Branch : CSE(AIML)
College : SR University
Btech 4th year

I am from Telangana.
Base Experience in Machine Learning and Deep learning.

Module 1 : Data Collection and Management

Week 1 - Day 1 (01/12/25)

- Dataset1 Link : https://www.kaggle.com/datasets/shivamb/all-exoplanets-dataset
    - Observation :
        1. Dataset contains exoplanets discovered till 2021.
        2. 50 columns, 9500 rows.
        3. It likely includes many planets good for training ML models.
    - Useful: Yes it can be used but only if required features exist in its columns.
        1. Some key columns are missing
        2. We can merge data from NASA's Exoplanet Archive.
    - Questions:
        1. Are all required features like planet and stars present?
        2. How many missing values does each important column have?
    - Comments : The dataset is usable is usable but may not be complete for habitability modeling. Values might need cleaning before moving to ML stages.

Dataset2 Link: https://www.kaggle.com/datasets/nasa/kepler-exoplanet-search-results

- Obervations:
    1. objects observed by the Kepler Space Telescope — specifically "Kepler Objects of Interest" (KOIs): candidate exoplanets, confirmed exoplanets, and false positives.
    2. It contains 9564 rows, and 50 columns.
    3. Columns are classified as identifiers like unique star, planet name, steller parameters, some diagnostics , errors in columns for measured values.
    4. Dataset includes a mixture of steller data+transit detection+ metadata about planet candidate.
- Usefulness :
    1. Steller parameters – star temperature, mass, radius, metallicity – are present; that supports part of our feature needs.
    2. It useful for transit-derived parameters.
    3. This is dataset is not primarily designed to provide planet physical properties like radius,mass, density,equilibrium.
- Comments : The dataset has 50 features per KOI, many columns may be irrelevant or redundant for habitability modeling; careful feature selection will be essential.

Week 1- Day 2

- Dataset3 Link : https://www.kaggle.com/datasets/arashnic/exoplanets
    - Observation :
        1. Dataset contains a variety of features including stellar parameters and planet- related parameters.
        2. Stellar effective temperature,stellar radius, stellar mass,and stellar metalicity as inherited from the parent archive.
        3. It has both planet-specific and host-star parameters.
    - Useful: Likely useful, it provides a mix of planetary parameters ,some exoplanets may only have partial data using them might require imputation or filtering.
        1. The dataset include planets with varying levels of confirmation.
        2. Diff sources contribute to the compiled dataset,units might be inconsistent.
    - Questions:
        1. How many rows have all these essential columns non-null?
        2. What fraction of rows are for confirmed planets vs candidates or false positives?
    - Comments : Use this dataset as your primary data source for Module 1 — but first run a data-audit script to examine completeness, null counts, unit consistency, and confirmation status.

WEEk1- Day 3

- Simple Findings about the datasets I have worked on.

- The dataset contains real NASA exoplanet data. It includes many confirmed planets with scientific parameters.
- It has both planet features and star features. This matches what your habitability project needs.
- Most key fields you require are present. Planet radius, mass, density, temperature, orbital period, and many star parameters are available.
- Some planets have missing values.Not all planets will have complete information for ML modeling.
- The dataset is suitable for Machine Learning. It is in CSV form, well-structured, and easy to load into pandas.
- Missing Values, duplicates, or incomplete entries must be handled.
- This makes it a better fit for Module 1 of your project.

- It contains the necessary parameters to calculate or model habitability.

Week 1 - Day 4

Handling Multiple Datasets Without a Habitability Column

- X (features) = planet + star parameters
- Y (label) = Habitability (Habitable / Non-Habitable)

But no dataset on Kaggle directly provides a habitability label, so we must create one using real scientific criteria.

1. Can we apply Logistic Regression on your dataset?

Yes — but only if you create a Y-label (habitability).
The original NASA Exoplanet datasets do not contain habitability labels.

So first, we must label the planets.

2. Why K-means clustering won't work directly

K-means is unsupervised, so it cannot give you a "habitable vs non-habitable" label.
You must first create the habitability label using rules, then you can apply logistic regression.

Week 1- Day 5

Descriptive statistics is the branch of statistics that summarises and describes the main features of a dataset. It does not make predictions or generalisations; it only explains what the data shows.

Core components of descriptive statistics:

Measures of Central Tendency
Mean, median, mode.
These describe the centre of the data.

Measures of Dispersion (Spread)
Range, variance, standard deviation, interquartile range.
These show how spread out the data is.

Measures of Shape
Skewness (left/right tilt), kurtosis (peaked/flat).

Tabular & Graphical Summaries
Frequency tables, histograms, bar charts, box plots, pie charts.
These visually explain data patterns.

Purpose:
To give a clear, simple summary of what the data looks like, making it easier to understand patterns before doing further analysis.

Difference between percentage and percentile, and how they relate to a distribution:

1. Percentage
   A percentage is just a number out of 100.It answers: “How many out of 100?”
   Example: 45% means 45 out of 100.
2. Percentile.
   A percentile is a position in a distribution.
   It answers: “What percentage of data lies below this value?”
   Example: If you are in the 90th percentile in an exam, 90% of students scored lower than you.

Relationship to distribution:
A percentile only makes sense when the data is arranged in a distribution (sorted from lowest to highest). It tells you the relative standing of a value within that distribution.
A percentage does not require any distribution; it is just a proportion.

Correlation is straightforward in regression because both X and Y are numeric.
In classification, Y is categorical, so standard correlation (like Pearson) is not directly applicable.
In summary,
Regression → correlation = linear relationship
Classification → correlation = strength of association / contribution to separating classes

Week 2- Day 1(08/12/25)

https://www.kdnuggets.com/2020/01/exoplanet-hunting-machine-learning.html

Reading this – it's helping explore the universe. The idea that we can program a computer to detect planets around distant stars, by learning from patterns humans may struggle to see, feels like science-fiction turned real. Also, implementing such a project in Python (with real astrophysics data) would be a great learning experience — combining data prep, ML techniques, domain knowledge, and a sense of contributing to real astronomy.

Task

Exoplanet Detection and Habitability Analysis

1. Overview

Researchers use machine learning to detect exoplanets by studying light-curve data, which shows how a star's brightness changes over time. When a planet passes in front of the star, there is a small dip in brightness, and ML models learn to identify these dips.

Across three papers (KDnuggets ML tutorial, IJRPR study, and the arXiv exoplanet ML paper), the same general pattern appears:

- Clean and preprocess the data
- Engineer useful features
- Handle class imbalance
- Train ML models
- Evaluate with proper metrics

These steps create a reliable pipeline for finding exoplanet signals and estimating habitability.

---

2. Handling Missing Values, Outliers, and Data Issues

All three works follow similar cleaning steps:

- **Missing values** are filled using mean, median, or zeros.
- **Outliers** in flux curves are removed or clipped, especially large spikes unrelated to transits.
- **Descriptive statistics** (mean, min, max, standard deviation) help check distribution shape and identify abnormal entries.
- **Inconsistent entries** are fixed through normalization and scaling.

This ensures the dataset becomes stable before training.

---

3. Feature Engineering

To improve model performance, several feature-engineering techniques are used:

- **PCA (Principal Component Analysis):** Reduces thousands of time-series points into a smaller set of important components.
- **t-SNE:** Used for 2D visualization to understand how planet vs non-planet samples cluster.
- **FFT (Fast Fourier Transform):** Extracts frequency patterns from light curves to help detect periodic transit signals.
- **TSFresh-style features:** Statistical summaries such as peaks, slopes, and entropy (used in the arXiv paper).
- **One-Hot Encoding:** Converts categorical values like star type into numeric form.

These engineered features make ML models learn patterns more effectively.

---

4. Approaches to Class Imbalance

All three papers highlight that exoplanet datasets are heavily imbalanced (very few planets, many non-planet signals). They tackle this using:

- **SMOTE:** Creates synthetic planet-like samples to balance the dataset.
- **Class Weights:** Gives more importance to rare classes during model training.
- **Threshold Tuning:** Adjusts decision score cutoffs to increase recall on rare planet cases.
- **Better metrics:** Precision, recall, F1, and ROC-AUC are used instead of accuracy, which is misleading in imbalanced data.

These methods prevent the model from ignoring rare exoplanet signals.

---

5. Classifiers Used

Common models include:

- **SVM (Support Vector Machine)** with RBF kernel
- **LightGBM / XGBoost** for tree-based learning
- **Neural networks or CNNs** for raw sequence modelling
- **Logistic Regression** on PCA-transformed features

SVM with balanced class weights and FFT/PCA features often performs strongly.

---

6. Habitability Score Index (HSI)

HSI is a numerical score representing how habitable a planet might be.
It typically uses:

- Planet radius
- Temperature
- Orbital period

- Semi-major axis
- Stellar flux
- Equilibrium temperature

HSI combines these into a single rating that helps classify planets as potentially habitable.

---

7. Stellar Compatibility Index (SCI)

SCI measures how suitable the host star is for supporting life.
It focuses on star properties like:

- Temperature
- Luminosity
- Metallicity
- Radius and mass
- Age
- Activity level (flares, radiation)

SCI helps the model understand whether the star creates a safe and stable environment for life to exist on its planets.

In short:

HSI measures the planet. SCI measures the star.
Together they give a complete habitability picture.

---

8. Final Combined Pipeline (Simple Summary)

1. Load dataset (light curves + planetary + stellar data).

2. Fix missing values and remove abnormal outliers.
3. Use descriptive statistics to understand distributions.
4. Encode categorical features (star type → one-hot).
5. Apply PCA, FFT, or TSFresh for feature extraction.
6. Visualize data with t-SNE if needed.
7. Handle imbalance using SMOTE or class weights.
8. Add Habitability Score Index (planet features).
9. Add Stellar Compatibility Index (star features).
10. Train ML models (SVM, LightGBM, etc.).
11. Evaluate using precision, recall, F1, ROC-AUC.
12. Predict exoplanets and classify habitability.

This creates a complete, practical ML workflow used by researchers for exoplanet discovery and life-potential analysis.

Week 2- Day 5

Phase 1: Project Setup & Dependencies
setup_environment.py
Create installation script for all required libraries
Dependencies: pandas, numpy, scikit-learn, imbalanced-learn, xgboost, matplotlib, seaborn, shap
Include requirements.txt for reproducibility
config.py
Centralized configuration file
Define paths, random seeds, performance metrics
Store metadata about features

Phase 2: Data Quality Assessment Module
01_data_quality_assessment.py
Load dataset and display basic information
Calculate missing value percentages for each feature
Analyze missingness patterns (MCAR/MAR/MNAR concepts explained)
Generate boxplots for outlier detection
Visualize distribution of every feature (histograms, KDE plots)
Produce descriptive statistics table
Learning: Understand data quality issues before cleaning

Output
HTML report with all visualizations
CSV with missing value summary

Phase 3: Data Cleaning Module
02_data_cleaning.py
Impute missing values:
Numerical → median (explained: robust to outliers)
Categorical → mode
Handle outliers using IQR method:
Calculate Q1, Q3, IQR for each feature
Replace outliers with median
Document all transformations applied
Learning: Why median over mean? Why IQR method?

Output
Cleaned dataset saved as CSV
Cleaning report documenting all changes

Phase 4: Encoding & Scaling Module
03_encoding_scaling.py
One-Hot Encoding Explained: Convert categorical variables to binary columns
Scaling Explained: Why standardization? What happens without it?
Create preprocessing pipeline using sklearn Pipeline
Demonstrate data leakage prevention
Learning: Production pipeline best practices

Output
Processed dataset
Saved preprocessing pipeline (pickle)

Week 3- Day 1

Phase 5: Exploratory Data Analysis (EDA)
04_eda.py
Class imbalance visualization (bar chart, pie chart)
Correlation heatmap with annotations
Scatter plots, KDEs, boxplots grouped by target class
Identify redundant features (high correlation)
Statistical tests for feature-target relationships
Learning: Understanding data patterns and relationships

Output
Comprehensive EDA report with all visualizations
Feature correlation matrix

Phase 6: Dimensionality Reduction Visualization
05_dimensionality_reduction.py
Apply PCA to reduce to 2D
Apply t-SNE for non-linear dimensionality reduction
Create scatter plots color-coded by habitability (0=red, 1=green)
Purpose: Visualize class separability and clustering patterns
Learning: PCA vs t-SNE, when to use each

Output
PCA and t-SNE scatter plots
Explained variance report for PCA

Week 3 – Checkpoint

- Performed data quality assessment to identify missing values, inconsistencies, and unreliable features.
- Fixing the data types, handling missing values, removing records which aren't used.
- Selected habitability such as radius, mass, temperature, orbital, parameters and steller properties.
- Feature scaling and Normalisation.
- Conducted Explonatory Data Analysis (EDA).
- Applied Dimensionality Reduction Techniques(PCA) and built baseline ML Classification models.
- Metrics used such as Accuracy, Precision, Recall, F1-Score, and Confusion matrix.
- Implemented SMOTE oversampling, step by step ML pipeline.

Milestone 2 – Completed Work

1. Feature Engineering

Applied data standardization to scale numerical features.

Performed normalization to bring features into a common range.

Used PCA (Principal Component Analysis) to reduce dimensionality and remove correlated features.

Used t-SNE for visualization and understanding data separation between classes.

Applied SMOTE to handle class imbalance and improve minority class prediction.

2. Models Implemented

Trained Support Vector Machine (SVM) model for classification.

Trained Random Forest classifier.

Compared models using the same preprocessed dataset.

Observed model behavior on imbalanced vs SMOTE-balanced data.

3. Best Model Evaluation

Selected the best-performing model based on minority class performance.

Evaluated models using confusion matrix.

Reported F1-score to balance precision and recall.

Reported precision to measure correctness of positive predictions.

Reported recall to measure how well the model detects minority (habitable) class.

Analyzed class-wise results instead of relying only on accuracy.

Week 4 - Day 5

1. What is a REST API?

A REST API (Representational State Transfer Application Programming Interface) is a way for two systems to communicate over the internet using HTTP.

- It works on:
- Client–Server model
- Uses URLs to identify resources
- Uses HTTP methods:
- GET → fetch data
- POST → create data
- PUT / PATCH → update data
- DELETE → remove data
- Data is usually exchanged in JSON format

It is stateless: every request is independent

2. What is synchronous vs asynchronous request?

Synchronous request

One task runs at a time. The program waits until the request is completed. Blocking in nature
Example:
API waits for model prediction → only then moves to next request

Asynchronous request

Multiple tasks can run without waiting. Non-blocking. Better for I/O tasks like API calls, DB queries
Example:
API handles new request while model inference is running

3. Difference between multithreading and concurrency?
   Concurrency
   - Multiple tasks in progress
   - Tasks share time on CPU
   - Not necessarily running at the same time
   - Multithreading
   - Multiple threads within a process
   - Threads can run in parallel if multiple CPU cores exist

   How many can run in parallel?
   Parallel threads ≈ number of CPU cores
   Example:
   16-core CPU → ~16 threads truly parallel

   Can we have infinite concurrent tasks?
   Yes, logically we can have many concurrent tasks
   But actual execution is limited by CPU cores and system resources

4. If Flask exists, why do we need FastAPI?
   Flask
   - Simple and flexible
   - Mostly synchronous
   - Async support is limited

   FastAPI
   - Built-in async/await support
   - Very fast (uses ASGI)
   - Automatic API documentation (Swagger)

   Better for:
   High-throughput ML inference APIs
   Async DB calls
   Production ML systems

   Advantage in async:
   FastAPI handles thousands of concurrent requests efficiently.

5. What is throughput?
   Throughput = number of requests processed per unit time
   Example:
   500 requests per second
   Higher throughput = better system performance
   Used to measure:
   - API performance

- Model serving capacity

6. What is inference speed?
   Inference speed = time taken by a trained ML model to produce output for one input
   Measured in:
   - milliseconds (ms)
   - latency per request

   Important for:
   - Real-time ML systems
   - Recommendation systems
   - Chatbots

7. What is SHA?
   SHA (Secure Hash Algorithm) is a cryptographic hashing method.
   Used for:
   - Password hashing
   - File integrity check
   - Git commit hashes

   Common types:
   - SHA-256
   - SHA-1 (used in Git, but not secure now)

8. Linux SSH connection (important for ML deployment)
   SSH (Secure Shell) is used to securely connect to a Linux server.
   Why important:
   - Cloud servers (AWS, GCP, Azure) run on Linux
   - ML models are deployed on Linux machines

   Basic flow:
   - Create cloud VM
   - Use SSH key
   - Connect using terminal
   - Clone GitHub repo
   - Run ML model / API

   Command:
   ssh username@server_ip
   Clone repo:
   git clone https://github.com/username/repo.git

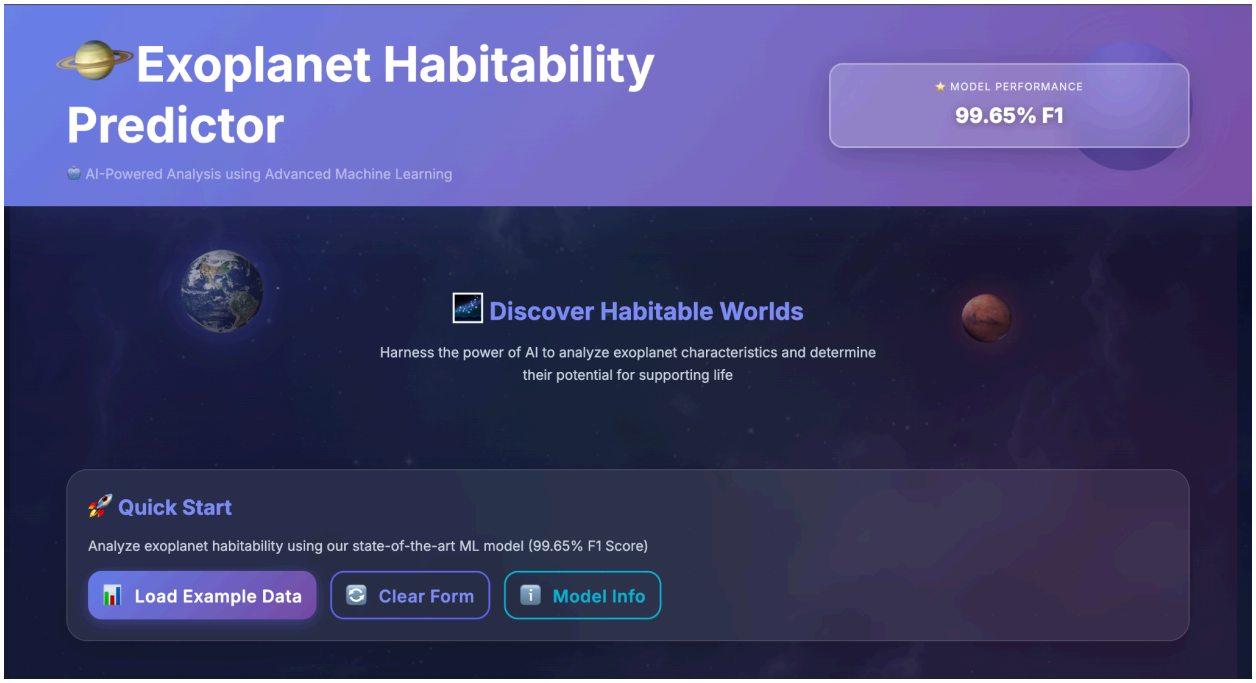
9. What are SHAP values?
   SHAP values explain why a model made a prediction.
   They show:
   - Feature contribution to prediction
   - Positive or negative impact of each feature

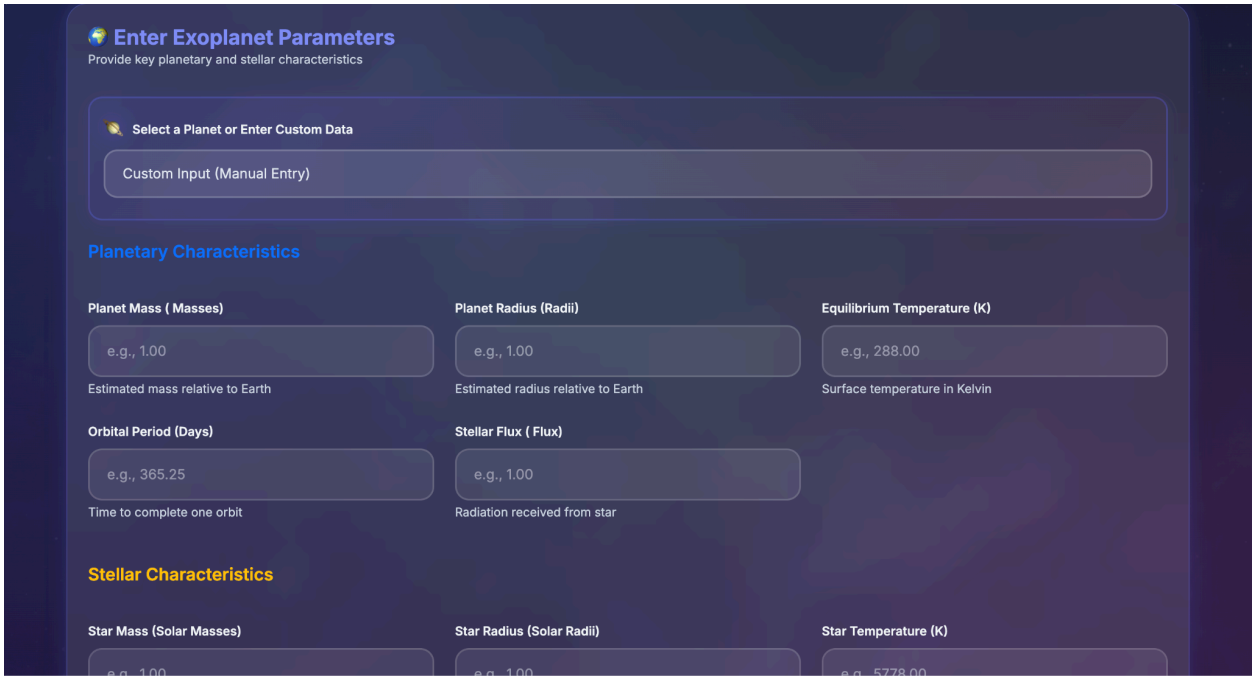
Why important:

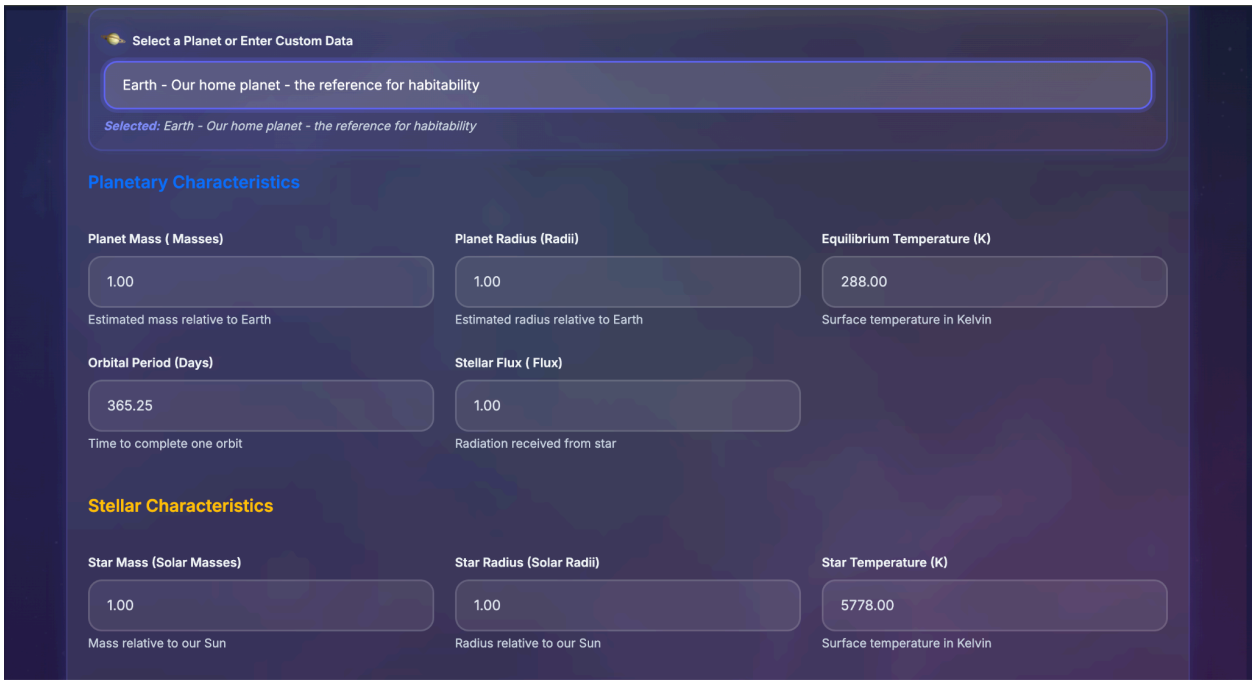
- Model interpretability
- Trust in ML models
- Required in finance, healthcare
- SHAP is considered one of the best interpretability methods.

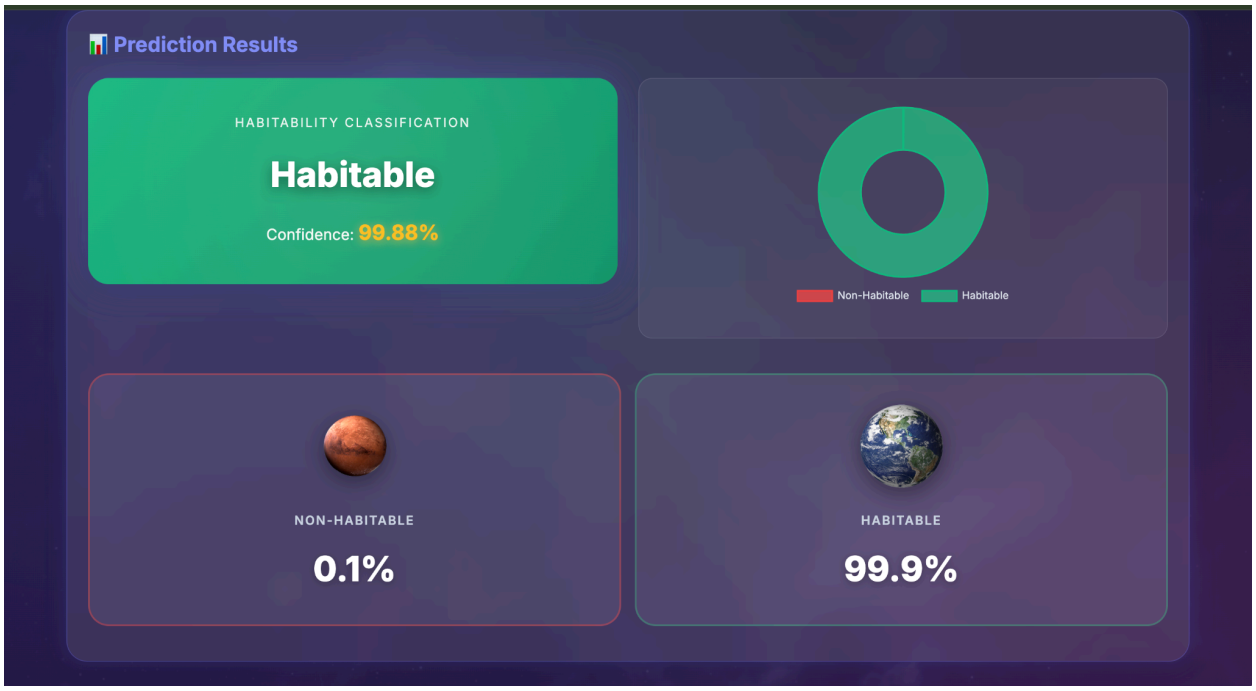
10. DS Algo | Hands-on ML Book (why important?)
    DS & Algorithms: improve problem-solving and coding skills
    Hands-On ML: practical understanding of ML concepts

Week 5 - Website Prototype




Enter Exoplanet Parameters
Provide key planetary and stellar characteristics
Select a Planet or Enter Custom Data
Custom Input (Manual Entry)
Planetary Characteristics
Planet Mass (Masses)
e.g., 1.00
Estimated mass relative to Earth
Planet Radius (Radii)
e.g., 1.00
Estimated radius relative to Earth
Equilibrium Temperature (K)
e.g., 288.00
Surface temperature in Kelvin
Orbital Period (Days)
e.g., 365.25
Time to complete one orbit
Stellar Flux (Flux)
e.g., 1.00
Radiation received from star
Stellar Characteristics
Star Mass (Solar Masses)
Star Radius (Solar Radii)
Star Temperature (K)


Select a Planet or Enter Custom Data
Earth - Our home planet - the reference for habitability
Selected: Earth - Our home planet - the reference for habitability
Planetary Characteristics
Planet Mass (Masses)
1.00
Estimated mass relative to Earth
Planet Radius (Radii)
1.00
Estimated radius relative to Earth
Equilibrium Temperature (K)
288.00
Surface temperature in Kelvin
Orbital Period (Days)
365.25
Time to complete one orbit
Stellar Flux (Flux)
1.00
Radiation received from star
Stellar Characteristics
Star Mass (Solar Masses)
1.00
Mass relative to our Sun
Star Radius (Solar Radii)
1.00
Radius relative to our Sun
Star Temperature (K)
5778.00
Surface temperature in Kelvin



Deployment Link : https://habitability-of-exoplanets-k49h.onrender.com

Video Demo Link:
https://drive.google.com/file/d/1LkS9UV28aSxNJprswAbuXdLn6jRwYDsJ/view?usp=drive_link