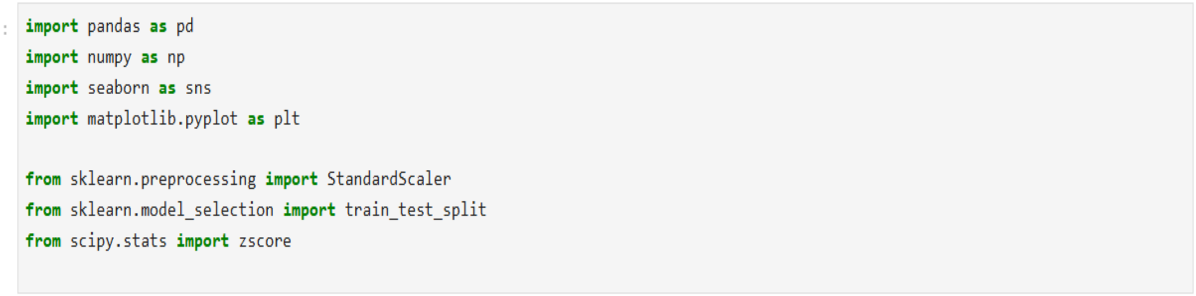
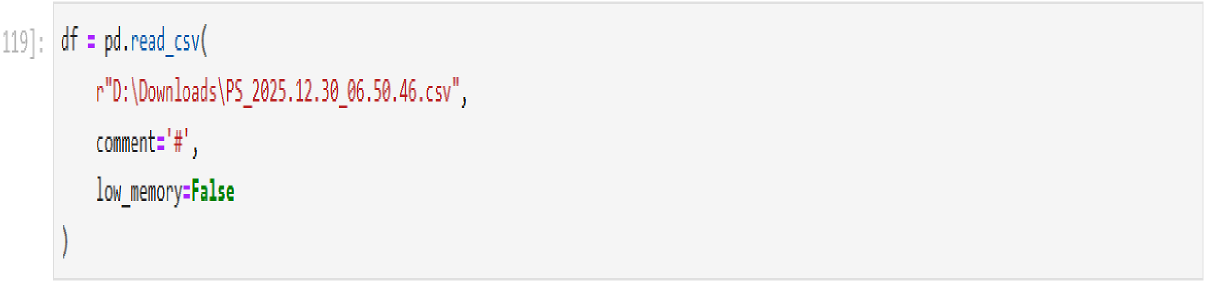
MILE STONE – 2 : DATA PREPROCESSING

IMPORTING LIBRARIES



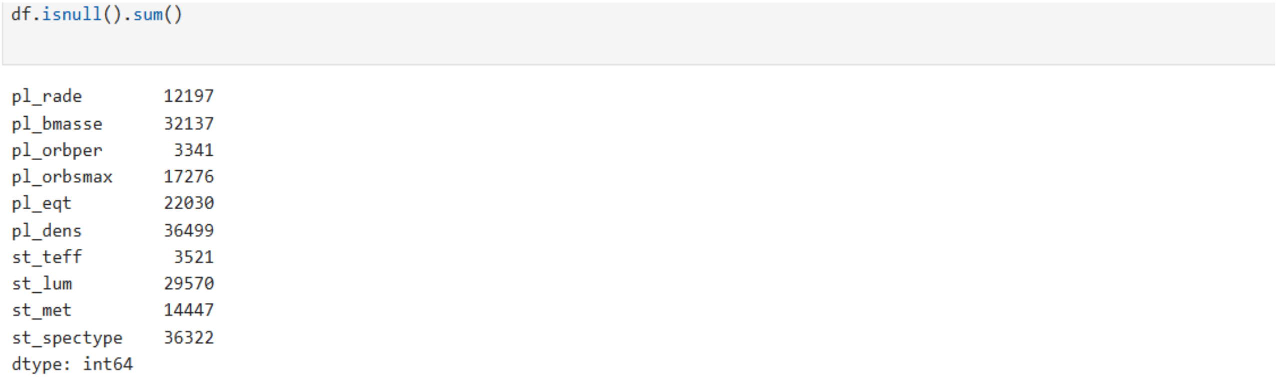
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from scipy.stats import zscore
```

LOADING DATASET



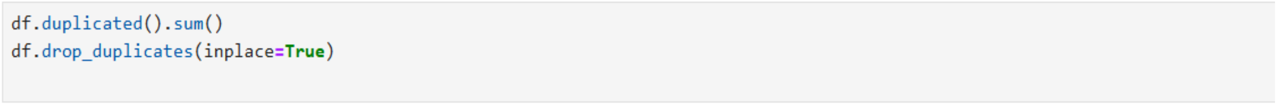
```
119]: df = pd.read_csv(
          r"D:\Downloads\PS_2025.12.30_06.50.46.csv",
          comment='#',
          low_memory=False
      )
```

DATA QUALITY ASSESSMENT

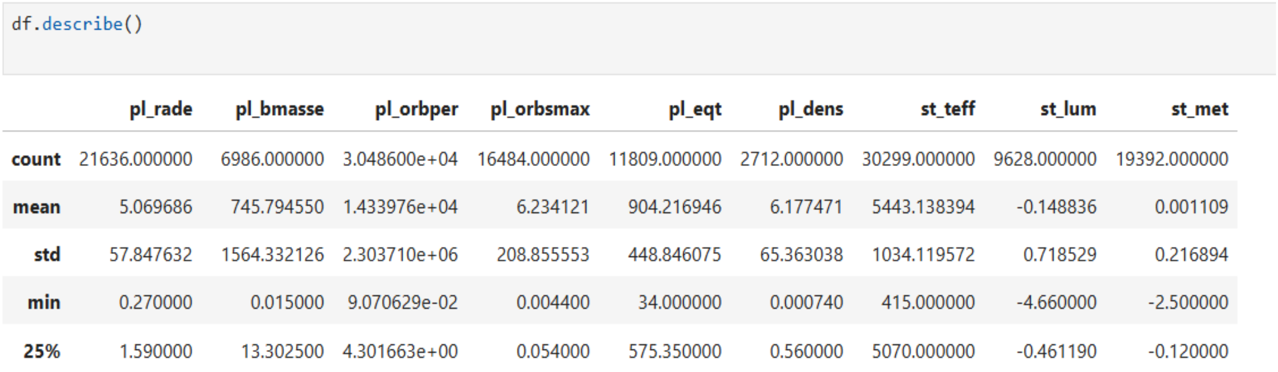


```
df.isnull().sum()
```

```
pl_rade        12197
pl_bmasse      32137
pl_orbper       3341
pl_orbsmax     17276
pl_eqt         22030
pl_dens        36499
st_teff         3521
st_lum         29570
st_met         14447
st_spectype    36322
dtype: int64
```



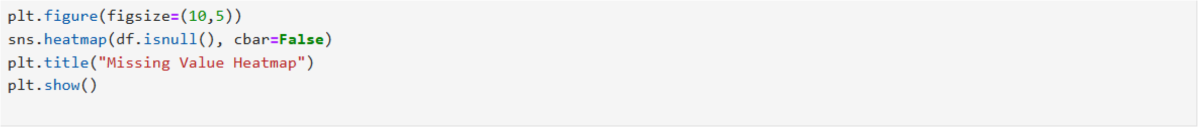
```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
```



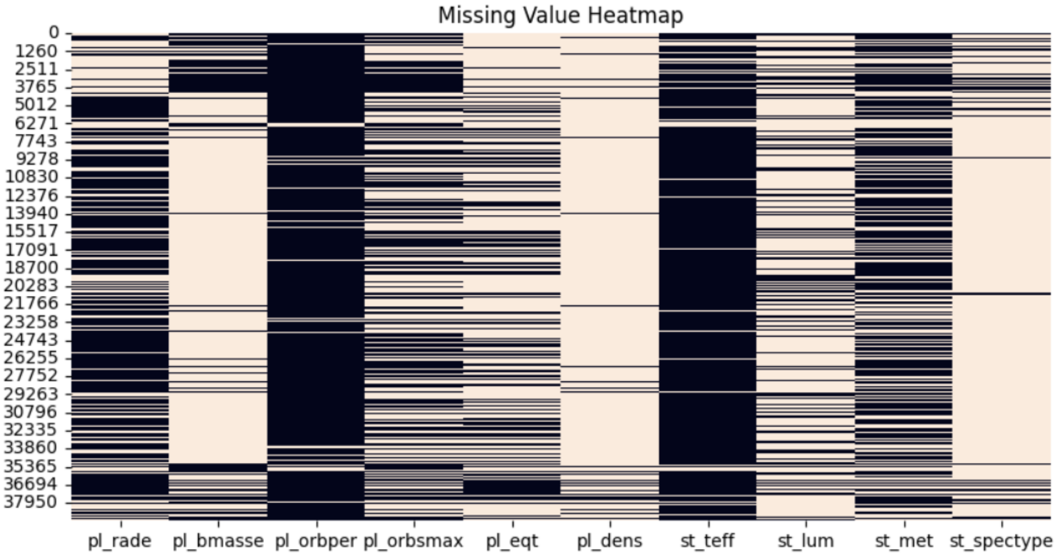
```
df.describe()
```

	pl_rade	pl_bmasse	pl_orbper	pl_orbsmax	pl_eqt	pl_dens	st_teff	st_lum	st_met
count	21636.000000	6986.000000	3.048600e+04	16484.000000	11809.000000	2712.000000	30299.000000	9628.000000	19392.000000
mean	5.069686	745.794550	1.433976e+04	6.234121	904.216946	6.177471	5443.138394	-0.148836	0.001109
std	57.847632	1564.332126	2.303710e+06	208.855553	448.846075	65.363038	1034.119572	0.718529	0.216894
min	0.270000	0.015000	9.070629e-02	0.004400	34.000000	0.000740	415.000000	-4.660000	-2.500000
25%	1.590000	13.302500	4.301663e+00	0.054000	575.350000	0.560000	5070.000000	-0.461190	-0.120000

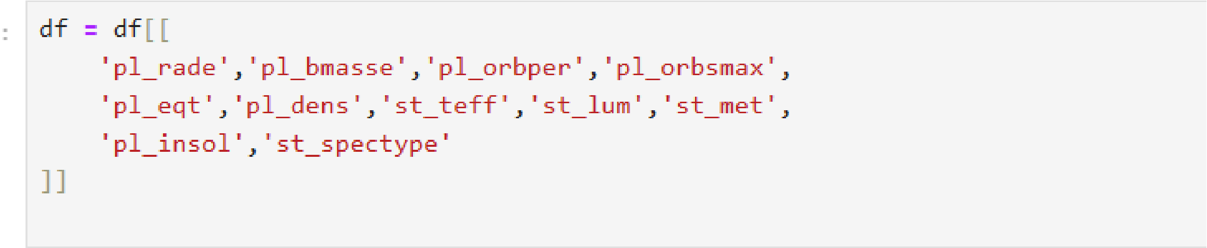
MISSING VALUE HEATMAP



```
plt.figure(figsize=(10,5))
sns.heatmap(df.isnull(), cbar=False)
plt.title("Missing Value Heatmap")
plt.show()
```

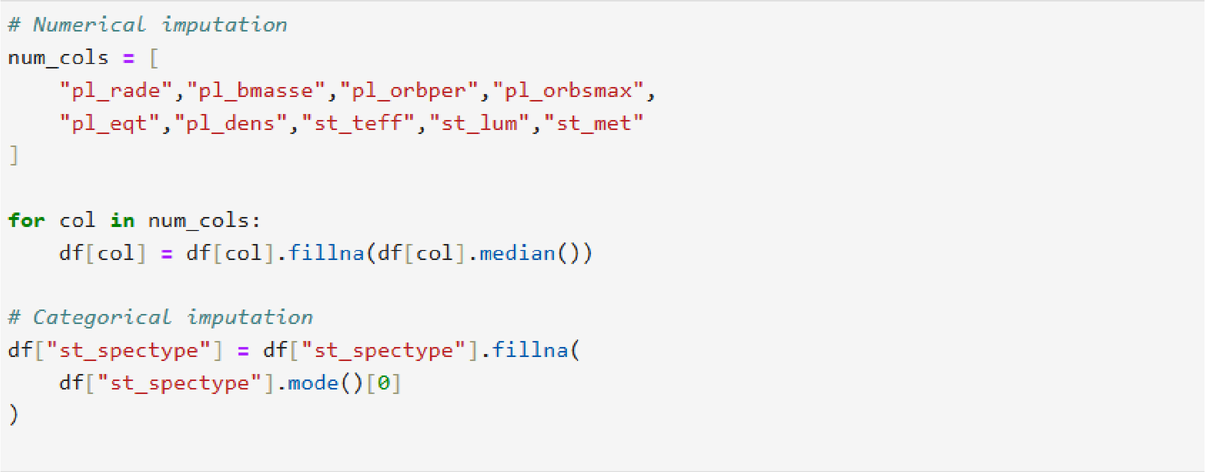


SELECTING RAW FEATURES



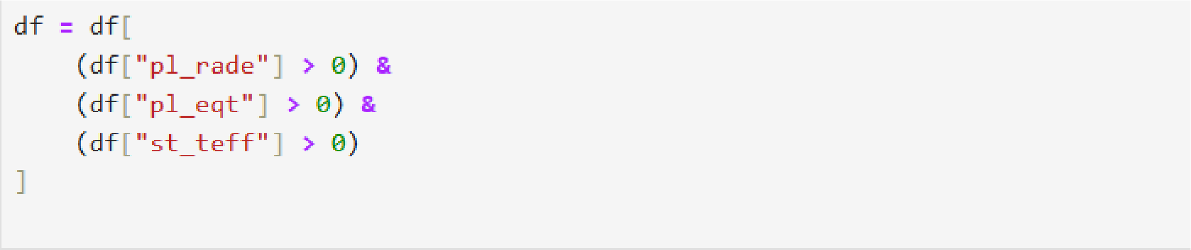
```
:  df = df[[
       'pl_rade','pl_bmasse','pl_orbper','pl_orbsmax',
       'pl_eqt','pl_dens','st_teff','st_lum','st_met',
       'pl_insol','st_spectype'
   ]]
```

HANDLING MISSING DATA

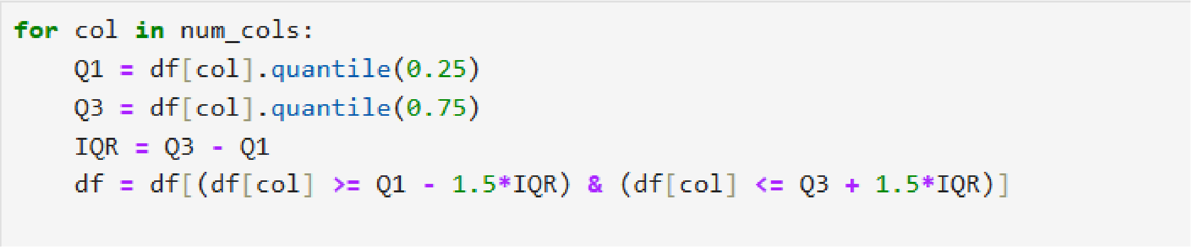


```
# Numerical imputation
num_cols = [
    "pl_rade","pl_bmasse","pl_orbper","pl_orbsmax",
    "pl_eqt","pl_dens","st_teff","st_lum","st_met"
]

for col in num_cols:
    df[col] = df[col].fillna(df[col].median())

# Categorical imputation
df["st_spectype"] = df["st_spectype"].fillna(
    df["st_spectype"].mode()[0]
)
```

OUTLIER DETECTION AND REMOVAL

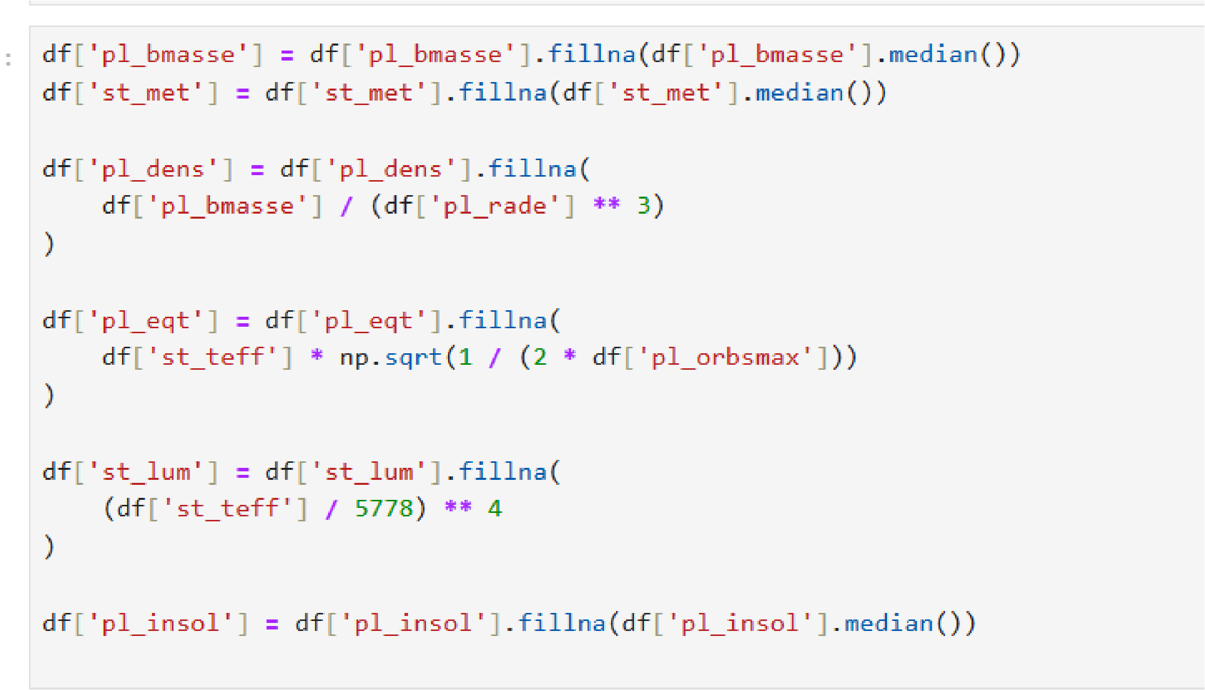


```
df = df[
    (df["pl_rade"] > 0) &
    (df["pl_eqt"] > 0) &
    (df["st_teff"] > 0)
]
```



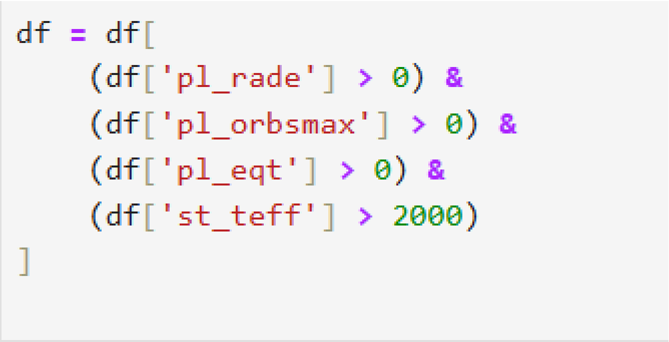
```
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    df = df[(df[col] >= Q1 - 1.5*IQR) & (df[col] <= Q3 + 1.5*IQR)]
```

IMPUTING SCIENTIFICALLY VALID VALUES



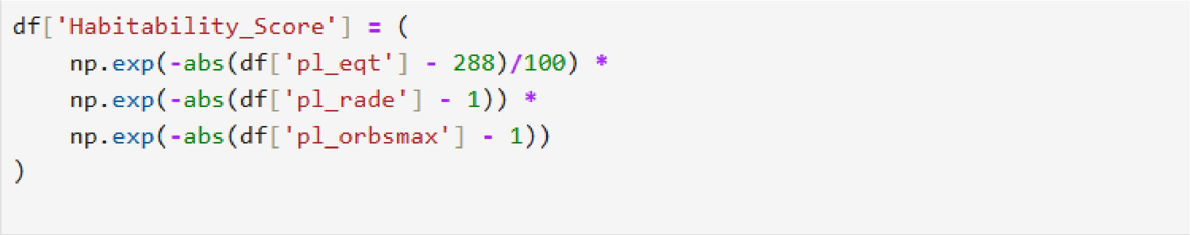
```
df['pl_bmasse'] = df['pl_bmasse'].fillna(df['pl_bmasse'].median())
df['st_met'] = df['st_met'].fillna(df['st_met'].median())

df['pl_dens'] = df['pl_dens'].fillna(
    df['pl_bmasse'] / (df['pl_rade'] ** 3)
)

df['pl_eqt'] = df['pl_eqt'].fillna(
    df['st_teff'] * np.sqrt(1 / (2 * df['pl_orbsmax']))
)

df['st_lum'] = df['st_lum'].fillna(
    (df['st_teff'] / 5778) ** 4
)

df['pl_insol'] = df['pl_insol'].fillna(df['pl_insol'].median())
```

REMOVING PHYSICALLY IMPOSSIBLE VALUES

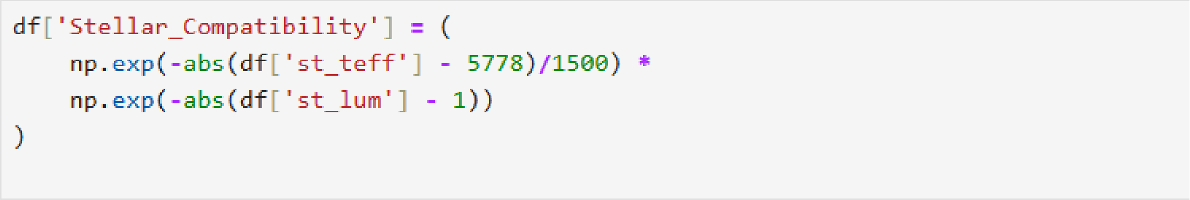


```
df = df[
    (df['pl_rade'] > 0) &
    (df['pl_orbsmax'] > 0) &
    (df['pl_eqt'] > 0) &
    (df['st_teff'] > 2000)
]
```

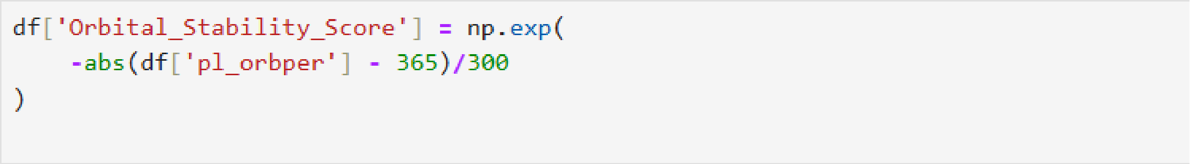
FEATURE ENGINEERING



```
df['Habitability_Score'] = (
    np.exp(-abs(df['pl_eqt'] - 288)/100) *
    np.exp(-abs(df['pl_rade'] - 1)) *
    np.exp(-abs(df['pl_orbsmax'] - 1))
)
```

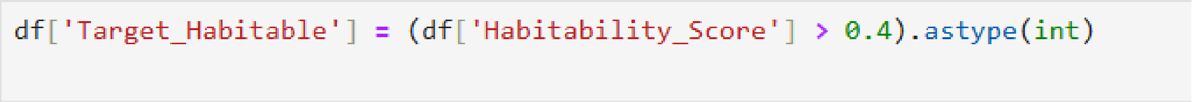


```
df['Stellar_Compatibility'] = (
    np.exp(-abs(df['st_teff'] - 5778)/1500) *
    np.exp(-abs(df['st_lum'] - 1))
)
```



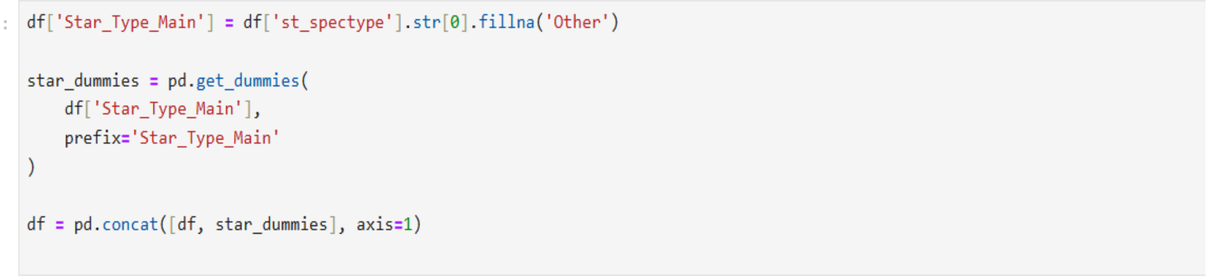
```
df['Orbital_Stability_Score'] = np.exp(
    -abs(df['pl_orbper'] - 365)/300
)
```

TARGET VARIABLE

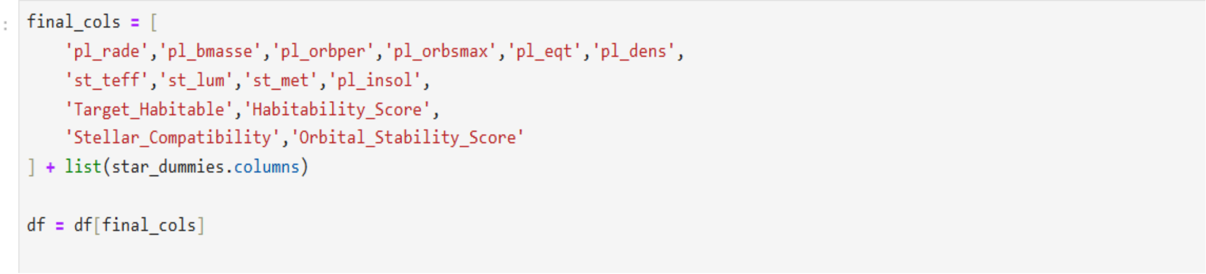


```
df['Target_Habitable'] = (df['Habitability_Score'] > 0.4).astype(int)
```

STAR TYPE ENCODING AND FINAL COLUMNS

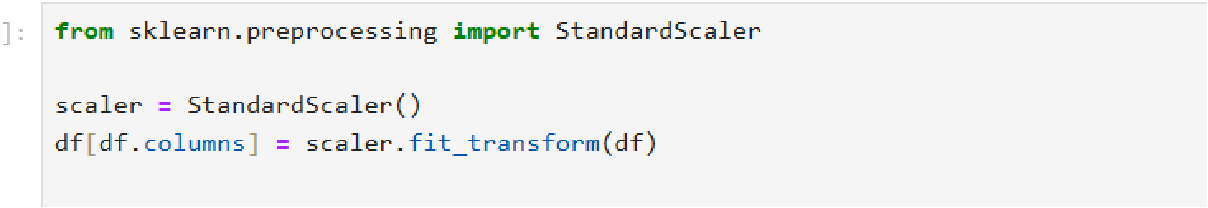


```
df['Star_Type_Main'] = df['st_spectype'].str[0].fillna('Other')

star_dummies = pd.get_dummies(
    df['Star_Type_Main'],
    prefix='Star_Type_Main'
)

df = pd.concat([df, star_dummies], axis=1)
```



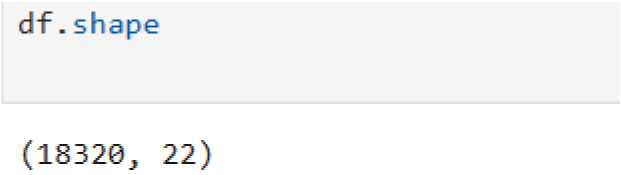
```
final_cols = [
    'pl_rade','pl_bmasse','pl_orbper','pl_orbsmax','pl_eqt','pl_dens',
    'st_teff','st_lum','st_met','pl_insol',
    'Target_Habitable','Habitability_Score',
    'Stellar_Compatibility','Orbital_Stability_Score'
] + list(star_dummies.columns)

df = df[final_cols]
```

STANDARDIZE



```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[df.columns] = scaler.fit_transform(df)
```

CHECK SHAPE



```
df.shape
```

```
(18320, 22)
```