**AI-Astro Larger Project**

**STEMx 2025 @LUMS**

**Supervisor:** Dr. Aquib Moin

**Students:** Al-Astro Group

**Project Area:** Observational Studies of Exoplanets Using Space-based Data

**Project Title:** *“Predictions for Exoplanet Earth Similarity Index (ESI) Using Unsupervised Machine Learning”*

**Purpose:** Create an end-to-end cloud-based supervised machine learning solution to predict how similar the exoplanets in a large dataset are to Earth, which would be tedious and time-consuming if done manually.

**Task list:**

**Task-1:** Self Study:

What are Exoplanets?

How do we observe them?

Why are they studied?

**Task-2:** Familiarization with the Exoplanet data:

**Training data:** Planetary Habitability Laboratory (PHL) at Arecibo:

Examine the dataset - <https://phl.upr.edu/home>

Data for prediction: NASA Planetary Systems Composite Parameters (PSCP)

Examine the dataset - <https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PSCompPars>

Preprocessed data files (csv format) will be provided.

**Task-3:** Problem Definition:

From an Exoplanet database containing thousands of records, how can we “smartly” figure out if some of the Exoplanets are “Earth-like” and could potentially sustain life?

**Task-4:** Solution: Supervised Machine Learning:

**Step-1:** Upload pre-processed exoplanet data.

What’s happening:

In the sidebar, users can either:

Upload their own CSV file (with no missing values), or

Toggle on an example dataset from the Habitable Worlds Catalog (hosted on Google Drive).

The app reads this data into a pandas DataFrame so it can be used for training the ML model.

**Step-2:** Build a “**Random Forest Regression**” model [for supervised machine learning].

What’s happening:

Once the dataset is ready:

The app splits the data into training and test sets based on a slider (e.g., 80% for training).

Users configure parameters for a Random Forest (number of decision trees, max features, etc.).

The model is trained using .fit() with input features X and target variable y.

This is the supervised learning part where the model learns how the features relate to the target value.

**Step-3:** Predict the target variable: Earth Similarity Index

What’s happening:

After training, the model uses .predict() to estimate the target variable:

On the training set (to check how well it learned), and

On the test set (to see how it performs on new, unseen data).

The app shows a comparison of actual vs. predicted values in a table and a scatter plot to visualize performance.

**Step-4:** Evaluate model performance (model accuracy).

What’s happening:

The app calculates and displays standard statistical regression metrics:

MSE (Mean Squared Error): **MSE** measures how far off the model’s predictions are from the actual values. It’s calculated by taking the average of the squares of the differences between predicted and actual values — lower is better.

R² Score (Coefficient of Determination) R² measures how well the model's predictions explain the variation in the actual target values.It’s calculated by comparing the squared differences between the actual values and the model’s predictions to the squared differences between the actual values and their mean — closer to 1 is better.

It also shows:

A bar chart of feature importance, highlighting which input features the model found most useful.

These results help users understand how well the model is performing and what features matter most.

**Step-5:** Save the trained model.

What’s happening:

After training:

The model is saved using joblib.dump() into a file called rf\_model.joblib.

A download button appears so users can download and reuse the trained model later, even outside the app.

This is useful if users want to apply the model to other data or load it into another script.

**Step-6:** Apply it to a new dataset (previously unseen by the model) for prediction.

What’s happening:

Users can upload a new CSV file with the same feature columns used during training.

The app checks for missing or mismatched columns.

It loads the saved model, makes predictions using .predict(), and adds the results to the new data.

The final prediction table is displayed and offered as a downloadable CSV.

**What is Random Forest Regression model?**

A Random Forest Regression model is a supervised (labeled data) machine learning technique used for regression, which in statistics refers to modeling the relationship between a dependent variable (the value we want to predict) and one or more independent variables (the input features). In simple terms, regression helps us understand how changes in input features affect the output value. Random Forest Regression does this by combining many decision trees, which are like yes/no flowcharts that split the data by asking simple questions (e.g., “Is the star mass > 1?”) and make a numerical prediction at the end. While a single decision tree may overfit or give unstable results, a random forest builds many trees using random subsets of the data and features. Each tree makes its own prediction, and the forest averages these predictions to produce a final result that is more accurate, stable, and generalizable — like gathering the opinions of many experts and averaging their answers.

**Task-5:** Now how do we implement this?

We will deploy a Python-based end-to-end web-app on Streamlit Cloud which will “host” the model, provide the endpoint and the interface for the model to operate on the data. The code will be written in GitHub Codespaces environment and the code repository will be maintained on GitHub.

A full demo will be provided, and you will be creating your own version. You will produce results for a new dataset.

**Links:**

<https://github.com/> (GitHub Codespaces will be accessed through GitHub)

<https://streamlit.io/>

**Step-1:** Create your GitHub and Streamlit accounts.

**Step-2:** From GitHub, access this repository: <https://github.com/aquibmoinssa/exoplanetml>

**Step-3:** “Fork” (copy) the repository to your own account.

**Step-4:** Go to Streamlit and import your copy of the repository to create your own webapp.

**Step-5:** Troubleshoot if needed and get the app up and running.

**Step-6:** Run the entire ML operation on the exoplanet data files to get ESI predictions.

**Step-7:** Obtain the results and examine the output.

**Step-8:** Study the results and try to get some insights.

**Task-6:** Additional Analysis by plotting key parameters (if time permits).

**Task-7:** Showcase and explain your work.

**Appendix**

Programming Language: Python

IDE: GitHub Codespaces

Code repository: GitHub

Deployment Platform: Streamlit Cloud

**Functionality Modules**

Machine Learning: Scikit-Learn

App Dashboard: Streamlit

Data Access: CSV files with data from PHL @Arecibo

Data Manipulation: Pandas

Data Retrieval: Zipfile

Model Capture: Joblib

Visualization: Altair

Plotting: Matplotlib