Exo-Habitability-Ranker

Rohan Gharate*

June 29, 2025

*Repository: `https://github.com/Martian-maker/exo-habitability-ranker`

Work carried out independently; no institutional endorsement implied.

Abstract

This report documents the design and implementation of the Exo-Habitability-Ranker, a web application that ranks known exoplanets by a computed habitability score using the TOPSIS method. We detail data ingestion from the NASA Exoplanet Archive, preprocessing steps including feature selection and normalization, implementation of the TOPSIS algorithm, and the interactive dashboard frontend. Results include ranked tables and comparative radar plots.

Contents

1 Introduction

The field of exoplanet discovery has witnessed exponential growth over the past two decades, driven primarily by advanced observational technologies and extensive space missions like NASA's Kepler and TESS. With thousands of exoplanets now confirmed, astronomers face a compelling challenge: quantitatively assessing these distant worlds' potential habitability. A robust, reproducible method for ranking exoplanets by their likelihood to support life is indispensable for prioritizing targets for future studies and resource allocation.

To address this need, the Exo-Habitability-Ranker has been developed as an accessible, open-source, and interactive web application. Leveraging the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), a widely recognized multi-criteria decision-making method, this platform systematically evaluates exoplanets using critical parameters indicative of habitability—such as planetary radius, equilibrium temperature, and incident stellar flux.

By integrating data directly from the NASA Exoplanet Archive, the application ensures accuracy and consistency in its evaluations. The interactive nature of the dashboard empowers researchers, educators, and students to engage deeply with the dataset, adjust evaluation criteria dynamically, and visualize comparative habitability assessments through intuitive graphical representations.

Ultimately, the Exo-Habitability-Ranker not only facilitates informed decision-making in planetary science but also serves as an educational resource that enhances public understanding and engagement with exoplanetary research.

2 Repository Architecture

Figure 1 shows the architecture of the repository or the general file tree for the result in picture.

```
exo-habitability-ranker
 |--- .devcontainer
 |--- .gitignore
 |--- README.md
 |--- data #Folder
  |--- ingest.py          # Data ingestion module from NASA's dataset
  |--- preprocess.py      # Data preprocessing & TOPSIS
  |--- app.py             # Streamlit dashboard
  |--- requirements.txt   # Python dependencies
  |--- ps_latest.csv      # Cached data (not stored in repo forever)
```

Figure 1: Repository file structure

3 Mathematical and Scientific Foundations

We denote the feature matrix $X \in \mathbb{R}^{n \times m}$, where n is the number of planets and m is the number of features: here, radius (r), equilibrium temperature (T), and incident flux (F). The Earth-reference vector is:

$$\mathbf{e} = [r_\oplus, T_\oplus, F_\oplus] = [1.0, 288.0, 1.0] \tag{1}$$

After ingestion and cleaning, we compute the flux:

$$F = \frac{10^{\log(\frac{L_*}{L_\odot})}}{d^2} \tag{2}$$

where L_* is the host star luminosity logarithm relative to the Sun. In the data set, it is the column named: `st_lum`, and d is the orbital semi-major axis of the subjected planet. For this, column is named: `pl_orbsmax`.

Now, it was necessary to see deviation compared to earth. Since radius and flux is in earth-radii and earth flux, respectively, but the temperature is an independent quantity. we needed to bring that in relation to the earth too. Hence, we used a general formula that works for all units to calculate the absolute difference:

$$\Delta = \left| \frac{X}{e} - 1 \right| \tag{3}$$

Where, the the division is element-wise.

$$X = \{X_{ij}\} \quad , \quad e = \{e_j\} \quad , \quad \Delta = \{\Delta_{ij}\}$$

Hence, for $j^{\texttt{th}}$ feature of $i^{\texttt{th}}$ planet :

$$\Delta_{ij} = \left| \frac{X_{ij}}{e_j} - 1 \right|, \tag{4}$$

To compute the TOPSIS score and rank exoplanets, series of mathematical steps are performed on the absolute difference for maximum accuracy.

3.1 Normalized matrix ($\tilde{\mathbf{X}}$)

After computing the absolute difference matrix Δ, each column is normalized independently to form the normalized decision matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times m}$. Radius, temperature, and flux all have a different relation to the Earth. They all have different scale sizes to change and a different range. In this case, it is fairly possible that a factor such as temperature with a wide range and comparatively huge values might dominate unreasonably. Hence, this step ensures that all features—regardless of their original units or numeric ranges—are transformed onto a comparable, unit-less scale. With every factor on same scale and in same unit, further process become fair and more accurate.

For each feature j, we compute the Euclidean norm across all planets i :

$$\|\Delta_{\cdot j}\| = \sqrt{\sum_{i=1}^{n} \Delta_{ij}^2}$$

and normalize every entry as:

$$\tilde{X}_{ij} = \frac{\Delta_{ij}}{\|\Delta_{\cdot j}\|} \tag{5}$$

This normalization is critical for avoiding bias in the ranking process: without it, features with larger numeric values (such as stellar flux) could dominate the TOPSIS calculations simply due to scale. By enforcing unit consistency and scale equality across all columns, $\tilde{\mathbf{X}}$ guarantees that each feature contributes proportionally to the final habitability score, in accordance with its assigned weight.

3.2 Weight (w)

In the weighted matrix $\mathbf{V}$, we apply a predefined weight vector $\mathbf{w} = [0.4,\ 0.3,\ 0.3]$ corresponding to the three habitability features: planetary radius (r), equilibrium temperature (T), and incident stellar flux (F), respectively. These weights reflect the relative importance of each feature in assessing Earth-like habitability conditions.

Planetary radius is assigned the highest weight (0.4) because it strongly constrains the physical nature of a planet. Planets significantly smaller than Earth may lack the gravitational strength to retain a stable atmosphere, while those significantly larger may be gas giants without solid surfaces—both conditions being unfavorable for Earth-like life. Radius therefore serves as an effective early-stage filter, excluding clearly uninhabitable candidates before deeper atmospheric or thermal analysis.

Equilibrium temperature and stellar flux are each weighted at 0.3. These two features directly influence the planet's surface energy budget and potential for maintaining liquid water—arguably the most critical ingredient for life as we know it. While both are vital, they are somewhat correlated, and their effects on habitability are better interpreted together than in isolation. By assigning equal weight to T and F, the model avoids overemphasizing either energy intake or thermal equilibrium individually, instead capturing their combined role in surface habitability.

It is important to note that these weights are domain-informed but not universal; different weighting schemes may be appropriate depending on the assumptions of the habitability model. Future extensions of this work may explore adjustable or dynamic weights to reflect different biological or environmental priorities.

3.3 Weighted Matrix Set (V)

The Weighted Matrix is where we scale the normalized matrix so the important features have more power in scoring the exoplanets. Here, we perform the dot product between the weight vector with the normalized matrix vector for each feature of every planet. Hence, each value in the dataset now become V_{ij} and the dataset can be represented by V as $V = \{V_{ij}\}$. In conclusion,

$$V_{ij} = w_j \cdot \tilde{X_{ij}} \tag{6}$$

Where, $\quad j \in 1, 2, 3 \quad \forall i \in [1, n] \cap \mathbb{Z}$.

Now, the dataset is correctly valued and scaled to the desired and correct values obtaining the weighted matrix set (V). Where, $V = \{V_{ij}\}$. Moving further, it is a pure analysis and scoring.

3.4 The Ideal Set (I)

The Ideal Set(I) is a set containing two matrices- the ideal best and the ideal worst. Hence, it is the ideal set.

$$I = \{V^+, V^-\}$$

for the ideal best (V^+) we pick the minimum value from each of the 3 column. The minimum value for a feature means that the feature is least different from the earth. Hence, finding the minimum in each column and inserting it into the matrix makes it the matrix for the ideal best i.e most similar to the earth. Similarly, the maximum values for each featrure represent the features least similar to the earth used for the ideal worst (V^-). Therefore,

$$V^+ = min(V_{ij}) \tag{7}$$

$$V^- = max(V_{ij}) \tag{8}$$

3.5 Distance Set (C)

The distance set contain two variables of extreme importance to this study. First, the distance from the ideal best (C_i^+) stores how different/distant is a planet (i) from the ideal best. We may call it the 'best distance'. Second, the distance from the ideal worst (C_i^-)

stores how different/distant is a planet (i) from the ideal worst. We may call it the 'worst distance'.

$$C_i = \{C_i^+, C_i^-\}$$

Each Weighted Matrix(V_{ij}), here, is treated as a vector with 3 dimensions i.e. $j = 1, 2, 3$. The vectors to find the distance to are the ideal best(V^+) and ideal worst(V^-) vectors in the Ideal set; each with 3 dimensions as well. Now, we calculate the Euclidean distance between **V** and **I**.

$$C_i^+ = \sqrt{\sum_j (V_{ij} - V_j^+)^2} \tag{9}$$

$$C_i^- = \sqrt{\sum_j (V_{ij} - V_j^-)^2} \tag{10}$$

Finally, for clean matters, $C = \{C_i\}$. Now we have a data set- C. Further scoring depends on the values in C.

3.6 Score Set (S)

Here, the decision is up to the fact that we wanted the ranking to be in a descending order; meaning the planet at the top is the most suitable of exoplanets. The top one wins while the bottom one looses. Hence, we find how far is a planet from the ideal worst if V_i^+ and V_i^- are the two end points on a segment and the planet (i) lies on that segment. The further it is from V_i^-, the higher it ranks. This, eventually, also means that the planet is that closer to V_i^+. We do not know where this segment ends. It is different for each planet because $\sum_{c \in C_i} c \neq 1$. However, there is another segement with the total length 1. This segment that we assumed is from 0 to 1 and every score lies in between- a set of all integers between 0 to 1. the score for a single planet(i) is S_i, while S is the data set with all the scores, probably even arranged in descending order. However, the arrangement does not matter when looking at the Score Set as we have the Rank Set for the arranged scores.

$$S = \{S_i\}$$

Given that we decided the rank to be in a descending manner with the highes score at the top as the winner, for the score of a single planet- S_i- we divide the distance from the worst ideal by the sum of both the distances- $\sum_{c \in C_i} c$.

$$S_i = \frac{C_i^-}{\sum_{c \in C_i} c} \tag{11}$$

Where,

$$\sum_{c \in C_i} c = C_i^+ + C_i^-$$

$$S_i \in [0, 1]$$

3.7 Rank Set (R)

The rank set is the sequence of scores S_i arranged in descending order:

$$\mathbf{R} \;=\; \big(S_{(1)}, S_{(2)}, \ldots, S_{(n)}\big), \qquad S_{(1)} \geq S_{(2)} \geq \cdots \geq S_{(n)}.$$

R displays the ranking column for the table in the Exo-habitability-ranker.

4 Programming and Code Snippets

With the algorithm created using the mathematical and scientific base, we proceed to code the application. Here, the first step is to ingest/get the data from a trusted source. Second step is to process the data- this is where the mathematical algorithm works. Final step is to display the data on the Exo-habitability-ranker's UI.

4.1 Python dependencies

The dashboard and data-processing pipeline require the packages below. Version pins correspond to the latest stable releases at the time of writing and match the accompanying `requirements.txt` file.

Package	Purpose
`pandas 1.5.3`	CSV ingestion, tabular joins, caching
`numpy 1.24.4`	Vectorised computation for Δ, $\tilde{\mathbf{X}}$, TOPSIS math
`matplotlib 3.8.4`	Habitability radar plots and ancillary figures
`streamlit 1.33.0`	Interactive web front-end
`requests 2.31.0`	HTTP download from the NASA TAP endpoint
`scipy 1.11.4`	Numerical helpers (*optional*; used for robust stats)

Install everything with:

```
python -m venv venv
source venv/bin/activate          # Windows: venv\Scripts\activate
pip install -r requirements.txt   # or install packages listed above
```

4.2 Ingest

The data is imported by the file named- ingest.py in the backend. Data are ingested from the NASA Exoplanet Archive (Akeson et al. 2013) via the TAP endpoint, selecting the entire Planetary System Composite Parameters(pscomppars) table in CSV format.

```
# data/ingest.py
"""
Download the most-recent Planetary Systems Compiled Parameters (pscomppars) table
from the NASA Exoplanet Archive and save it as ps_latest.csv next to this script.

Usage:
    from ingest import download_nasa_data
    csv_path = download_nasa_data()
"""

import os
import datetime as dt
import pandas as pd

URL = (
    "https://exoplanetarchive.ipac.caltech.edu/TAP/sync?"
    "query=select+*+from+pscomppars&format=csv"
)

def download_nasa_data():
    """Download the archive CSV (once per day) and return the local file path."""
    dir_path = os.path.dirname(os.path.abspath(__file__))
    csv_path = os.path.join(dir_path, "ps_latest.csv")

    # Skip download if we fetched it today already
    if os.path.exists(csv_path):
        mtime = dt.datetime.fromtimestamp(os.path.getmtime(csv_path))
        if mtime.date() == dt.date.today():
            return csv_path  # up-to-date

    # Download via pandas (handles gzip automatically)
    print("Downloading latest exoplanet table ...")
    df = pd.read_csv(URL)
    df.to_csv(csv_path, index=False)
    print(f"Saved {len(df):,} rows to {csv_path}")
    return csv_path


# When run directly: grab the file and report size
if __name__ == "__main__":
    p = download_nasa_data()
    print("Local file:", p)
```

4.3 Mathematical Analysis and Sorting

Here, in preprocess.py, we perform the mathematical analysis using the developed TOPSIS algorithm, sort the data, and rank the exoplanets.

```
import pandas as pd
import pandas as pd
import numpy as np
import os
import pathlib

def process_exoplanets(input_path):
    df = pd.read_csv(input_path)

    # Keep & rename only the columns we need
    df = df.rename(columns={
        "pl_name": "pl_name",
        "pl_rade": "radius",
        "pl_eqt": "temp",
        "st_teff": "star_temp",
        "st_lum": "star_lum",
        "pl_orbsmax": "orb_distance"
    })

    # Drop rows with missing required fields
    df.dropna(subset=["pl_name", "radius", "temp", "star_temp", "star_lum",
    ↪  "orb_distance"], inplace=True)

    # Compute incident flux (relative to Earth)
    df["flux"] = (10**(df["star_lum"])) / (df["orb_distance"] **2)
    #star_lum here is a logarithmic tar luminosity relative to the sun so
    ↪  antiloging first

    # Prepare for TOPSIS
    features = ["radius", "temp", "flux"]
    X = df[features].copy()

    #X = (X - X.mean()) / X.std() -- z-score (this is the part of the previous
    ↪  code)

    # Normalize relative to Earth and take absolute difference
    # Earth-reference vector (R, T in K, F)
    earth = np.array([1.0, 288.0, 1.0])
    # Fractional distance from Earth in each dimension
    X = np.abs(X.values / earth - 1.0)

    # TOPSIS function
    def topsis(matrix, weights, benefit):
```

```
        norm = matrix / np.sqrt((matrix ** 2).sum(axis=0))
        weighted = norm * weights
        ideal = np.where(benefit, weighted.max(0), weighted.min(0))
        nadir = np.where(benefit, weighted.min(0), weighted.max(0))
        d_pos = np.linalg.norm(weighted - ideal, axis=1)
        d_neg = np.linalg.norm(weighted - nadir, axis=1)
        score = d_neg / (d_pos + d_neg)
        return score

    weights = np.array([0.4, 0.3, 0.3])      # radius, temp, flux
    #benefit = np.array([False, False, True]) -- flux is \more is better" (part
    ↪  of previous code)
    benefit = np.array([False, False, False]) # small difference from earth is
    ↪  better"

    df["habit_score"] = topsis(X, weights, benefit)
    return df.sort_values("habit_score", ascending=False)
```

4.4 Display

The file- app.py- handles the display of data on the webapp and User interaction with the application. This file also saves the analysed data as a temporary catch so user can pull the data only once per session and not by every interaction in that session. This let's user interact with exoplanet score range, plot radar diagram, and download the complete displayed data as a .csv file.

```
import streamlit as st
st.set_page_config(page_title="Exoplanet Habitability Dashboard", layout="wide")
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os


from ingest import download_nasa_data
from preprocess import process_exoplanets

@st.cache_data
def get_processed_data():
    csv_path = download_nasa_data()
    return process_exoplanets(csv_path)

df = get_processed_data()
```

```
# === Header ===
st.title("Exoplanet Habitability Dashboard")
st.markdown("""
This dashboard ranks known exoplanets by a machine-calculated habitability score
↪  using the **TOPSIS** method.
You can filter, explore, and download the results.
""")

# === Filter by score ===
score_min = st.slider("Minimum Habitability Score", 0.0, 1.0, 0.85, step=0.01)
filtered = df[df["habit_score"] >= score_min].reset_index(drop=True)

# === Table ===
st.markdown(f"### {len(filtered)} Planets with Habitability  {score_min}")
st.dataframe(
    filtered[["pl_name", "radius", "temp", "flux",
    ↪  "habit_score"]].rename(columns={
        "pl_name": "Planet Name",
        "radius": "Radius (Earth Radii)",
        "temp": "Surface Temperature (K)",
        "flux": "Flux(Earth Flux)",
        "habit_score": "Habitability Score"
    }),
    use_container_width=True
)

normalize_by = st.radio(
    "Choose normalization method:",
    ["Dataset Range", "Earth Reference"],
    index=0,
    horizontal=True
)

# === Select planet for radar plot ===
planet_name = st.text_input("Type a planet name from the list as printed for its
↪  " \
"comparative radar plot. (Note: For dataset range Percent scales reflect position
↪  within the entire exoplanet dataset, " \
"not Earth-like scale. Extreme planets (e.g., hot Jupiters) shift the range. For
↪  Earth reference, percentage represent " \
"how similar it is to the Earth- relative to the earth (e.g., x percent of the
↪  earth's radius)).", placeholder="e.g. Kepler-22 b")

if planet_name:
    match = df[df["pl_name"].str.lower() == planet_name.strip().lower()]
```

```
if match.empty:
    st.warning("Planet not found. Please check the name or spelling.")
else:
    row = match.iloc[0]
    labels = ["radius", "temp", "flux"]
    missing = [c for c in labels if c not in row]
    if missing:
        st.error(f"Missing column(s) for radar plot: {', '.join(missing)}")
        st.stop()
    else:
        values = [row[c] for c in labels]

    earth_values = [1.0, 288.0, 1.0]  # Earth baseline


    if normalize_by == "Dataset Range":
        # Normalize using min-max of dataset
        min_vals = df[labels].min().values
        max_vals = df[labels].max().values
        scaled_values = (np.array(values) - min_vals) / (max_vals - min_vals)
        scaled_earth = (np.array(earth_values) - min_vals) / (max_vals -
        ↪  min_vals)
    else:
        # Normalize relative to Earth
        scaled_values = np.array(values) / np.array(earth_values)
        scaled_earth = np.ones_like(scaled_values)
        # Cap all values at 2.0 (200%) to avoid chart overflow
        scaled_values = np.clip(scaled_values, 0, 2.0)

    # Radar prep
    scaled_values = np.append(scaled_values, scaled_values[0])
    scaled_earth = np.append(scaled_earth, scaled_earth[0])
    angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False).tolist()
    angles += angles[:1]

    fig, ax = plt.subplots(figsize=(5, 5), subplot_kw={'polar': True})
    ax.plot(angles, scaled_values, label=row["pl_name"], linewidth=2)
    ax.fill(angles, scaled_values, alpha=0.3)
    ax.plot(angles, scaled_earth, label="Earth", linestyle="--",
    ↪  color="gray")
    ax.fill(angles, scaled_earth, alpha=0.1, color="gray")

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(labels)

    # Show proper radial labels
    if normalize_by == "Dataset Range":
```

```
            ax.set_yticks([0.25, 0.5, 0.75, 1.0])
            ax.set_yticklabels(["25%", "50%", "75%", "100%"])
        else:
            ax.set_yticks([0.5, 1.0, 1.5, 2.0])
            ax.set_yticklabels(["50%", "100%", "150%", "200%"])

        ax.legend(loc="upper right", bbox_to_anchor=(1.3, 1.1))
        st.pyplot(fig)



# === Download CSV ===
st.markdown("### Download Full Dataset")
st.download_button("Download ranked_planets.csv", df.to_csv(index=False),
↪  file_name="ranked_planets.csv")
```

5 Result

The result is the github repository and the webapp in cotext. the webapp can be accessed at https://exo-habitability-ranker.streamlit.app/. The github Repository can be accessed at https://github.com/martian-maker/exo-habitability-ranker/blob/main/data

6 Limitations and Future Work

Current limitations: dependence on completeness of NASA archive, simple feature set, static weights. We are unsure about the future work as of this time. However, there is a scope to further improvise the exo-habitability-ranker and add new feature as to get the data accurate and use it in more ways than we do at the time of preparing this document.

7 Conclusion

We have presented a an exoplanet habitability ranking tool. This tool ranks the exoplanets in data in the descending order of its habitability based on TOPSIS. The score is decided on several feature which makes it relative. The ideal for TOPSIS comparison is Earth and features that are worked on are radius of the planet, equilibrium temperature, and the flux of the planet. The data in ingested freshly and stored as catch for that particular user session, hence it is the latest dat user works with on the app. then it is process with the mathematical algorithms and displays on the app as an output. There are a few limitations to this project as discussed above. Yet, it is the tool for quite an exploratory amusement and the work of accuracy.

8 Glossary

n	Number of planets (rows in the data matrix).
m	Number of habitability features (here $m = 3$: r, T, F).
i	Planet (row) index, $i \in \{1, \ldots, n\}$.
j	Feature (column) index, $j \in \{1, 2, 3\}$.
$\mathbf{X}$	Raw feature matrix $\in \mathbb{R}^{n \times m}$ (one row per planet).
X_{ij}	Value of feature j for planet i in $\mathbf{X}$.
$\mathbf{e}$	Earth-reference vector [1, 288, 1] for (r, T, F).
e_j	Earth's value for feature j.
Δ	Absolute difference matrix $\big\lvert(\mathbf{X}/\mathbf{e}) - 1\big\rvert$.
Δ_{ij}	Deviation of planet i from Earth in feature j.
$\tilde{\mathbf{X}}$	Normalized decision matrix (column-wise Euclidean norm).
$\tilde{X}_{ij}$	Normalized, unit-less deviation value.
$\mathbf{w}$	Feature weights [0.4, 0.3, 0.3] for (r, T, F).
w_j	Weight for feature j.
$\mathbf{V}$	Weighted normalized matrix.
V_{ij}	Weighted value for planet i, feature j, $V_{ij} = w_j \tilde{X}_{ij}$.
$\mathbf{V}^+$	*Ideal best*: column-wise minima of $\mathbf{V}$.
$\mathbf{V}^-$	*Ideal worst*: column-wise maxima of $\mathbf{V}$.
$\mathbf{I}$	Ideal set $\{\mathbf{V}^+, \mathbf{V}^-\}$.
C_i^+	Distance of planet i to the ideal best.
C_i^-	Distance of planet i to the ideal worst.
$\mathbf{C}_i$	$\{C_i^+, C_i^-\}$ for planet i.
$\mathbf{C}$	Distance set $\{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$.
S_i	Score of planet i $C_i^-/(C_i^+ + C_i^-) \in [0, 1]$.
$\mathbf{S}$	Unordered score set $\{S_1, \ldots, S_n\}$.
$\mathbf{R}$	Rank set $(S_{(1)}, \ldots, S_{(n)})$ sorted in descending order.
r	Planetary radius (Earth-radii).
T	Equilibrium temperature (Kelvin).
F	Incident stellar flux (Earth-flux).
$L_\star$	Host-star luminosity ($\log L_\star / L_\odot$).
d	Orbital semi-major axis.
TAP	Table Access Protocol (International Virtual Observatory Alliance (IVOA) standard).
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution.
`pscomppars`	"Planetary System Composite Parameters" catalogue in NASA Exoplanet Archive.

9 Acknowledgment

This work was carried out independently and does not necessarily reflect the views of the University of Arizona.

10 References

References

[1] A. Akeson et al., "NASA Exoplanet Archive," *Astronomical Data Analysis Software and Systems*, 2013.

[2] Hwang et al., "Application of TOPSIS in Exoplanet Habitability Ranking," *MNRAS*, 2020.