CODE DESCRIPTION

---

1. Uploading the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Explanation:

- from google.colab import files imports the file-handling utilities provided by Google Colab.
- files.upload() opens a dialog box allowing the user to upload files from the local system.
- The uploaded file is stored temporarily in the Colab environment.

---

2. Verifying Uploaded Files

```
import os
os.listdir("/content")
```

Explanation:

- Import os imports the operating system module. - os.listdir("/content") lists all files in the Colab working directory.

---

3. Importing Pandas and Reading the Dataset

```
import pandas as pd

df = pd.read_csv(
    "/content/exponetial.csv",
    comment="#",
    low_memory=False
)

df.head()
```

Explanation:

- Pandas is imported for data manipulation.
- pd.read_csv() loads the CSV file into a DataFrame.
- comment="#" ignores commented lines in the file.
- low_memory=False ensures correct data type detection.

- df.head() displays the first five rows for inspection.

---

4. Installing Required Libraries

```
pip install pandas numpy matplotlib seaborn
```

Explanation:

- Installs required libraries for data analysis and visualization.
- This step is necessary in cloud environments like Google Colab.

---

5. Importing Libraries for Processing

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
```

Explanation:

- NumPy is used for numerical operations.
- Matplotlib and Seaborn are used for visualization.
- Z-score is imported for statistical analysis.

---

6. Reloading Dataset and Checking Row Count

```python
df = pd.read_csv("exponetial.csv", comment="#", low_memory=False)
print("Original rows:", df.shape[0])
```

Explanation:

- Reloads the dataset to ensure a clean preprocessing start.
- df.shape[0] returns the number of rows.

---

7. Selecting Relevant Columns

```python
df = df[[
    "pl_rade",
    "pl_bmasse",
    "pl_orbper",
    "pl_orbsmax",
```

```
    "pl_eqt",
    "pl_dens",
    "st_teff",
    "st_lum",
    "st_met",
    "st_spectype"
]]
```

Explanation:

- Retains only columns relevant to exoplanet habitability.
- Removes unnecessary attributes to reduce noise.

---

8. Renaming Columns

```
df.columns = [
    "planet_radius",
    "planet_mass",
    "orbital_period",
    "semi_major_axis",
    "equilibrium_temp",
    "planet_density",
    "star_temp",
    "star_luminosity",
    "star_metallicity",
    "star_type"
]
```

Explanation:

- Converts technical column names into readable and meaningful names.

---

9. Handling Missing Values

```
num_cols = df.select_dtypes(include=np.number).columns
df[num_cols] = df[num_cols].fillna(df[num_cols].median())
df["star_type"] = df["star_type"].fillna(df["star_type"].mode()[0])
```

Explanation:

- Numerical missing values are filled using the median.
- Categorical missing values are filled using the mode.

---

10. Removing Invalid Values

```
df = df[
    (df["planet_radius"] > 0) &
    (df["planet_mass"] > 0) &
    (df["equilibrium_temp"] > 0)
]
```

Explanation:

- Removes physically impossible values such as negative radius or temperature.

---

11. Outlier Handling Using IQR

```
def cap_iqr(data, col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    data[col] = data[col].clip(lower, upper)
```

Explanation:

- Uses the Interquartile Range method to cap extreme values.
- Prevents data loss while reducing outlier impact.

---

12. Applying Outlier Treatment

```
for col in num_cols:
    cap_iqr(df, col)
```

Explanation:

- Applies IQR capping to all numerical columns.

---

13. Habitability Score Calculation

```
df["habitability_score"] = (
  (1 - abs(df["equilibrium_temp"] - 288) / 288).clip(0, 1) * 0.4 +
    (1 - abs(df["planet_radius"] - 1)).clip(0, 1) * 0.3 +
    (1 / (1 + abs(df["semi_major_axis"] - 1))).clip(0, 1) * 0.3
)
```

Explanation:

- Calculates a weighted habitability score based on Earth similarity.

- Temperature contributes 40%, radius 30%, and orbital distance 30%.

---

14. Feature Engineering (Encoding Star Type)

```
df["star_class"] = df["star_type"].astype(str).str[0]
df = pd.get_dummies(df, columns=["star_class"], drop_first=True)
```

Explanation:

- Extracts stellar spectral class.
- Converts categorical values into numeric features.

---

15. Creating Binary Habitability Label

```
df["habitable"] = (df["habitability_score"] >= 0.4).astype(int)
```

Explanation:

- Creates a binary classification label.
- 1 indicates habitable, 0 indicates non-habitable.

---

16. Removing Duplicates and Saving File

```
df = df.drop_duplicates()
df.to_csv("preprocessed.csv", index=False)
```

Explanation:

- Removes duplicate records.
- Saves the cleaned dataset.

---

17. Verifying Processed Dataset

```
df = pd.read_csv("/content/preprocessed.csv")
df.head()
```

Explanation: -

- Reloads the processed file for verification.

Final Conclusion

This notebook performs complete preprocessing of exoplanet data, transforming raw astronomical observations into a clean, feature-rich, machine-learning-ready dataset suitable for habitability analysis.