**Cell 1: Import Libraries**

python

Copy

Download

import pandas as pd

import matplotlib.pyplot as plt

**Connection to Problem Statement**

This step is foundational for all subsequent operations, as it imports the necessary libraries for data manipulation (pandas) and visualization (matplotlib).

**Definitions**

* pandas**(as**pd**):** A library for data manipulation and analysis. Provides structures like DataFrames for handling tabular data.
* matplotlib.pyplot**(as**plt**):** A plotting library for creating static, interactive, and animated visualizations in Python.

**Why Used?**

* Pandas is used to load, clean, and manipulate the dataset (e.g., handling missing values, removing outliers).
* Matplotlib is used for visualizing data (e.g., box plots for outlier detection).

**Cell 2: Load Dataset**

python

Copy

Download

df = pd.read\_csv('datasets/AcademicPerformance\_New.csv')

**Connection to Problem Statement**

Step 1 involves scanning variables for missing values and inconsistencies, which requires loading the dataset.

**Definitions**

* pd.read\_csv()**:** Reads a CSV file into a pandas DataFrame.

**Why Used?**

To load the dataset (AcademicPerformance\_New.csv) into a DataFrame for further analysis.

**Cell 3: Display First Few Rows**

python

Copy

Download

df.head()

**Connection to Problem Statement**

Provides an initial look at the data to understand its structure and identify potential issues (Step 1).

**Definitions**

* df.head()**:** Displays the first 5 rows of the DataFrame.

**Why Used?**

To quickly inspect the dataset and verify that it loaded correctly.

**Cell 4: Dataset Information**

python

Copy

Download

df.info()

**Connection to Problem Statement**

Helps identify missing values and data types (Step 1).

**Definitions**

* df.info()**:** Prints a summary of the DataFrame, including column names, non-null counts, and data types.

**Why Used?**

To check for missing values (Non-Null Count) and inconsistencies in data types.

**Cell 5: Handling Missing Values (Median Imputation)**

python

Copy

Download

df['WT'].fillna(df['WT'].mean(), inplace=True)

df['DSBDA'].fillna(df['DSBDA'].mean(), inplace=True)

df['AI'].fillna(df['DSBDA'].mean(), inplace=True)

**Connection to Problem Statement**

Step 1: Handling missing values using the mean (a suitable technique).

**Definitions**

* fillna()**:** Replaces missing values (NaN) with a specified value (here, the mean of the column).
* inplace=True**:** Modifies the DataFrame directly instead of returning a new one.

**Why Used?**

To address missing values in the columns WT, DSBDA, and AI by filling them with the mean of their respective columns.

**Cell 6: Drop Rows with Missing 'Roll No'**

python

Copy

Download

df.dropna(subset=['Roll No'], inplace=True)

**Connection to Problem Statement**

Step 1: Removing rows with missing values in critical columns (Roll No).

**Definitions**

* dropna()**:** Removes rows with missing values.
* subset=['Roll No']**:** Specifies the column to check for missing values.

**Why Used?**

To ensure the dataset does not contain rows with missing Roll No values, which are likely essential for analysis.

**Cell 7: Verify Missing Values Handled**

python

Copy

Download

df.info()

**Connection to Problem Statement**

Confirms that missing values have been addressed (Step 1).

**Why Used?**

To verify that the previous operations (fillna and dropna) successfully handled missing values.

**Cell 8: Check for Duplicates**

python

Copy

Download

df.duplicated().sum()

**Connection to Problem Statement**

Part of Step 1 (identifying inconsistencies like duplicates).

**Definitions**

* df.duplicated()**:** Returns a boolean Series indicating duplicate rows.
* .sum()**:** Counts the number of True values (duplicates).

**Why Used?**

To check for and count duplicate rows in the dataset.

**Cell 9: Convert 'Roll No' to Integer**

python

Copy

Download

df['Roll No'].astype(int)

**Connection to Problem Statement**

Ensures consistency in data types (Step 1).

**Definitions**

* astype(int)**:** Converts the column values to integers.

**Why Used?**

To standardize the Roll No column as integers for consistency.

**Cell 10: Outlier Detection (Box Plot for 'WT')**

python

Copy

Download

plt.boxplot(df['WT'])

plt.title('WT')

plt.grid(True)

plt.show()

**Connection to Problem Statement**

Step 2: Scanning numeric variables for outliers.

**Definitions**

* plt.boxplot()**:** Creates a box plot to visualize the distribution of data and identify outliers.
* plt.title()**,**plt.grid()**,**plt.show()**:** Customize and display the plot.

**Why Used?**

To visually detect outliers in the WT column.

**Cell 11: Calculate IQR for 'WT'**

python

Copy

Download

Q1 = df['WT'].quantile(0.25)

Q3 = df['WT'].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

lower, upper

**Connection to Problem Statement**

Step 2 and 3: Using IQR to identify and later remove outliers.

**Definitions**

* quantile(0.25)**and**quantile(0.75)**:** Calculate the first and third quartiles.
* **IQR (Interquartile Range):** Q3 - Q1.
* **Lower and Upper Bounds:** Define the range for non-outlier data (Q1 - 1.5\*IQR to Q3 + 1.5\*IQR).

**Why Used?**

To compute the bounds for outlier detection in the WT column.

**Cell 12: Remove Outliers for 'WT'**

python

Copy

Download

df = df[~((df['WT'] < lower) | (df['WT'] > upper))]

**Connection to Problem Statement**

Step 3: Removing outliers identified using IQR.

**Definitions**

* ~**:** Logical NOT operator (inverts the condition).
* (df['WT'] < lower) | (df['WT'] > upper)**:** Filters rows where WT is outside the bounds.

**Why Used?**

To exclude outliers from the dataset.

**Cell 13: Verify Outliers Removed (Box Plot for 'WT')**

python

Copy

Download

plt.boxplot(df['WT'])

plt.title('WT')

plt.grid(True)

plt.show()

**Connection to Problem Statement**

Step 3: Confirming outliers have been removed.

**Why Used?**

To visually verify that outliers in WT were successfully removed.

**Cell 14: Repeat for 'DSBDA' (Box Plot)**

python

Copy

Download

plt.boxplot(df['DSBDA'])

plt.title('DSBDA')

plt.grid(True)

plt.show()

**Connection to Problem Statement**

Step 2 and 3: Extending outlier detection to DSBDA.

**Why Used?**

To detect outliers in the DSBDA column.

**Cell 15: Calculate IQR for 'DSBDA'**

python

Copy

Download

Q1 = df['DSBDA'].quantile(0.25)

Q3 = df['DSBDA'].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

lower, upper

**Why Used?**

To compute bounds for outlier detection in DSBDA.

**Cell 16: Remove Outliers for 'DSBDA'**

python

Copy

Download

df = df[~((df['DSBDA'] < lower) | (df['DSBDA'] > upper))]

**Why Used?**

To exclude outliers from the DSBDA column.

**Cell 17: Verify Outliers Removed (Box Plot for 'DSBDA')**

python

Copy

Download

plt.boxplot(df['DSBDA'])

plt.title('DSBDA')

plt.grid(True)

plt.show()

**Why Used?**

To confirm successful removal of outliers in DSBDA.

**Summary of Steps**

1. **Data Loading and Inspection:**
   * Loaded the dataset and inspected its structure (head(), info()).
2. **Handling Missing Values:**
   * Filled missing values with the mean (fillna()).
   * Dropped rows with missing Roll No (dropna()).
3. **Outlier Detection and Removal:**
   * Used box plots and IQR to identify and remove outliers for WT and DSBDA.
4. **Verification:**
   * Confirmed missing values and outliers were addressed (info(), box plots).