PreProcessing

This steps below describes the various stages dataset goes through preprocessing

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

1. LOAD & INITIAL FILTERING

```
filename = '/content/PS_2026.01.02_20.41.33.csv'

# Read file (ignoring comment lines starting with #)
df = pd.read_csv(filename, comment='#')

# Filter for the "Default Parameter Set" (Best available data)
df = df[df['default_flag'] == 1].copy()

# Map Dataset Columns to User Features
features_map = {
    'pl_rade': 'Planet Radius',
    'pl_bmasse': 'Planet Mass',
    'pl_orbper': 'Orbital Period',
    'pl_orbsmax': 'Semi-Major Axis',
    'pl_eqt': 'Equilibrium Temperature',
    'pl_dens': 'Planet Density',
    'st_teff': 'Stellar Temperature',
    'st_lum': 'Stellar Luminosity',
    'st_met': 'Stellar Metallicity',
    'st_spectype': 'Star Type',
    'pl_insol': 'Insolation Flux' # Required for Habitability
Calculation
}

# Keep only relevant columns
df_selected = df[list(features_map.keys())].copy()
```

```
/tmp/ipython-input-2806536548.py:4: DtypeWarning: Columns (4,5) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(filename, comment='#')
```

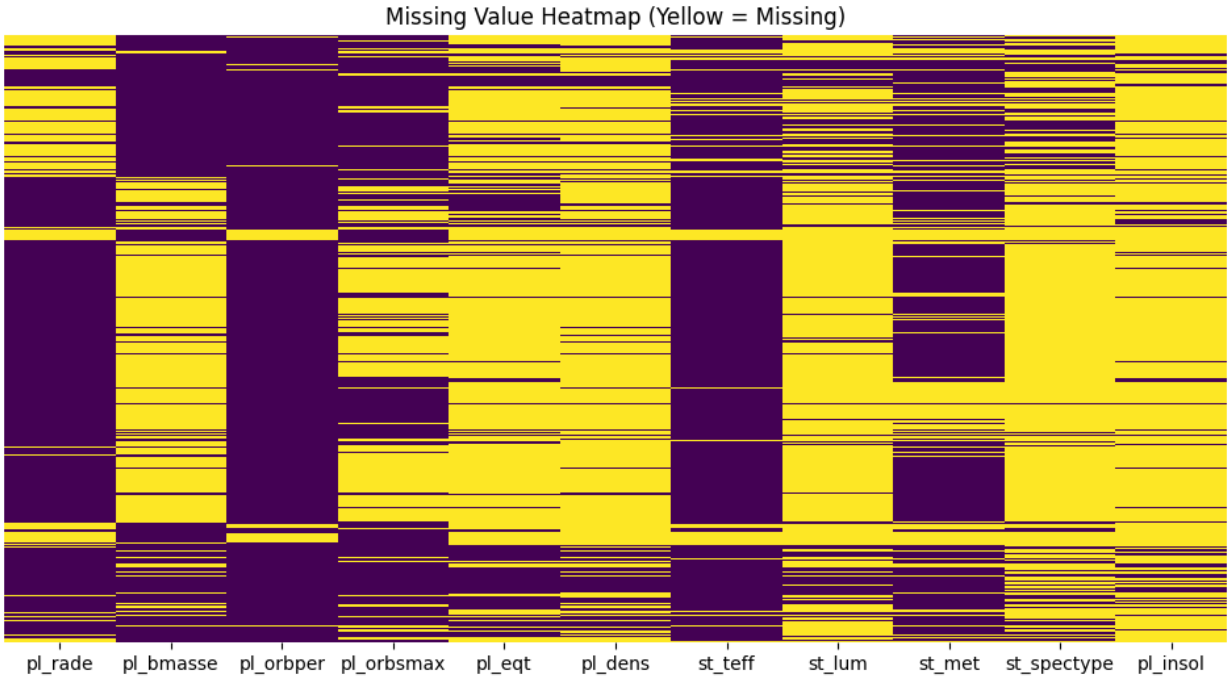
1. DATA QUALITY Checking

```
print("--- Generating Data Quality Reports ---")

# A. Generate Missing Value Heatmap
plt.figure(figsize=(12, 6))
```

```
sns.heatmap(df_selected.isnull(), cbar=False, cmap='viridis',
yticklabels=False)
plt.title('Missing Value Heatmap (Yellow = Missing)')
plt.savefig('missing_heatmap.png')
print("Saved: missing_heatmap.png")

--- Generating Data Quality Reports ---
Saved: missing_heatmap.png
```



1. HANDLING MISSING DATA

```
print("--- Handling Missing Data ---")

# Define Numerical and Categorical columns
num_cols = ['pl_rade', 'pl_bmasse', 'pl_orbper', 'pl_orbsmax',
'pl_eqt',
            'pl_dens', 'st_teff', 'st_lum', 'st_met', 'pl_insol']
cat_cols = ['st_spectype']

# Strategy: Median for numbers (robust to outliers), Mode for text
num_imputer = SimpleImputer(strategy='median')
cat_imputer = SimpleImputer(strategy='most_frequent')

# Apply Imputation
df_selected[num_cols] =
num_imputer.fit_transform(df_selected[num_cols])
df_selected[cat_cols] =
cat_imputer.fit_transform(df_selected[cat_cols])
```

```
--- Handling Missing Data ---
```

1. OUTLIER DETECTION

```
print("--- Removing Outliers ---")

# A. Physically Impossible Values
# Radius <= 0 or Temperature <= 0 are physics errors
mask_impossible = (df_selected['pl_rade'] <= 0) |
(df_selected['pl_eqt'] <= 0) | (df_selected['st_teff'] <= 0)
df_selected = df_selected[~mask_impossible]

# B. Extreme Values (Cap/Remove)
# Removing likely errors: Radius > 50 Earth Radii (Jupiter is ~11)
mask_extreme = df_selected['pl_rade'] > 50
df_selected = df_selected[~mask_extreme]
```

```
--- Removing Outliers ---
```

1. FEATURE ENGINEERING

```
print("--- Engineering Features ---")

# A. Target Variable: Binary Habitability (Class Label)
# Rule: Rocky (Radius < 2.0) AND In Habitable Zone (Insolation 0.2-1.5
OR Temp 180-310K)
is_rocky = df_selected['pl_rade'] <= 2.0
in_hz_insol = df_selected['pl_insol'].between(0.2, 1.5)
in_hz_temp = df_selected['pl_eqt'].between(180, 310)

df_selected['Target_Habitable'] = ((is_rocky) & (in_hz_insol |
in_hz_temp)).astype(int)

# B. Habitability Score Index (Continuous)
# Mathematical proxy for Earth-likeness
df_selected['Habitability_Score'] = (
    np.exp(-np.abs(df_selected['pl_rade'] - 1.0)) +              #
Radius Similarity
    np.exp(-np.abs((df_selected['pl_eqt'] - 288.0)/288.0)) +     #
Temp Similarity
    np.exp(-np.abs(df_selected['pl_insol'] - 1.0))               #
Flux Similarity
) / 3.0

# C. Stellar Compatibility Index
# Gaussian curve peaking at Sun-like temperature (5778 K)
df_selected['Stellar_Compatibility'] = np.exp(-
np.abs((df_selected['st_teff'] - 5778.0)/1000.0))

# D. Orbital Stability Factor
# Combined metric of Period and Distance
```

```
df_selected['Orbital_Stability_Score'] =
np.log1p(df_selected['pl_orbper'] * df_selected['pl_orbsmax'])

# E. Categorical Encoding (Star Type)
# Extract the first letter (e.g., 'G2 V' -> 'G')
df_selected['Star_Type_Main'] =
df_selected['st_spectype'].astype(str).str[0].str.upper()

# Filter to standard main sequence types, map others to 'Other'
valid_types = ['O', 'B', 'A', 'F', 'G', 'K', 'M']
df_selected['Star_Type_Main'] =
df_selected['Star_Type_Main'].apply(lambda x: x if x in valid_types
else 'Other')

# One-Hot Encoding
ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
type_encoded = ohe.fit_transform(df_selected[['Star_Type_Main']])
type_cols = ohe.get_feature_names_out(['Star_Type_Main'])

# Create DataFrame for encoded columns
df_encoded_types = pd.DataFrame(type_encoded, columns=type_cols,
index=df_selected.index)

# Join and drop original text columns
df_final = pd.concat([df_selected, df_encoded_types], axis=1)
df_final.drop(columns=['st_spectype', 'Star_Type_Main'], inplace=True)

--- Engineering Features ---
```

FEATURE SCALING (Standardization)

```
print("--- Scaling Features ---")

# Define columns to scale (Do not scale the Target Variable!)
cols_to_scale = num_cols + ['Habitability_Score',
'Stellar_Compatibility', 'Orbital_Stability_Score']

scaler = StandardScaler()
df_final[cols_to_scale] =
scaler.fit_transform(df_final[cols_to_scale])

# ==========================================
# 7. EXPORT
# ==========================================
df_final.to_csv('preprocessed.csv', index=False)
print("\nSuccess! 'preprocessed.csv' has been created.")
print(f"Final Dataset Shape: {df_final.shape}")
print(df_final.head())
```

```
--- Scaling Features ---

Success! 'preprocessed.csv' has been created.
Final Dataset Shape: (6064, 21)
     pl_rade  pl_bmasse  pl_orbper  pl_orbsmax    pl_eqt   pl_dens
st_teff  \
0  -0.357554   4.112518  -0.024711   -0.037713 -0.093477 -0.028238 -
0.384423
4  -0.357554   3.899939  -0.023503   -0.036641 -0.093477 -0.028238 -
0.857096
7  -0.357554   0.616646  -0.025565   -0.038940 -0.093477 -0.028238 -
0.374412
9  -0.357554   1.936297  -0.015686   -0.032852 -0.093477 -0.028238
0.098977
17 -0.357554   0.094249  -0.021736   -0.036245 -0.093477 -0.028238
0.241994

      st_lum    st_met  pl_insol  ...  Habitability_Score  \
0   3.966854 -1.769338 -0.081864  ...           -0.310124
4   0.056449 -0.227448 -0.081864  ...           -0.310124
7   3.706981 -1.448111 -0.081864  ...           -0.310124
9   0.056449  0.029533 -0.081864  ...           -0.310124
17  0.056449  0.286515 -0.081864  ...           -0.310124

    Stellar_Compatibility  Orbital_Stability_Score
Star_Type_Main_A  \
0               -0.840104                1.485643                0.0

4               -1.577405                1.757317                0.0

7               -0.818612                1.126459                0.0

9                0.632465                2.437453                0.0

17               1.296003                1.950254                0.0


    Star_Type_Main_B  Star_Type_Main_F  Star_Type_Main_G
Star_Type_Main_K  \
0                0.0               0.0               1.0
0.0
4                0.0               0.0               1.0
0.0
7                0.0               0.0               0.0
1.0
9                0.0               0.0               1.0
0.0
17               0.0               0.0               1.0
0.0
```

```
    Star_Type_Main_M  Star_Type_Main_Other
0                0.0                   0.0
4                0.0                   0.0
7                0.0                   0.0
9                0.0                   0.0
17               0.0                   0.0

[5 rows x 21 columns]
```