

START MY PROJECT FOR (Practical_Machine_Learning_and_Data_Exploration):

Section 1 – Project Introduction & Loading the Dataset

Project: Body Performance Data Analysis

In this project, the dataset used is called bodyPerformance.csv.

The main steps are:

- Load the dataset
- Explore the data
- Fix missing values
- Detect and handle outliers
- Create visualizations

Note: Dataset link from Kaggle:

- <https://www.kaggle.com/datasets/kukuroo3/body-performance-data>

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 sns.set(style="whitegrid")
7
8 df = pd.read_csv("bodyPerformance.csv")
9
10 df.head()
```

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
0	27.0	M	172.3	75.24	21.3	80.0	130.0	54.9	18.4	60.0	217.0	C
1	25.0	M	165.0	55.80	15.7	77.0	126.0	36.4	16.3	53.0	229.0	A
2	31.0	M	179.6	78.00	20.1	92.0	152.0	44.8	12.0	49.0	181.0	C
3	32.0	M	174.5	71.10	18.4	76.0	147.0	41.4	15.2	53.0	219.0	B
4	28.0	M	173.8	67.70	17.1	70.0	127.0	43.5	27.1	45.0	217.0	B

Next steps: [Generate code with df](#) [New interactive sheet](#)

Section 2 – Explore the Data

In this part, check basic information about the dataset:

- Number of rows and columns
- Column names and data types
- Simple statistics
- Missing values

```
1 print("Dataset shape:", df.shape)
2
3 print("\nData info:")
4 print(df.info())
5
6 print("\nStatistics:")
7 print(df.describe())
8
9 print("\nMissing values per column:")
10 print(df.isnull().sum())
```

```
3 weight_kg      13393 non-null float64
4 body fat_%     13393 non-null float64
5 diastolic      13393 non-null float64
6 systolic      13393 non-null float64
7 gripForce     13393 non-null float64
8 sit and bend forward_cm 13393 non-null float64
9 sit-ups counts 13393 non-null float64
10 broad jump_cm 13393 non-null float64
11 class         13393 non-null object
dtypes: float64(10), object(2)
memory usage: 1.2+ MB
None

Statistics:
      age      height_cm      weight_kg      body fat_%      diastolic \
count 13393.000000 13393.000000 13393.000000 13393.000000 13393.000000
mean   36.775106   168.559007   67.447316   23.240165   78.796842
std     13.625639    8.426583   11.949666    7.256844   10.742033
min     21.000000   125.000000   26.300000    3.000000    0.000000
25%     25.000000   162.000000   58.200000   18.000000   71.000000
50%     32.000000   169.200000   67.400000   22.000000   79.000000
75%     48.000000   174.800000   75.300000   28.000000   86.000000
max     64.000000   193.800000   138.100000   78.400000   156.200000

      systolic      gripForce      sit and bend forward_cm      sit-ups counts \
count 13393.000000 13393.000000 13393.000000 13393.000000
mean   130.234817   36.963877    15.209268   39.771224
std     14.713954   10.624864    8.456677   14.276698
min      0.000000    0.000000   -25.000000    0.000000
25%    120.000000   27.500000   10.900000   30.000000
50%    130.000000   37.900000   16.200000   41.000000
75%    141.000000   45.200000   20.700000   50.000000
max    201.000000   70.500000   213.000000   80.000000

      broad jump_cm
count 13393.000000
mean   198.129627
std     39.866000
min      0.000000
25%    162.000000
50%    193.000000
75%    221.000000
max    303.000000
```

```
Missing values per column:
age      0
gender   0
height_cm 0
weight_kg 0
body fat_% 0
diastolic 0
systolic  0
gripForce 0
sit and bend forward_cm 0
sit-ups counts 0
broad jump_cm 0
class    0
dtype: int64
```

Section 3 – Fix Missing Values

In this part, Steps as follow:

1. Check the percentage of missing values.
2. Drop any column with too many missing values.
3. Fill the remaining missing values using the median.

```
1 # Missing percentage
2 missing_pct = df.isnull().mean() * 100
3 print(missing_pct)
4
5 # Drop columns with more than 30% missing
6 cols_to_drop = missing_pct[missing_pct > 30].index
7 df = df.drop(columns=cols_to_drop)
8
9 # Fill missing numeric values with median
10 numeric_cols = df.select_dtypes(include=["float64", "int64"]).columns
11
12 for col in numeric_cols:
13     df[col] = df[col].fillna(df[col].median())
14
15 print("\nMissing values after filling:")
16 print(df.isnull().sum())
```

```
age      0.0
gender   0.0
height_cm 0.0
weight_kg 0.0
body fat_% 0.0
diastolic 0.0
systolic  0.0
gripForce 0.0
sit and bend forward_cm 0.0
sit-ups counts 0.0
broad jump_cm 0.0
class    0.0
dtype: float64

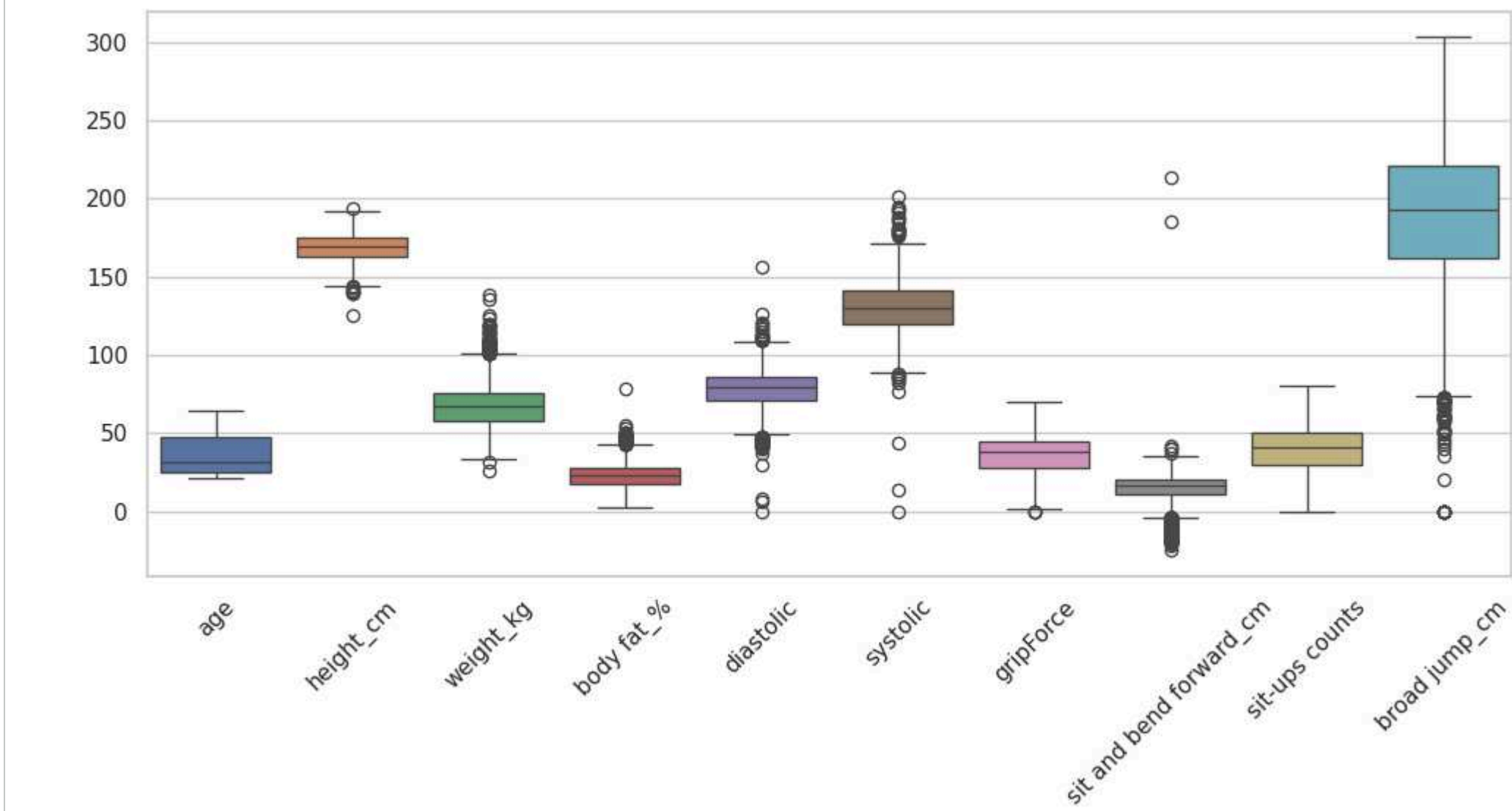
Missing values after filling:
age      0
gender   0
height_cm 0
weight_kg 0
body fat_% 0
diastolic 0
systolic  0
gripForce 0
sit and bend forward_cm 0
sit-ups counts 0
broad jump_cm 0
class    0
dtype: int64
```

Section 4 – Detect Outliers

In this part, detect outliers using:

- Boxplots
- IQR method
- Z-score method

```
1 #Boxplots
2 plt.figure(figsize=(12,5))
3 sns.boxplot(data=df[numeric_cols])
4 plt.xticks(rotation=45)
5 plt.show()
```



1 #IQR method

2 def outliers_iqr(column):

3 Q1 = column.quantile(0.25)

4 Q3 = column.quantile(0.75)

5 IQR = Q3 - Q1

6 low = Q1 - 1.5 * IQR

7 high = Q3 + 1.5 * IQR

8 return (column < low) | (column > high)

9

10 data_iqr = df.copy()

11

12 for col in numeric_cols:

13 data_iqr[col + "_iqr_outlier"] = outliers_iqr(data_iqr[col])

14

15 data_iqr[data_iqr.filter(like="_iqr_outlier").any(axis=1)].head()

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	...	age_iqr_outlier	height_cm_iqr_outlier	weight_kg_iqr_outlier	body fat_%_iqr_outlier	diastolic_iqr_outlier	systolic_iqr_outlier	gripForce_iqr_outlier	sit and bend forward_cm_iqr_outlier	sit-ups counts_iqr_outlier	broad jump_cm_iqr_outlier
47	26.0	M	172.1	66.10	24.9	62.0	102.0	19.9	-9.9	44.0	...	False	False	False	False	False	False	False	True	False	False
58	42.0	M	174.1	103.10	30.5	78.0	123.0	50.7	3.6	25.0	...	False	False	True	False	False	False	False	False	False	False
75	22.0	M	171.9	78.46	22.9	64.0	116.0	43.6	-9.3	48.0	...	False	False	False	False	False	False	False	True	False	False
136	61.0	F	153.2	62.20	37.0	71.0	111.0	25.7	3.0	0.0	...	False	False	False	False	False	False	False	False	False	True
139	22.0	M	180.3	91.70	32.1	75.0	134.0	28.6	-6.9	36.0	...	False	False	False	False	False	False	False	True	False	False

5 rows × 22 columns

1 #Z-score method

2 def outliers_z(col, threshold=3):

3 mean = col.mean()

4 std = col.std()

5 z = (col - mean) / std

6 return abs(z) > threshold

7

8 data_z = df.copy()

9

10 for col in numeric_cols:

11 data_z[col + "_z_outlier"] = outliers_z(data_z[col])

12

13 data_z[data_z.filter(like="_z_outlier").any(axis=1)].head()

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	...	age_z_outlier	height_cm_z_outlier	weight_kg_z_outlier	body fat_%_z_outlier	diastolic_z_outlier	systolic_z_outlier	gripForce_z_outlier	sit and bend forward_cm_z_outlier	sit-ups counts_z_outlier	broad jump_cm_z_outlier
136	61.0	F	153.2	62.2	37.0	71.0	111.0	25.7	3.0	0.0	...	False	False	False	False	False	False	False	False	False	True
231	26.0	M	180.2	103.4	24.7	70.0	128.0	67.6	16.2	48.0	...	False	False	True	False	False	False	False	False	False	False
296	63.0	M	160.5	68.2	28.7	84.0	193.0	37.1	14.3	31.0	...	False	False	False	False	False	True	False	False	False	False
303	30.0	M	185.2	108.9	23.1	98.0	143.0	54.0	27.7	52.0	...	False	False	True	False	False	False	False	False	False	False
327	24.0	M	176.1	74.8	24.3	96.0	125.0	35.7	-12.0	1.0	...	False	False	False	False	False	False	False	True	False	False

5 rows × 22 columns

Section 5 – Fix Outliers

In this part, two ways to fix outliers will be used:

1. Replace outliers with NaN then fill them again using KNN
2. Remove rows with outliers

1 def replace_outliers(column):

2 Q1 = column.quantile(0.25)

3 Q3 = column.quantile(0.75)

4 IQR = Q3 - Q1

5 low = Q1 - 1.5 * IQR

6 high = Q3 + 1.5 * IQR

7 return column.where((column >= low) & (column <= high), np.nan)

8

9 clean_data = df.copy()

10

11 for col in numeric_cols:

12 clean_data[col] = replace_outliers(clean_data[col])

13

14 print(clean_data.isnull().sum())

15

16 from sklearn.impute import KNNImputer

17 imputer = KNNImputer(n_neighbors=3)

18 data_imputed = pd.DataFrame(imputer.fit_transform(clean_data[numeric_cols]),

19 columns=numeric_cols)

20

21 data_imputed.head()

age

gender

height_cm

weight_kg

body fat_%

diastolic

systolic

gripForce

sit and bend forward_cm

sit-ups counts

broad jump_cm

class

dtype: int64

	age	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm
0	27.0	172.3	75.24	21.3	80.0	130.0	54.9	18.4	60.0	217.0
1	25.0	165.0	55.80	15.7	77.0	126.0	36.4	16.3	53.0	229.0
2	31.0	179.6	78.00	20.1	92.0	152.0	44.8	12.0	49.0	181.0
3	32.0	174.5	71.10	18.4	76.0	147.0	41.4	15.2	53.0	219.0
4	28.0	173.8	67.70	17.1	70.0	127.0	43.5	27.1	45.0	217.0

Next steps: [Generate code with data_imputed](#) [New interactive sheet](#)

Section 6 – Data Visualization

In this part, the following charts will be used:

- Heatmap
- Histograms
- Boxplots
- Scatter plot
- Bar chart

1 #Heatmap

2 plt.figure(figsize=(10,7))

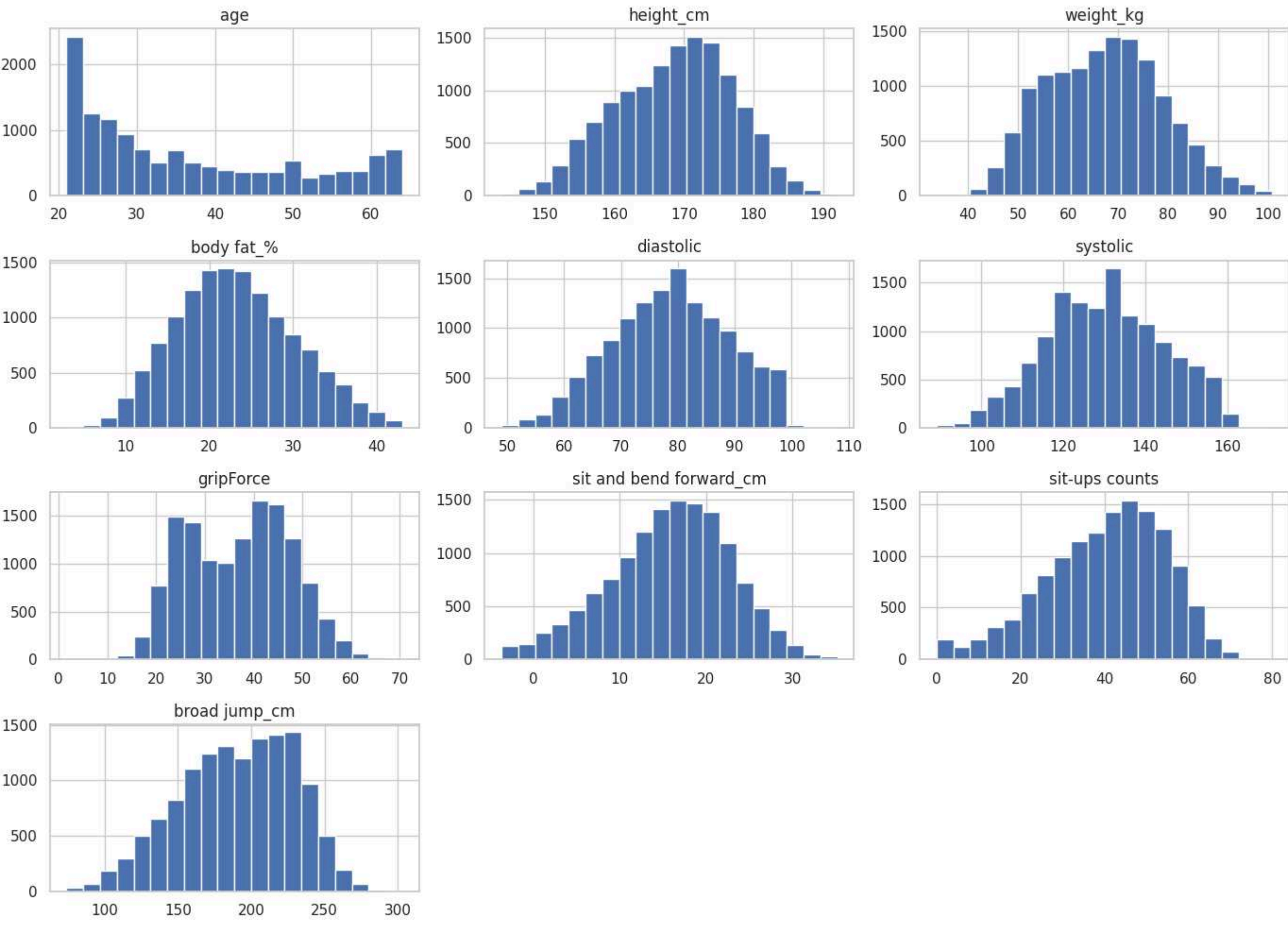
3 sns.heatmap(data_imputed.corr(), annot=True, cmap="coolwarm")

4 plt.show()

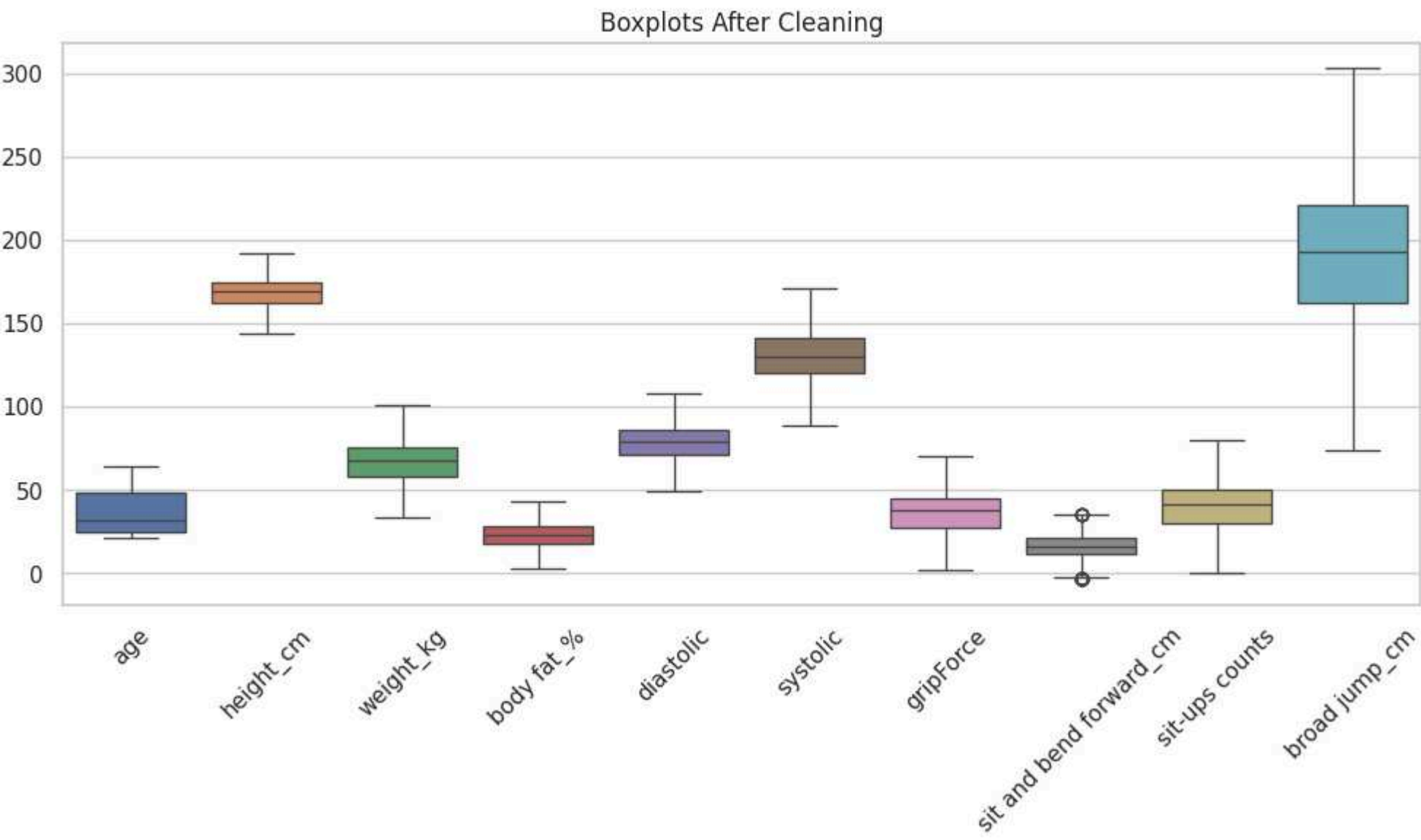
1 #Histograms

2 data_imputed.hist(figsize=(14,10), bins=20)

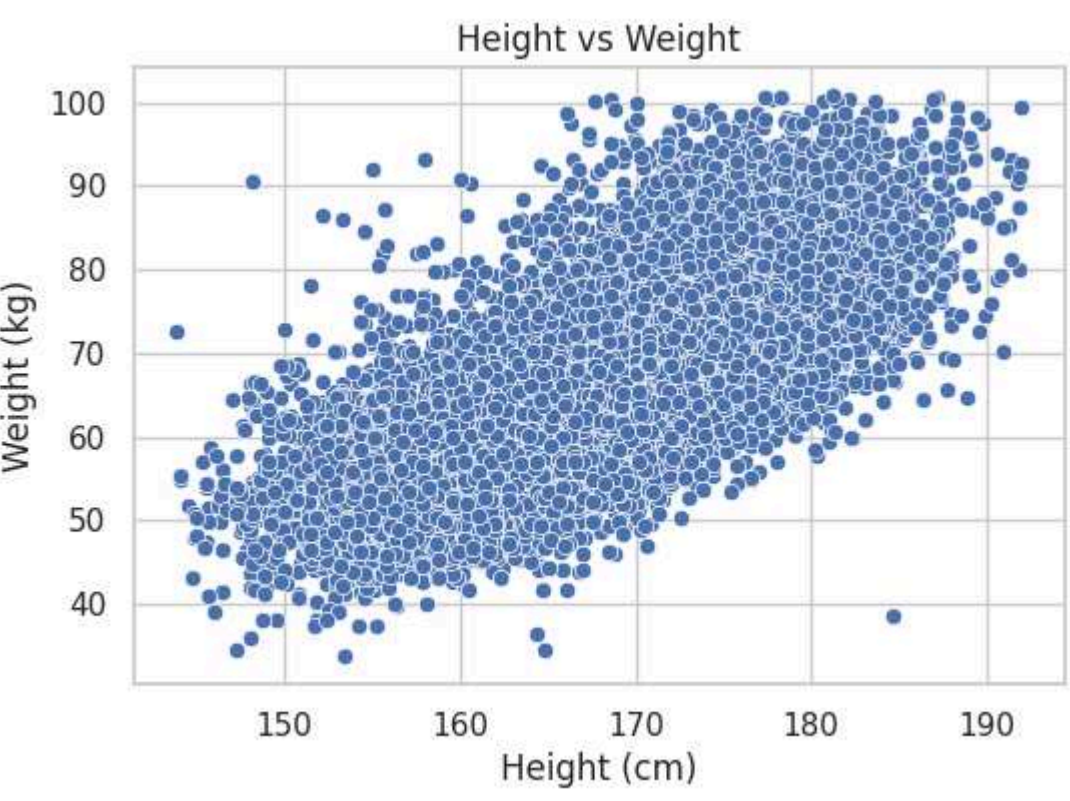
```
3 plt.tight_layout()
4 plt.show()
```



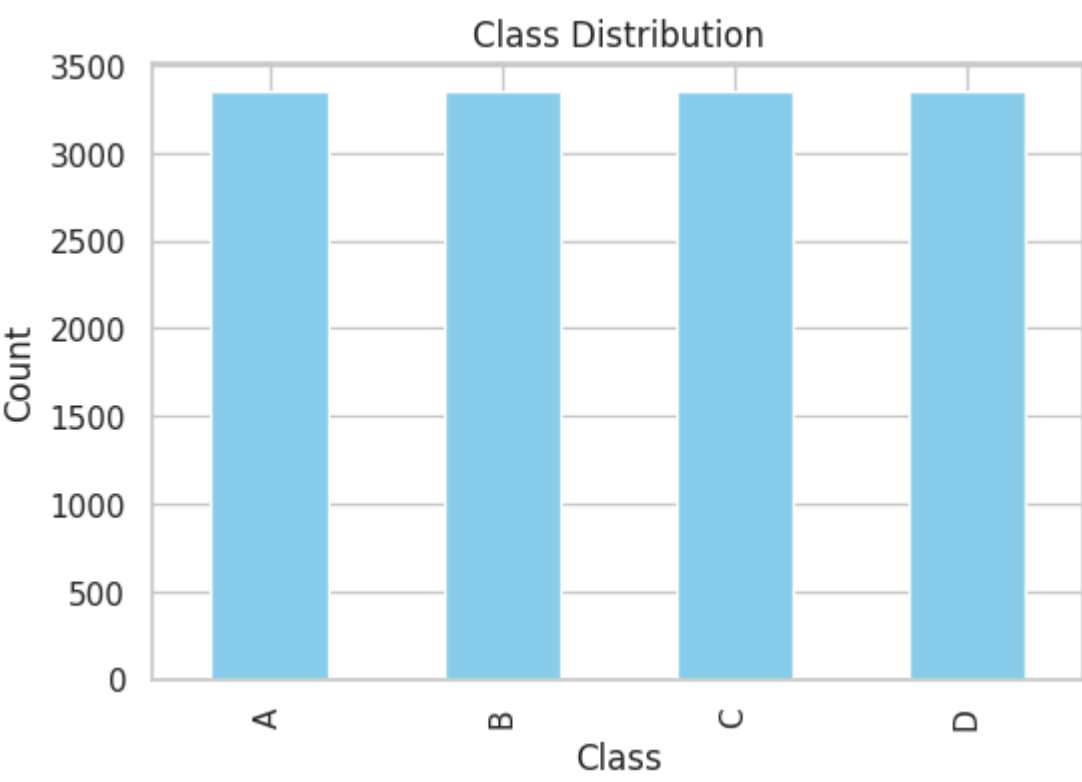
```
1 # Boxplots
2 clean_numeric = data_imputed.copy()
3
4 plt.figure(figsize=(12, 5))
5 sns.boxplot(data=clean_numeric)
6 plt.xticks(rotation=45)
7 plt.title("Boxplots After Cleaning")
8 plt.show()
```



```
1 #Scatter plot
2 clean_numeric = data_imputed.copy()
3
4 if "height_cm" in clean_numeric.columns and "weight_kg" in clean_numeric.columns:
5     plt.figure(figsize=(6, 4))
6     sns.scatterplot(x=clean_numeric["height_cm"], y=clean_numeric["weight_kg"])
7     plt.title("Height vs Weight")
8     plt.xlabel("Height (cm)")
9     plt.ylabel("Weight (kg)")
10    plt.show()
11 else:
12    print("Height and weight columns not found in the dataset.")
```



```
1 #Bar chart
2 if "class" in df.columns:
3     df["class"].value_counts().sort_index().plot(
4         kind="bar", figsize=(6, 4), color="skyblue"
5     )
6     plt.title("Class Distribution")
7     plt.xlabel("Class")
8     plt.ylabel("Count")
9     plt.show()
10 else:
11     print("Class column not found.")
```



DONE-PROJECT-FOR-Body-Performance-Data-Analysis