Name - Ningaraddi Raddi

SapId -500110910

In [4]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt
```

In [5]:
```
# Google Colab file upload
from google.colab import files

# Upload the CSV file
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving bodyPerformance.csv to bodyPerformance (3).csv
```

In [6]:
```
# Assuming the file name is bodyPerformance.csv after upload, adjust if needed
data = pd.read_csv('bodyPerformance.csv')

# Convert all columns to numeric, coercing errors to NaN
data = data.apply(pd.to_numeric, errors='coerce')
```

In [7]:
```
### 1.1 Measure of Central Tendency
# Mean
mean = data.mean()

# Geometric Mean
geometric_mean = stats.gmean(data.select_dtypes(include=[np.number]), axis=0)

# Select only numeric columns and filter out non-positive values for harmonic mean
numeric_data = data.select_dtypes(include=[np.number])
positive_data = numeric_data[numeric_data > 0].dropna()

# Calculate the harmonic mean
harmonic_mean = stats.hmean(positive_data, axis=0)

# Mode
mode = data.mode().iloc[0]

# Median
median = data.median()

# Print central tendency measures
print(f"Mean:\n{mean}\n")
print(f"Geometric Mean:\n{geometric_mean}\n")
print(f"Harmonic Mean:\n{harmonic_mean}\n")
print(f"Mode:\n{mode}\n")
print(f"Median:\n{median}\n")
```

```
Mean:
age                         36.775106
gender                            NaN
height_cm                 168.559807
weight_kg                  67.447316
body fat_%                 23.240165
diastolic                  78.796842
systolic                  130.234817
gripForce                  36.963877
sit and bend forward_cm    15.209268
sit-ups counts             39.771224
broad jump_cm             190.129627
class                             NaN
dtype: float64

Geometric Mean:
[ 34.4243199           nan 168.34732191  66.39359291  22.05345026
   0.           0.           0.                 nan   0.
   0.                 nan]

Harmonic Mean:
[nan nan nan nan nan nan nan nan nan nan nan nan]

Mode:
age                         21.0
gender                       NaN
height_cm                 170.0
weight_kg                  70.5
body fat_%                 23.1
diastolic                  80.0
systolic                  120.0
gripForce                  43.1
sit and bend forward_cm    20.0
sit-ups counts             45.0
broad jump_cm             211.0
class                        NaN
Name: 0, dtype: float64

Median:
age                         32.0
gender                       NaN
height_cm                 169.2
weight_kg                  67.4
body fat_%                 22.8
diastolic                  79.0
systolic                  130.0
gripForce                  37.9
sit and bend forward_cm    16.2
sit-ups counts             41.0
broad jump_cm             193.0
class                        NaN
dtype: float64
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:197: RuntimeWarni
ng: invalid value encountered in log
  log_a = np.log(a)
```

In [8]:

```
### 1.2 Measure of Dispersion
# Variance
variance = data.var()

# Standard Deviation
std_dev = data.std()
```

```
# Skewness
skewness = data.skew()

# Inter Quartile Range (IQR)
iqr = data.quantile(0.75) - data.quantile(0.25)

# Range
data_range = data.max() - data.min()

# Calculate Mean Absolute Deviation (MAD) manually
mad_values = data.select_dtypes(include=[np.number]).apply(lambda x: np.mean(np.abs

# Print dispersion measures
print(f"Variance:\n{variance}\n")
print(f"Standard Deviation:\n{std_dev}\n")
print(f"Skewness:\n{skewness}\n")
print(f"IQR:\n{iqr}\n")
print(f"Range:\n{data_range}\n")
print(f"MAD:\n{mad_values}\n")
```

```
Variance:
age                         185.658051
gender                             NaN
height_cm                    71.007293
weight_kg                   142.794526
body fat_%                   52.661786
diastolic                   115.391275
systolic                    216.500428
gripForce                   112.887736
sit and bend forward_cm      71.515386
sit-ups counts              203.824115
broad jump_cm              1589.457435
class                              NaN
dtype: float64

Standard Deviation:
age                         13.625639
gender                            NaN
height_cm                    8.426583
weight_kg                   11.949666
body fat_%                   7.256844
diastolic                   10.742033
systolic                    14.713954
gripForce                   10.624864
sit and bend forward_cm      8.456677
sit-ups counts              14.276698
broad jump_cm               39.868000
class                             NaN
dtype: float64

Skewness:
age                          0.599896
gender                            NaN
height_cm                   -0.186882
weight_kg                    0.349805
body fat_%                   0.361132
diastolic                   -0.159637
systolic                    -0.048654
gripForce                    0.018456
sit and bend forward_cm      0.785492
sit-ups counts              -0.467830
broad jump_cm               -0.422623
class                             NaN
dtype: float64

IQR:
age                         23.0
gender                       NaN
height_cm                   12.4
weight_kg                   17.1
body fat_%                  10.0
diastolic                   15.0
systolic                    21.0
gripForce                   17.7
sit and bend forward_cm      9.8
sit-ups counts              20.0
broad jump_cm               59.0
class                        NaN
dtype: float64

Range:
age                         43.0
gender                       NaN
height_cm                   68.8
```

```
weight_kg                 111.8
body fat_%                 75.4
diastolic                 156.2
systolic                  201.0
gripForce                  70.5
sit and bend forward_cm   238.0
sit-ups counts             80.0
broad jump_cm             303.0
class                       NaN
dtype: float64

MAD:
age                     11.844362
gender                        NaN
height_cm                6.919084
weight_kg                9.680199
body fat_%               5.833442
diastolic                8.651310
systolic                12.026424
gripForce                9.068306
sit and bend forward_cm  6.268510
sit-ups counts          11.571289
broad jump_cm           32.726099
class                         NaN
dtype: float64
```

In [9]:
```
### 1.3 Correlation Between Features
# Correlation matrix
correlation = data.corr()
print(f"Correlation Matrix:\n{correlation}\n")
```

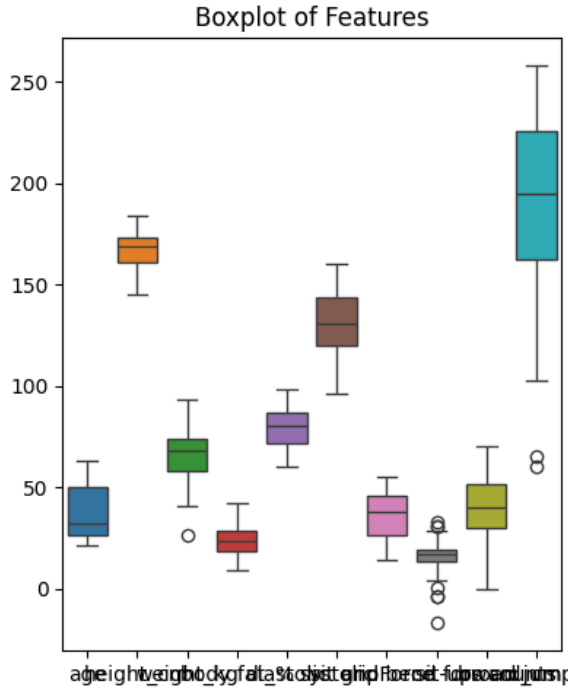
```
Correlation Matrix:
                              age  gender  height_cm  weight_kg  body fat_%  \
age                      1.000000     NaN  -0.293980  -0.099966    0.242302
gender                        NaN     NaN        NaN        NaN         NaN
height_cm               -0.293980     NaN   1.000000   0.734909   -0.515440
weight_kg               -0.099966     NaN   0.734909   1.000000   -0.084065
body fat_%               0.242302     NaN  -0.515440  -0.084065    1.000000
diastolic                0.158508     NaN   0.145933   0.262317    0.048059
systolic                 0.211167     NaN   0.210186   0.338943   -0.030376
gripForce               -0.179583     NaN   0.735024   0.700119   -0.541788
sit and bend forward_cm -0.070033     NaN  -0.221970  -0.296249   -0.071225
sit-ups counts          -0.544581     NaN   0.500424   0.294899   -0.608912
broad jump_cm           -0.435172     NaN   0.674589   0.479564   -0.673273
class                         NaN     NaN        NaN        NaN         NaN

                         diastolic  systolic  gripForce  \
age                       0.158508  0.211167  -0.179583
gender                         NaN       NaN        NaN
height_cm                 0.145933  0.210186   0.735024
weight_kg                 0.262317  0.338943   0.700119
body fat_%                0.048059 -0.030376  -0.541788
diastolic                 1.000000  0.676309   0.202062
systolic                  0.676309  1.000000   0.286012
gripForce                 0.202062  0.286012   1.000000
sit and bend forward_cm  -0.072098 -0.082434  -0.112577
sit-ups counts            0.016547  0.056276   0.576669
broad jump_cm             0.097243  0.152894   0.746853
class                          NaN       NaN        NaN

                         sit and bend forward_cm  sit-ups counts  \
age                                    -0.070033       -0.544581
gender                                       NaN             NaN
height_cm                              -0.221970        0.500424
weight_kg                              -0.296249        0.294899
body fat_%                             -0.071225       -0.608912
diastolic                              -0.072098        0.016547
systolic                               -0.082434        0.056276
gripForce                              -0.112577        0.576669
sit and bend forward_cm                 1.000000        0.177153
sit-ups counts                          0.177153        1.000000
broad jump_cm                           0.026487        0.748273
class                                        NaN             NaN

                         broad jump_cm  class
age                          -0.435172    NaN
gender                             NaN    NaN
height_cm                     0.674589    NaN
weight_kg                     0.479564    NaN
body fat_%                   -0.673273    NaN
diastolic                     0.097243    NaN
systolic                      0.152894    NaN
gripForce                     0.746853    NaN
sit and bend forward_cm       0.026487    NaN
sit-ups counts                0.748273    NaN
broad jump_cm                 1.000000    NaN
class                              NaN    NaN
```

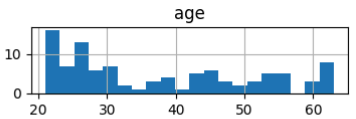
In [12]:

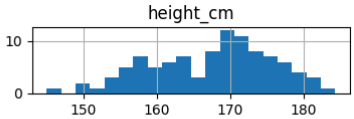
```
# Convert all columns to numeric, coercing errors to NaN
data = data.apply(pd.to_numeric, errors='coerce')

# Select only numeric columns
numeric_data = data.select_dtypes(include=[np.number])
```

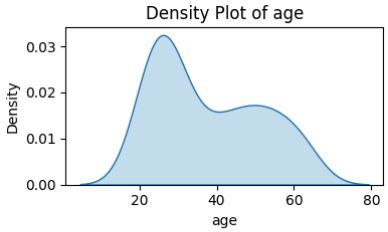
```
# Drop columns with all NaN values
numeric_data = numeric_data.dropna(axis=1, how='all')

plt.figure(figsize=(15, 12))

# Boxplot
plt.subplot(2, 3, 1)
sns.boxplot(data=numeric_data.sample(n=100, random_state=1))  # Sample data for fas
plt.title('Boxplot of Features')

# Histograms
plt.subplot(2, 3, 2)
numeric_data.sample(n=100, random_state=1).hist(bins=20, figsize=(12, 6))  # Reduce
plt.title('Histograms of Features')

# Density Plots
for i, col in enumerate(numeric_data.columns[:3]):  # Limit to first 3 columns
    plt.subplot(2, 3, 3 + i)
    if numeric_data[col].dropna().shape[0] > 1:  # Ensure there are enough data poi
        sns.kdeplot(numeric_data[col].dropna().sample(n=100, random_state=1), fill=
    plt.title(f'Density Plot of {col}')

# Scatterplot Example: Choose any two features to compare (Feature1 vs Feature2)
plt.subplot(2, 3, 4)
sns.scatterplot(x=numeric_data.iloc[:, 0].sample(n=100, random_state=1),
                y=numeric_data.iloc[:, 1].sample(n=100, random_state=1))  # Sample
plt.title('Scatterplot of Two Features')

# Bar chart of categorical data if applicable
plt.subplot(2, 3, 5)
if 'diastolic' in data.columns:  # Replace with an actual categorical feature name
    sns.countplot(data=data['diastolic'].dropna().sample(n=100, random_state=1))  #
    plt.title('Bar Chart of diastolic')

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

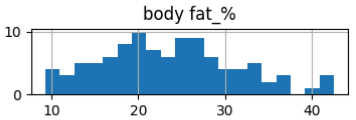
```
<ipython-input-12-a2da8589cdfb>:24: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases la
ter; explicitly call ax.remove() as needed.
  plt.subplot(2, 3, 3 + i)
<ipython-input-12-a2da8589cdfb>:24: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases la
ter; explicitly call ax.remove() as needed.
  plt.subplot(2, 3, 3 + i)
<ipython-input-12-a2da8589cdfb>:24: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases la
ter; explicitly call ax.remove() as needed.
  plt.subplot(2, 3, 3 + i)
```

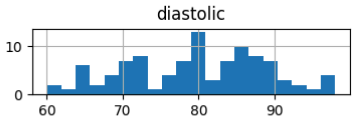

Boxplot of Features
250
200
150
100
50
0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

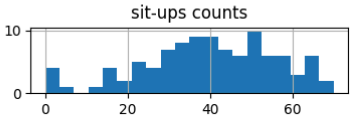

age
10
0
20
30
40
50
60

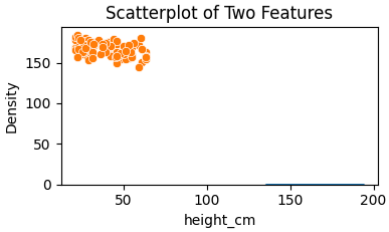

height_cm
10
0
150
160
170
180

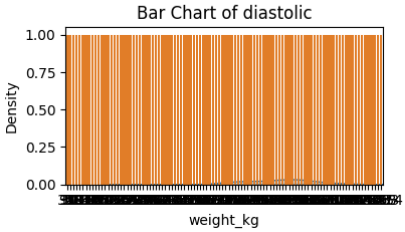

Density Plot of age
Density
0.03
0.02
0.01
0.00
20
40
60
80
age


body fat_%
10
0
10
20
30
40


diastolic
10
0
60
70
80
90


sit-ups counts
10
0
0
20
40
60


Scatterplot of Two Features
Density
150
100
50
0
50
100
150
200
height_cm


Bar Chart of diastolic
Density
1.00
0.75
0.50
0.25
0.00
weight_kg