


Assignment Pandas

Please use this dataset.

<https://www.kaggle.com/datasets/nadeemajeedch/fitness-tracker-dataset>

login with your Kaggle account. And click the new Notebook.

 NADEEM MAJEED · UPDATED 2 DAYS AGO

2

New Notebook

Download

Fitness Tracker Dataset

Insights into Exercise Habit, Health Metric, and Fitness Level Across Individual

Data CardCode (0)Discussion (0)Suggestions (0)Settings

Pending Actions

USABILITY SCORE: 6.47

Add file information
Help others navigate your dataset with a description of each file

Include column descriptors
Empowers others to understand your data by describing its features

Specify provenance
Let others know how the data was collected and organized in the metadata tab

Specify update frequency
Let other users know if the dataset will be regularly updated in the metadata tab

You will get the following Notebook page.

notebook8016fb0ebeDraft saved

FileEditViewRunSettingsAdd-onsHelp

+

Draft Session (16m)

HDDCPURAM

[1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

/kaggle/input/fitness-tracker-dataset/gym_members_exercise_tracking_synthetic_data.csv

[2]:

```
df = pd.read_csv("/kaggle/input/fitness-tracker-dataset/gym_members_exercise_tracking_synthetic_data.csv")
df.head()
```

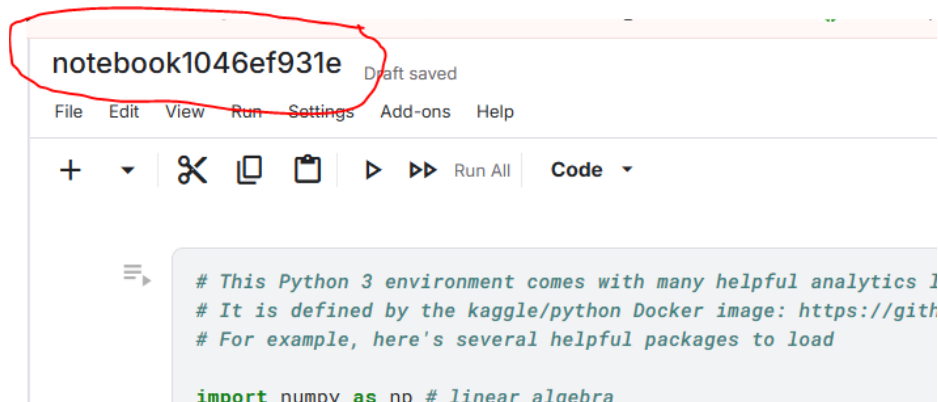
[2]:

	Age	Gender	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Session_Duration (hours)	Calories_Burned	Workout_Type	Fat_Percentage	Water_Intake (liters)	Workout_Frequency (days/week)
0	34.0	Female	86.7	1.86	174	152.0	74.0	1.12	712.0	Strength	12.8	2.4	
1	28.0	Male	81.7	1.88	165	155.0	73.0	1.00	688.0	Cardio	13.0	2.0	

On execution of the cell, you will get the path of the file. Use this path to load the data file.

```
df= pd.read_csv("/kaggle/input/fitness-tracker-dataset/gym_members_exercise_tracking_synthetic_data.csv")
```

Click here and change the file name.



Don't forget to vote for the dataset. 😊

NADEEM MAJEED · UPDATED 2 DAYS AGO

Fitness Tracker Dataset

Insights into Exercise Habit, Health Metric, and Fitness Level Across Individual



Data Card Code (0) Discussion (0) Suggestions (0) Settings

Tasks: Preprocessing and EDA Steps

Step 1: Load the Data

- Import the necessary libraries (pandas, numpy, matplotlib, seaborn).
- Load the dataset into a pandas DataFrame using `pd.read_csv()` or from the provided file.
- Display the first few rows using `.head()` to understand the structure.

Step 2: Inspect the Data

1. **Shape:** Use `.shape` to check the number of rows and columns.
2. **Columns:** Use `.columns` to list column names.
3. **Info:** Use `.info()` to examine data types and non-null counts for each column.
4. **Description:** Use `.describe()` to summarize numeric columns (mean, min, max, etc.).

Step 3: Identify and Handle Missing Values

1. Use `.isnull().sum()` to check the number of missing values per column.
2. Visualize missing data using a heatmap (`sns.heatmap`) to identify patterns.
3. Handle missing values:
 - **Age, Weight (kg), Height (m), and Numeric Columns:**
 - Impute missing values using the mean or median.
 - **Gender and Workout_Type:**
 - Impute missing values using the mode (most frequent value).
 - Document and justify your imputation strategy.

Step 4: Check for Duplicates

1. Use `.duplicated().sum()` to check for duplicate rows.
2. Remove duplicates, if any, using `.drop_duplicates()`.

Step 5: Validate Data

1. **Numeric Columns:**
 - Check for invalid entries (e.g., special characters like ? in `Max_BPM`).
 - Convert columns like `Max_BPM` to numeric using `pd.to_numeric()` with `errors='coerce'`.
 - Replace invalid values with `NaN` and impute as needed.
2. **Categorical Columns:**
 - Use `.unique()` to check for inconsistencies in `Gender`, `Workout_Type`, etc.
 - Standardize inconsistent values (e.g., `Male`, `M` → `Male`).

Step 6: Create New Features

1. **BMI Validation:**
 - Verify if the `BMI` column is consistent with the formula:
$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)}^2}$$

`BMI=Weight (kg)/Height (m)2`
 - Recalculate BMI where inconsistencies are found.
2. **Workout Intensity:**
 - Create a new column: `Workout_Intensity = Avg_BPM / Max_BPM`.

Step 7: Explore Data Distributions

1. Plot histograms for numeric columns (Age, Calories_Burned, etc.) to understand their distributions.
2. Use box plots to check for outliers in columns like Age, BMI, and Calories_Burned.
3. Examine the distribution of categorical columns (Gender, Workout_Type) using bar plots.

Step 8: Handle Outliers

1. Use the IQR method to detect outliers in numeric columns.
2. Decide whether to:
 - Remove outliers.
 - Transform them (e.g., log transformation).
 - Cap them (e.g., set to a specific threshold).

Step 9: Analyze Relationships

1. **Correlation:**
 - Use `.corr()` and visualize with a heatmap to find relationships between numeric columns (e.g., Calories_Burned, Session_Duration).
2. **Categorical vs. Numeric:**
 - Compare Calories_Burned and Workout_Type using a bar plot.
 - Analyze differences in BMI across Gender using a box plot.
3. **Multi-Variable:**
 - Use pair plots (`sns.pairplot`) to analyze relationships between key metrics (e.g., BMI, Calories_Burned, Workout_Frequency).

Step 10: Encode Categorical Variables

1. Convert Gender and Workout_Type to numeric formats:
 - Use one-hot encoding (`pd.get_dummies()`) or label encoding.

Step 11: Normalize Numeric Columns

1. Normalize or standardize columns with large ranges (Calories_Burned, Session_Duration, etc.) if needed for further analysis or machine learning.

Step 12: Summarize Findings

1. Highlight key insights from the data:
 - Trends in Calories_Burned based on Workout_Type and Gender.
 - Correlation between Session_Duration and Calories_Burned.
 - Any notable differences in BMI across Workout_Frequency.

If you think any other related task, you can add in notebook.

Happy Learning 😊