.0       28.0         60.0
208       23.0             22.0         19.0       39.0         55.0
93        15.0             20.0         11.0       22.0         33.0
..         ...              ...          ...        ...          ...
249       10.0             10.0          8.0       13.0         23.0
225       24.0             26.0         22.0       42.0         72.0
368       25.0             22.0         23.0       36.0         67.0
175       20.0             20.0         19.0       30.0         50.0
285       21.0             22.0         16.0       30.0         61.0

[80 rows x 12 columns] 3      M
18     M
377    M
248    M
177    M
      ..
```

```
#Initialize the model
rf = RandomForestClassifier(random_state=42)
gnb = GaussianNB()
mnb = MultinomialNB()
```

```
#Training the Models
rf.fit(X_train, y_train)
gnb.fit(X_train, y_train)
mnb.fit(X_train, y_train)
```

```
MultinomialNB()
```

```
#Predictions
y_pred_rf = rf.predict(X_test)
y_pred_gnb = gnb.predict(X_test)
y_pred_mnb = mnb.predict(X_test)
```

```
#Evaluation
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", accuracy_rf * 100, "%")
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
print("Gaussian Naive Bayes Accuracy:", accuracy_gnb * 100, "%")
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
print("Multinomial Naive Bayes Accuracy:", accuracy_mnb * 100, "%")
```

```
Random Forest Accuracy: 96.25 %
Gaussian Naive Bayes Accuracy: 93.75 %
Multinomial Naive Bayes Accuracy: 53.75 %
```

```
rf.predict([X.iloc[6]])
```

```
array(['S'], dtype=object)
```

```
y.iloc[6]
```

```
'S'
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.