ExoHabitAI – Data Preprocessing and Feature Engineering Documentation

1. Introduction

The objective of this preprocessing module is to transform raw exoplanet observational data into a structured and machine-learning–ready dataset for predicting planetary habitability. Astronomical datasets obtained from space observation missions contain missing values, inconsistent measurements, extreme outliers, and non-informative attributes. Therefore, systematic preprocessing and feature engineering are required before applying machine learning algorithms.

This notebook implements a complete preprocessing pipeline including data cleaning, validation using physical constraints, feature construction, encoding, normalization, and dataset export.

2. Dataset Description

The dataset consists of observed properties of exoplanets and their host stars collected from astronomical catalogs. Each row represents one discovered planet, while columns describe planetary characteristics, orbital parameters, and stellar properties.

Selected Core Features

Feature	Description
Planet Radius (`pl_rade`)	Size of planet relative to Earth
Planet Mass (`pl_bmasse`)	Mass of planet
Orbital Period (`pl_orbper`)	Time taken to orbit host star
Semi-major Axis (`pl_orbsmax`)	Average orbital distance
Equilibrium Temperature (`pl_eqt`)	Estimated surface temperature
Stellar Temperature (`st_teff`)	Host star surface temperature
Stellar Luminosity (`st_lum`)	Energy output of star
Spectral Type (`st_spectype`)	Star classification

Feature selection was performed to retain only scientifically relevant parameters influencing planetary habitability.

3. Preprocessing Pipeline Overview

The preprocessing workflow follows the sequence below:

```
Data Loading
→ Dataset Inspection
→ Feature Selection
→ Missing Value Handling
→ Duplicate Removal
→ Physical Validation
→ Outlier Treatment
→ Feature Engineering
→ Encoding
→ Target Variable Creation
→ Feature Scaling
→ Dataset Export
```

4. Data Loading

The dataset is loaded using the Pandas library. Metadata lines beginning with "#" are ignored during loading to ensure only tabular data is processed.

```
pd.read_csv(file, comment="#")
```

This step ensures compatibility with astronomical datasets containing descriptive headers.

5. Dataset Inspection

Initial exploration is performed using:

- Dataset shape inspection
- Data type verification
- Statistical summaries
- Missing value analysis

This stage identifies inconsistencies and determines suitable preprocessing strategies.

6. Missing Value Handling

Astronomical observations frequently contain incomplete measurements due to observational limitations.

Strategy Used

Data Type	Method
Numerical features	Median imputation
Categorical features	Mode imputation

Median imputation is preferred because astronomical data often contains extreme values that distort mean-based filling.

Example 1: Extremely massive planets skew average mass values.
Example 2: High stellar temperatures create unrealistic averages.

7. Duplicate Removal

Duplicate records are removed to prevent biased learning outcomes.

```
df.drop_duplicates()
```

Duplicate observations may artificially increase the importance of certain planetary conditions during training.

8. Physical Validity Filtering

Scientific constraints are applied to ensure physically meaningful data.

Conditions enforced:

- Planet radius > 0
- Planet mass > 0
- Temperature > 0 Kelvin

These filters eliminate impossible measurements produced by observational or recording errors.

Example 1: Negative radius values violate physical laws.
Example 2: Negative Kelvin temperatures are thermodynamically impossible.

9. Outlier Treatment

Extreme values are controlled using percentile clipping (1st–99th percentile).

```
clip(lower_percentile, upper_percentile)
```

This prevents rare astronomical objects from dominating model training.

Example 1: Gas giants with extremely large radii distort scaling.
Example 2: Highly luminous stars produce disproportionate feature ranges.

10. Feature Engineering

Feature engineering converts raw scientific measurements into interpretable machine learning signals.

10.1 Habitability Score Index

A composite score estimating similarity to Earth-like conditions using:

- Temperature proximity
- Planet radius similarity
- Orbital distance

Higher similarity results in higher habitability scores.

Example 1: Earth-like temperature → high score.
Example 2: Extremely hot planets → low score.

10.2 Stellar Compatibility Index

Measures how similar a host star is to the Sun using stellar temperature and luminosity.

Stable Sun-like stars are more likely to support habitable environments.

Example 1: G-type stars increase compatibility.
Example 2: Very hot stars reduce stability likelihood.

10.3 Orbital Stability Factor

Derived from orbital distance and period to estimate climate stability.

Stable orbital dynamics indicate reduced environmental fluctuation.

11. Categorical Encoding

Categorical features such as stellar spectral type are converted into numerical format using one-hot encoding.

```
pd.get_dummies()
```

Machine learning models require numerical inputs for training.

12. Target Variable Creation

A classification label is generated from the habitability score.

Binary Classification

```
Habitable = 1 if score > threshold
Else = 0
```

Multi-class Classification

Class	Meaning
0	Low Habitability
1	Moderate Habitability
2	High Habitability

This enables both binary and multi-class prediction models.

13. Feature Scaling

Numerical features are standardized using:

```
StandardScaler()
```

Scaling ensures equal contribution of all features during model training.

Example 1: Temperature values are much larger than radius values.
Example 2: Without scaling, large-range variables dominate learning.

14. Final Dataset Generation

The processed dataset is exported for model training.

```
df.to_csv("processed_dataset.csv")
```

This separates preprocessing from modeling and ensures reproducibility.

15. Assumptions and Limitations

1. Habitability threshold selection is heuristic and may require tuning.
2. Linear similarity assumptions simplify complex astrophysical relationships.
3. Observational uncertainty is not explicitly modeled.

16. Conclusion

The preprocessing pipeline transforms raw exoplanet observational data into a clean, validated, and feature-enriched dataset suitable for machine learning applications. By integrating domain knowledge with data engineering techniques, the workflow ensures scientific consistency while improving predictive model performance.