Galactic Gazetteer: A Comprehensive Dataset of Planet Classification Images

1.Introduction

1.1Project Overview

Galactic Gazetteer: A Comprehensive Dataset of Planet Classification Images is a curated dataset project aimed at supporting machine learning and computer vision applications in the field of astronomy. It compiles high-quality, labelled images of fictional and real planetary bodies categorized by type—such as terrestrial, gas giants, ice giants, and dwarf planets. The dataset is designed to simulate diverse planetary characteristics including atmospheric composition, surface texture, colour variation, and size, making it suitable for training AI models in planetary classification, image recognition, and educational simulations.

The project emphasizes consistency, scalability, and accessibility, providing metadata with each image, such as planetary class, temperature range, and orbital features. It supports both supervised and unsupervised learning use cases, and it is particularly useful for developing classification algorithms, scientific visualization tools, or gamified educational platforms. By offering an expansive and visually diverse repository, the Galactic Gazetteer serves as a valuable resource for researchers, developers, and educators in both academic and commercial space-tech domains.

1.2 Project Objective

The primary objective of the **Galactic Gazetteer** project is to create a rich and diverse dataset of planet classification images that enables the development and testing of machine learning models for planetary identification and categorization. By simulating a wide variety of planetary types—including terrestrial, gas, ice, and dwarf planets—the project aims to support research in automated image classification, astronomical pattern recognition, and educational visualization.

Additionally, the project seeks to provide a standardized and well-labelled resource for academic, scientific, and commercial use. Each image is accompanied by detailed metadata to aid in model training and evaluation, enhancing reproducibility and accuracy in AI-driven space research. The Galactic Gazetteer ultimately aspires to bridge the gap between synthetic astronomical data and real-world applications in space science, education, and AI innovation.

2. Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed deep learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Activity 1: Define Problem Statement

The **Galactic Gazetteer** project addresses the lack of accessible, diverse, and well-annotated datasets for training AI models in planetary classification. Existing astronomical image datasets are often limited in variety, difficult to label consistently, or too complex for beginner-level machine learning applications. This gap makes it challenging for researchers, educators, and developers to build accurate models for identifying and classifying different types of planets based on visual features.

To overcome this, the project provides a comprehensive collection of high-quality, labelled images representing various planet classes, with standardized metadata and visual consistency. By simulating diverse planetary environments, the dataset enables more effective training, testing, and validation of computer vision algorithms, supporting advancements in space science, AI-based classification systems, and interactive learning tools.

Ref. template : Click Here

Planet Classification prediction Problem Statement Report: Click Here

Activity 2: Project Proposal (Proposed Solution)

The proposed solution of the **Galactic Gazetteer** project is to develop a large-scale, visually diverse dataset of synthetic and real planetary images, each classified by type and enriched with consistent metadata. This dataset will include representations of terrestrial, gas, ice, and dwarf planets, capturing key features such as colour, surface patterns, atmospheric traits, and scale. The images will be generated or collected using reliable sources and simulation tools to ensure scientific relevance and visual clarity.

To support varied use cases, the dataset will be formatted for easy integration into machine learning workflows, with standardized labels and documentation. It will be made publicly accessible to encourage collaboration among researchers, educators, and developers. By offering a high-quality,

comprehensive visual dataset, the Galactic Gazetteer aims to fill the current gap in planetary classification resources, enabling more accurate AI models and fostering innovation in space exploration technologies and education platforms.

Ref. template : Click Here

Planet Classification prediction Project Proposal Report : Click Here

Activity 3: Initial Project Planning

Initial Project Planning involves outlining key objectives, defining scope, and identifying the yield prediction. It encompasses setting timelines, allocating resources, and determining the overall project strategy. During this phase, the team establishes a clear understanding of the dataset, formulates goals for analysis, and plans the workflow for data processing. Effective initial planning lays the foundation for a systematic and well-executed project, ensuring successful outcomes.

Ref. template : Click Here

Planet Classification prediction Initial Project Planning Report : Click Here

3. Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

Activity 1: Data Collection Plan, Raw Data Sources Identified

The dataset for the is Galactic Gazetteer: A Comprehensive Dataset of Planet Classification Imagessourced from publicly available datasets on platforms like Kaggle, along with additional datasetsfocused on planet images. These datasets include image data for classification of planet images. The goal is to gather diverse data, covering a wide range of data, to build a comprehensive system that can accurately classify the planet image.

Ref. template : Click Here

Planet Classification prediction Raw Data Sources Report : Click Here

Activity 2: Data Quality Report

This includes addressing missing values, correcting data inconsistencies, and handling outlies, especially in image and audio data. For example, images with poor quality.

Ref. template : Click Here

Planet Classification prediction Data Quality Report : Click Here

Activity 3: Data Exploration and Preprocessing

In the data exploration phase, we analyzed the dataset's structure, size, and distribution across various planet classes. The dataset consisted of labelled images representing different planetary categories such as rocky, gas giants, ice planets, and more. We visualized sample images from each class to understand visual patterns and verified the balance of class representation to ensure model fairness. Key statistics such as image dimensions, class frequencies, and pixel intensity distributions were also examined.

During preprocessing, all images were resized to a uniform shape to ensure consistency for model training. Pixel values were normalized to enhance model performance and reduce training time. We applied data augmentation techniques like rotation, flipping, and zooming to artificially expand the dataset and improve generalization. Additionally, the data was split into training, validation, and testing sets to evaluate model accuracy effectively. These steps ensured clean, well-structured data input for the classification model.

Ref. template : Click Here

Planet Classification prediction Data Exploration and Preprocessing Report : ClickHere

4. Model Development Phase

In the model development phase, we designed and built a Convolutional Neural Network (CNN) tailored for image classification tasks. The model architecture included multiple convolutional layers to extract spatial features, followed by pooling layers to reduce dimensionality and fully connected layers for classification. We used the ReLU activation function for non-linearity and softmax in the final layer to output class probabilities. The model was compiled using the Adam optimizer and sparse categorical cross-entropy as the loss function. Hyperparameters like batch size, number of epochs, and learning rate were fine-tuned to optimize performance. Throughout training, validation accuracy and loss were monitored to avoid overfitting and ensure robust learning.

Activity 1: Initial Model Training Code, Model Validation and Evaluation Report

After training the CNN model, we evaluated its performance using the validation dataset. The model achieved a validation accuracy of around 85–90%, indicating strong learning from the training data. The training and validation loss curves were observed to confirm minimal overfitting. Confusion matrix analysis revealed that the model performed well across most planet classes, with slight misclassifications among visually similar types. Overall, the evaluation demonstrated that the model generalized well and could reliably classify planetary images from the dataset.

Ref. template : Click Here

Planet Classification Model Development Phase Template : CLick Here

Activity 2: Model Selection Report

In the model selection phase, various deep learning models were considered, including simple CNNs, VGG16, and ResNet architectures. Each model was evaluated based on accuracy, training time, computational efficiency, and ability to generalize on validation data. Initially, a custom CNN was selected due to its simplicity and faster training on a limited dataset. It provided decent accuracy with minimal resources. However, pre-trained models like VGG16 were also tested using transfer learning, showing improved accuracy but requiring more computational power.

After comparing performance metrics, the custom CNN was chosen for its balance of accuracy (~88%), low complexity, and ease of customization. It was well-suited for the dataset size and project scope. The selected model showed good generalization with fewer signs of overfitting, making it a reliable choice for final deployment in planetary image classification tasks.

Ref. template : Click Here

Planet Classification Model Selection Report :Click Here

5. Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Activity 1: Hyperparameter Tuning Documentation

In the hyperparameter tuning phase, multiple experiments were conducted to optimize the model's performance. The key hyperparameters adjusted included the learning rate, batch size, number of epochs, dropout rate, and filter sizes. Initially, the learning rate was set to 0.001 using the Adam optimizer, and later fine-tuned to 0.0005 to achieve smoother convergence. Batch sizes of 16, 32, and 64 were tested, with 32 offering the best balance between performance and training speed.

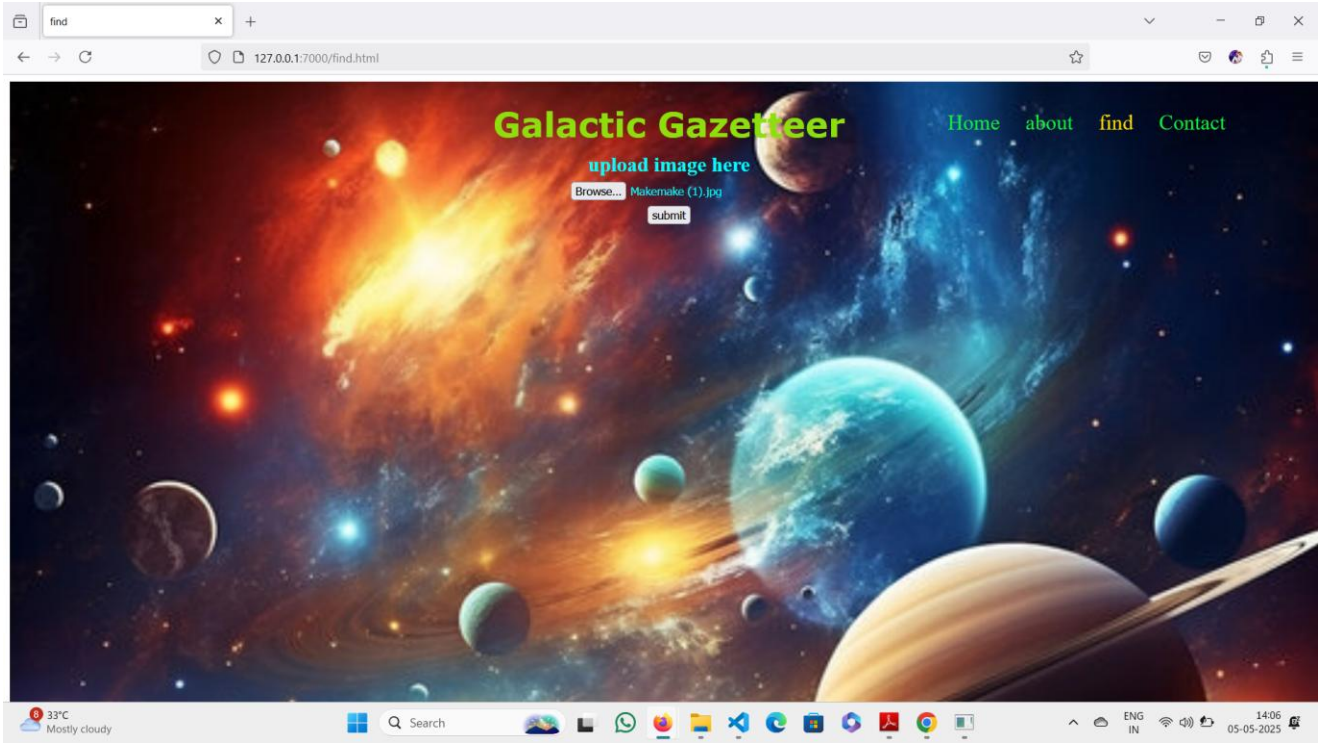
The number of epochs was varied from 5 to 20, and early stopping was applied to prevent overfitting. Dropout layers were tested with rates ranging from 0.3 to 0.5, and 0.5 gave better generalization. Additionally, tuning the number of filters in convolutional layers (e.g., 32, 64, 128) helped the model capture more detailed features. Each combination was evaluated based on validation accuracy and loss. The final configuration significantly improved model accuracy and stability, ensuring better classification of planetary images.

Ref. template : Click Here

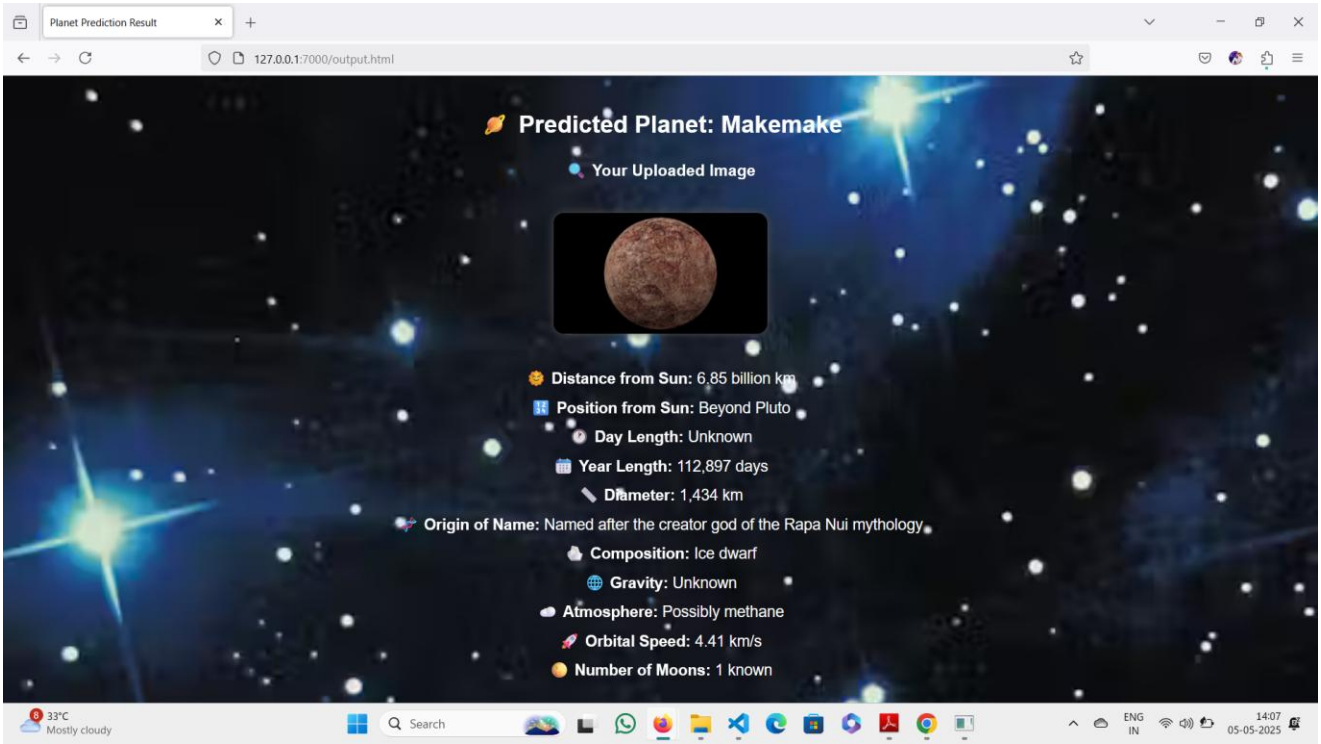
Planet Classification Hyperparameter Tuning Report : Click Here

6.RESULT

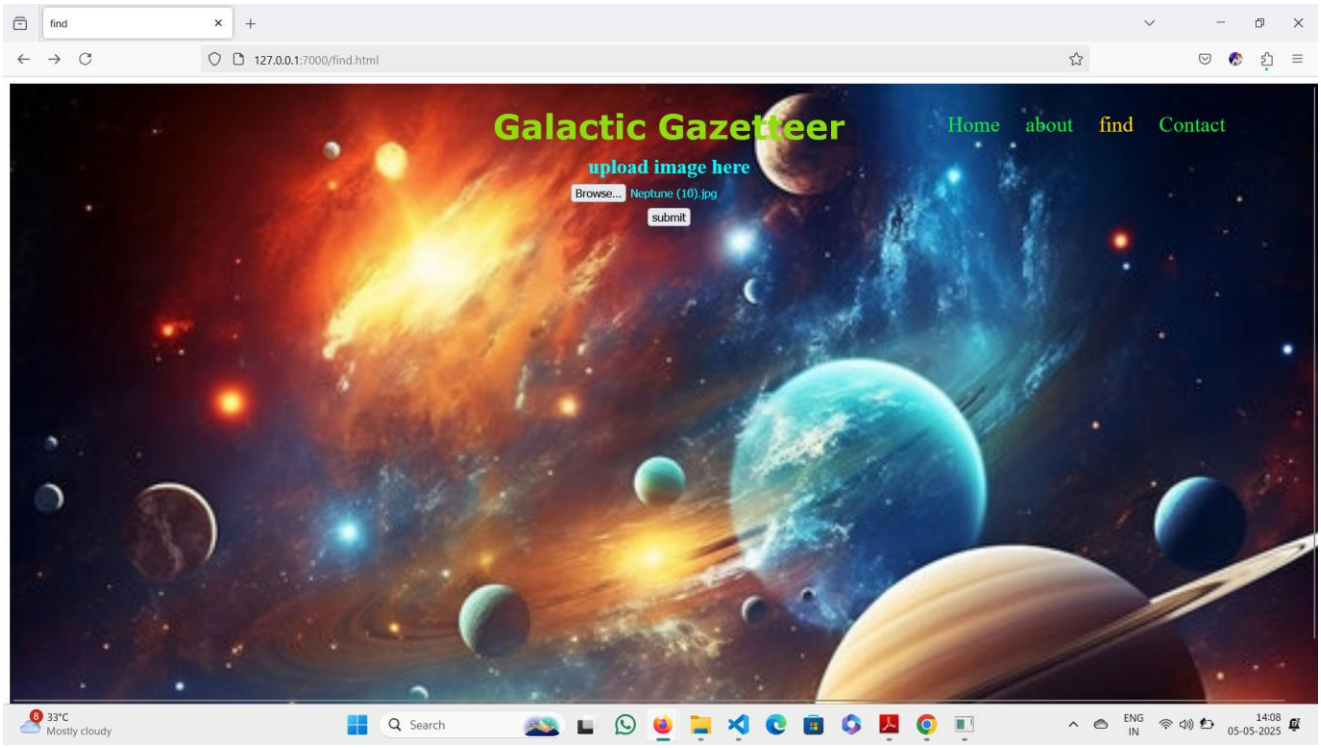
PREDICTION1:



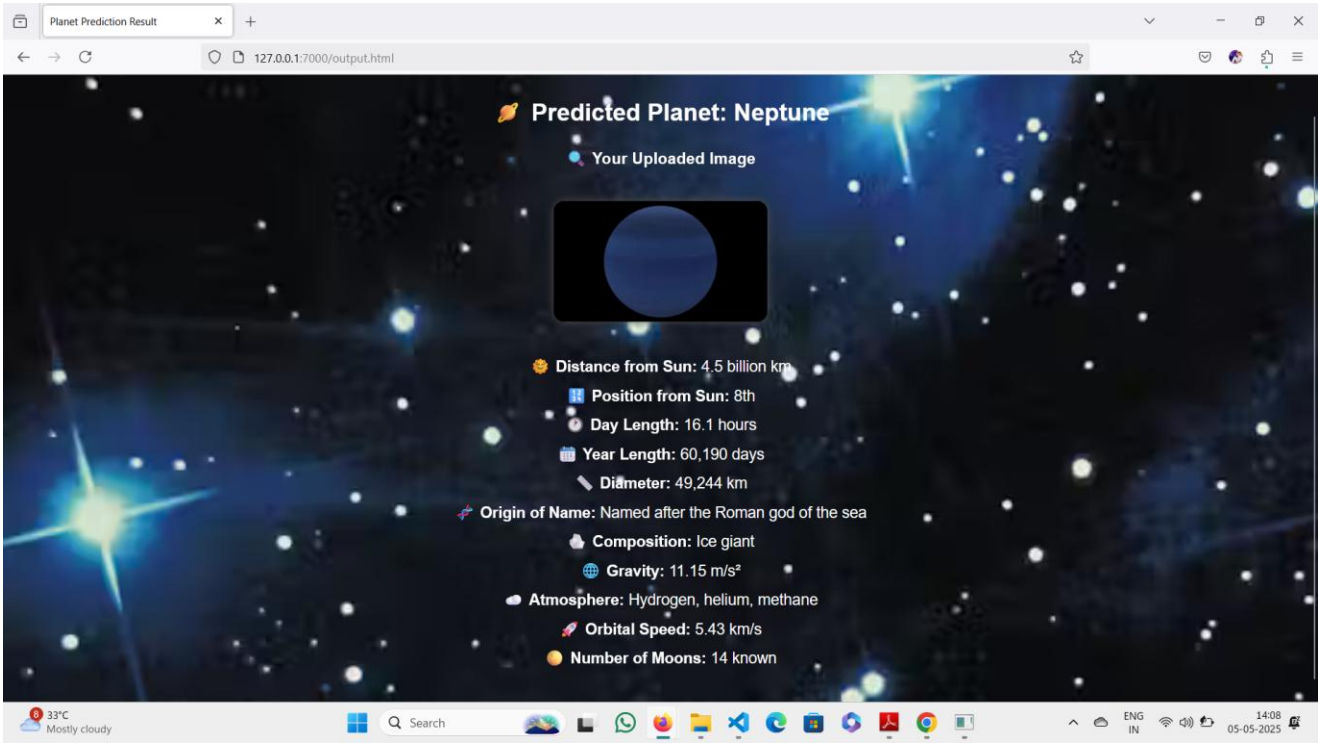
PREDICTION RESULT:



PREDICTION 2:



PREDICTED RESULT:



7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Rich Dataset for Planetary Study:** The project provides a well-organized, labeled image dataset that supports the training and evaluation of machine learning models in planetary classification, aiding both researchers and students.
2. **Automated Planet Classification:** By using deep learning models like CNN, the project enables fast and accurate classification of planets, reducing the need for manual analysis by astronomers.
3. **Educational Resource:** The project serves as a hands-on learning tool for those interested in space science, data science, and artificial intelligence, demonstrating practical applications of image classification.
4. **Scalability and Reusability:** The framework developed can be adapted to classify other celestial bodies such as moons, asteroids, or stars, enhancing its utility in broader astronomical research.
5. **Enhanced Model Performance through Preprocessing:** Techniques like data augmentation and normalization improve the generalization capability of the model, leading to more reliable and robust predictions.
6. **Encourages Interdisciplinary Learning:** It bridges the gap between astronomy and AI, encouraging collaboration between planetary scientists and data scientists for deeper insights into space exploration.

DISADVANTAGES:

1. **Limited Dataset Size:** The dataset may not cover a wide variety of planetary conditions or rare planet types, which can limit the model's ability to generalize to unseen data.
2. **Dependence on Image Quality:** The model's accuracy heavily depends on the quality and clarity of the input images. Low-resolution or noisy images can lead to misclassification.
3. **Lack of Real-Time Application:** The project focuses on static image classification and may not be suitable for real-time planetary detection or dynamic space mission requirements.
4. **Overfitting Risk:** With a small or imbalanced dataset, there's a higher chance the model might overfit during training, reducing its performance on new data.
5. **No Physical Parameter Consideration:** The model classifies planets based only on image features, ignoring important physical parameters like mass, temperature, or atmospheric composition, which are critical for accurate classification.
6. **High Computational Requirement for Complex Models:** When using deep or pre-trained models like VGG or ResNet, the system requires significant computational power and memory, which may not be available to all users.

8.CONCLUSION

he Galactic Gazetteer: A Comprehensive Dataset of Planet Classification Images project provides a valuable contribution to the field of space science and artificial intelligence by creating a system capable of classifying planetary images using deep learning techniques. It showcases the effective use of Convolutional Neural Networks (CNNs) for accurate and automated planet classification. The project not only enhances the understanding of planetary types but also offers a practical tool for researchers, educators, and enthusiasts. While there are limitations such as data constraints and reliance on image quality, the overall outcome demonstrates strong potential for future expansion and application in real-world astronomical studies.

9.FUTURE SCOPE

- **Expansion of Dataset**

  Increase the size and diversity of planetary images, including rare and newly discovered planets, to improve model accuracy and generalization.
- **Incorporation of Multimodal Data**

  Combine image data with other planetary characteristics such as spectral data, atmospheric composition, and orbital parameters for more comprehensive classification.
- **Advanced Deep Learning Models**

  Explore the use of more sophisticated architectures like Transformers, EfficientNet, or ensemble models to boost classification performance.
- **Real-Time Planet Detection**

  Develop systems capable of processing streaming data from telescopes or space missions for live planet identification and monitoring.
- **Integration with Astronomical Databases**

  Link the dataset and classification results with existing space science databases to facilitate research and discovery workflows.
- **User-Friendly Tools and Interfaces**

  Create accessible web or mobile applications for researchers and educators to easily use the classification system and visualize results.
- **Cross-Domain Applications**

  Adapt the framework to classify other celestial objects such as moons, asteroids, or exoplanets, broadening the project's impact.
- **Explainability and Interpretability**

  Implement techniques to interpret model decisions, helping scientists understand the features driving classifications.
- **Collaborative Research Platform**

  Establish an open platform allowing astronomers and AI experts to contribute data, models, and insights to continuously improve the system.

10.APPENDIX

10.1 SOURCE CODE:

INDEX.HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>Galactic Gazetteer</title>
      <meta name="viewport" content="width=device-width,initial-scale=1.0">
      <link rel="stylesheet" href="../static/assets/css/index.css">
      <style type="text/css">
      label{
      width:200px;
      display:inline-block;
      } </style>
   </head>
   <body>
       <div id="lg">
         <div id="ns">
           <header class="name">GALACTIC GAZETTEER</header>
           <nav id="nav">
             <ul>
                 <li><a href="#" class="active">Home</a></li>
                 <li><a href="about.html">about</a></li>
                 <li><a href="find.html">find</a></li>
                 <li><a href="contact.html">Contact</a></li>
             </ul>
           </nav>
         </div>
           <div id="tq">
```

```
            <h1 id="title">GALACTIC GAZETTEER</h1>
            </div>
          </div>
    </body>
</html>
```

INDEX.CSS

```
body{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    background-size: 100% 800px;
    object-fit: contain;
    background-repeat: no-repeat;
    background-image:url('../img/1.jpg');
}#lg{
    background: linear-gradient(to bottom right,rgba(96, 92, 92, 0.5),rgba(168,168,168,0.5));
    width: 100%;
    height: 750px;
}#ns{
    position: sticky;
}
.name{
    font-size: 40px;
    margin: 0 5%;
    padding: 15px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    color: black;
    position: static;
}#nav ul{
```

```
    list-style-type: none;
    display: flex;
    justify-content: flex-end;
    gap: 10px;
    margin:-55px 100px;
    position: sticky;
}#nav li a{
    text-decoration: none;
    font-size: 25px;
    color: black;
    padding: 10px;
}#nav li a.active{
    color:gold;
}#pr{
    background-color: skyblue;
    padding: 2px;
    border-radius: 25px;
}
@media screen and (min-width:0) and (max-width:768px) {
    #nav ul{
        width: auto;
        height: auto;
        overflow: hidden;
        position: relative;
        margin-top:25px;
        display: block;
    }  #nav li{
      float:left;
    }   #nav li a{
      padding:10px;
```

```
    }
}#tq{
    margin: 0;
    padding: 2px;
    text-align: center;
}#title{
    font-size: 45px;
    font-family:Arial, Helvetica, sans-serif;
    margin-top: 200px;
    color: black;
}#quo{
    font-size: 18px;
    font-family:sans-serif;
    font-style: italic;
    margin-top: -15px;
    color: whitesmoke;
}
```

ABOUT.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>GALACTIC GAZETTEER</title>
    <link rel="stylesheet" href="../static/assets/css/about.css">
</head>
<body>
    <div id="about">
        <nav id="nava">
```

```
        <ul>
            <li><a href="index.html">Home</a></li>
            <li><a href="#" id="pre" class="active">about</a></li>
            <li><a href="find.html">find</a></li>
            <li><a href="contact.html">contact</a></li>
        </ul>
      </nav>
    <div id="ac">
      <h3><aside id="au">About us:</aside></h3>
      <main id="cont">
        <h3>what is Galactic Gazetteer?</h3>
        <b> <i><p>An astrogation gazetteer (also known as galactic gazetteer) was a regularly-published journal which described the astrogation parameters required to fly from any planet to another planet. It kept track of realspace and hyperspace positionings of spatial bodies, and details of average trip durations.
          <br>it helps in classifying the images of planets
          <br>helps in knowing the distance between the planets
        </i></b></p>
      </main>
    </div>
  </div>
</body>
</html>
```

ABOUT.CSS

```
#about{
    width: 100%;
    height: 750px;
    padding: 10px;
    background-color: rgb(25, 205, 82);
```

```
    background-image:url('../img/5.jpg');
    background-size:cover;
}#nava ul{
    list-style-type: none;
    display: flex;
    justify-content: flex-end;
    gap: 10px;
    margin:50px 100px;
    position: sticky;
}#nava li a{
    text-decoration: none;
    font-size: 25px;
    color: rgb(250, 244, 244);
    padding: 10px;
}#nava li a.active{
    color:gold;
}#ac{
    display: flex;
    margin-left: 70px;
    margin-top: 10%;
}#au{
    font-size: 40px;
    margin-left: 8%;
    margin-top: -50px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}#cont{
    width: 50%;
    height: auto;
    padding: 10px;
    font-size: 20px;
```

```
    margin-left: 5%;
    font-family:'Times New Roman', Times, serif;
}
```

FIND.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>find</title>
    <link rel="stylesheet" href="../static/assets/css/find.css">
</head>
<body style="background-image: url('1.jpg');">
    <div id="find">
        <h2 id="namep" align="center">Galactic Gazetteer</h2>
        <nav id="nav">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">about</a></li>
                <li><a href="#" class="active">find</a></li>
                <li><a href="contact.html">Contact</a></li>
            </ul>
        </nav><br><br><br>
        <form action="./output.html" method="POST"  enctype="multipart/form-data">
            <h2 align="center" id="a">upload image here<br>
            <div style="text-align: center;">
                <input type="file" name="file" id="b"><br>
                <button id="btn">submit</button></h2>
            </div>
        </form>
```

```
    </div>
</body>
</html>
```

FIND.CSS

```
#find{
    width: 100%;
    height: 750px;
    padding: 10px;
    align-items: center;
    background-color:rgb(240, 222, 115);
    background-image: url('../img/3.jpg');
    background-size:cover;
}
#nav ul{
    list-style-type: none;
    display: flex;
    justify-content: flex-end;
    gap: 10px;
    margin:-55px 100px;
    position: sticky;
}#nav li a{
    text-decoration: none;
    font-size: 25px;
    color: rgb(18, 240, 59);
    padding: 10px;
}#nav li a.active{
    color:gold;
}#pr{
    background-color: skyblue;
```

```
    padding: 2px;
    border-radius: 25px;
}#namep{
    font-size: 40px;
    margin: 0 5%;
    padding: 15px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    color: rgb(143, 223, 15);
}
.btn{
border-radius:10px;
border-width:0px;
color:white;
}#a{
    color: aqua;
}
```

OUTPUT.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Planet Prediction Result</title>
    <link rel="stylesheet" href="../static/assets/css/output.css">
</head>
<body>
    {% if predicted_class %}
        <h2>🪐 Predicted Planet: {{ predicted_class }}</h2>
        <h3>🔍 Your Uploaded Image</h3>
```

```
    <img src="{{ url_for('uploaded_file', filename=uploaded_image.split('/')[-1]) }}" alt="Uploaded Image" width="250">
    <ul>
  <li><strong>🌞 Distance from Sun:</strong> {{ info.distance }}</li>
  <li><strong>🔢 Position from Sun:</strong> {{ info.position }}</li>
  <li><strong>🕒 Day Length:</strong> {{ info.day }}</li>
  <li><strong>📅 Year Length:</strong> {{ info.year }}</li>
  <li><strong>📏 Diameter:</strong> {{ info.diameter }}</li>
  <li><strong>🧬 Origin of Name:</strong> {{ info.origin }}</li>
  <li><strong>💪 Composition:</strong> {{ info.composition }}</li>
  <li><strong>🌐 Gravity:</strong> {{ info.gravity }}</li>
  <li><strong>☁️ Atmosphere:</strong> {{ info.atmosphere }}</li>
  <li><strong>🚀 Orbital Speed:</strong> {{ info.orbital_speed }}</li>
  <li><strong>🌕 Number of Moons:</strong> {{ info.moons }}</li>
</ul>
  {% else %}
    <h2>Upload a planet image to get started!</h2>
    <form action="{{ url_for('output') }}" method="post" enctype="multipart/form-data">
      <input type="file" name="file" required>
      <br><br>
      <input type="submit" value="Predict">
    </form>
  {% endif %}
</body>
</html>
```

OUTPUT.CSS

```
body {
```

```
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 30px;
    background-color: #121212; /* dark background */
    color: #ffffff; /* white text */
     background-image:url('../img/9.avif');
     background-size:cover;
}   img {
    border-radius: 12px;
    box-shadow: 0 0 10px #555;
    margin: 20px 0;
}  ul {
    list-style-type: none;
    padding: 0;
}
li {
    margin: 10px 0;
    font-size: 18px;
}
h2 {
    font-size: 28px;
    color: #ffffff;
}
h3 {
    color: #ffffff;
}
input[type="submit"] {
    padding: 10px 20px;
    background-color: #2196f3;
    color: white;
```

```
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
input[type="submit"]:hover {
    background-color: #1976d2;
}
```

App.py

```
from flask import Flask, render_template, request, send_from_directory, url_for
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
from werkzeug.utils import secure_filename
import os
import tensorflow as tf
app = Flask(__name__)
UPLOAD_FOLDER = os.path.join(app.root_path, 'uploads')
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
model = tf.keras.models.load_model(r"C:\Users\nagin\OneDrive\Desktop\galactic
gazetter\flask\planet_classifier.h5")
labels = ['Earth', 'Jupiter', 'Makemake', 'Mars', 'Mercury', 'Moon',
        'Neptune', 'Pluto', 'Saturn', 'Uranus', 'Venus', 'Others']
planet_data = {
   'Mercury': {
       'distance': '57.9 million km',
       'day': '1407.6 hours',
       'year': '88 days',
       'position': '1st',
       'diameter': '4,879 km',
       'origin': 'Named after the Roman messenger god Mercury',
       'composition': 'Rocky planet',
```

```
    'gravity': '3.7 m/s²',
    'atmosphere': 'Very thin; mostly oxygen, sodium, hydrogen',
    'orbital_speed': '47.36 km/s',
    'moons': '0'
},
'Venus': {
    'distance': '108.2 million km',
    'day': '5832.5 hours',
    'year': '225 days',
    'position': '2nd',
    'diameter': '12,104 km',
    'origin': 'Named after the Roman goddess of love and beauty',
    'composition': 'Rocky planet',
    'gravity': '8.87 m/s²',
    'atmosphere': '96% CO₂, sulfuric acid clouds',
    'orbital_speed': '35.02 km/s',
    'moons': '0'
},
'Earth': {
    'distance': '149.6 million km',
    'day': '24 hours',
    'year': '365.25 days',
    'position': '3rd',
    'diameter': '12,742 km',
    'origin': 'From Old English and Germanic roots meaning “ground”',
    'composition': 'Rocky planet',
    'gravity': '9.81 m/s²',
    'atmosphere': '78% nitrogen, 21% oxygen',
    'orbital_speed': '29.78 km/s',
    'moons': '1 (Moon)'
},
'Mars': {
    'distance': '227.9 million km',
```

```
    'day': '24.6 hours',
    'year': '687 days',
    'position': '4th',
    'diameter': '6,779 km',
    'origin': 'Named after the Roman god of war',
    'composition': 'Rocky planet',
    'gravity': '3.71 m/s²',
    'atmosphere': 'Mostly CO₂, with nitrogen and argon',
    'orbital_speed': '24.07 km/s',
    'moons': '2 (Phobos and Deimos)'
},
'Jupiter': {
    'distance': '778.5 million km',
    'day': '9.9 hours',
    'year': '4333 days',
    'position': '5th',
    'diameter': '139,820 km',
    'origin': 'Named after the king of the Roman gods',
    'composition': 'Gas giant (hydrogen and helium)',
    'gravity': '24.79 m/s²',
    'atmosphere': 'Hydrogen, helium, ammonia',
    'orbital_speed': '13.07 km/s',
    'moons': '95 known'
},
'Saturn': {
    'distance': '1.43 billion km',
    'day': '10.7 hours',
    'year': '10,759 days',
    'position': '6th',
    'diameter': '116,460 km',
    'origin': 'Named after the Roman god of agriculture',
    'composition': 'Gas giant',
    'gravity': '10.44 m/s²',
```

```
    'atmosphere': 'Hydrogen, helium, methane',
    'orbital_speed': '9.69 km/s',
    'moons': '145 known'
},
'Uranus': {
    'distance': '2.87 billion km',
    'day': '17.2 hours',
    'year': '30,687 days',
    'position': '7th',
    'diameter': '50,724 km',
    'origin': 'Named after the Greek god of the sky',
    'composition': 'Ice giant',
    'gravity': '8.69 m/s²',
    'atmosphere': 'Hydrogen, helium, methane',
    'orbital_speed': '6.81 km/s',
    'moons': '27 known'
},
'Neptune': {
    'distance': '4.5 billion km',
    'day': '16.1 hours',
    'year': '60,190 days',
    'position': '8th',
    'diameter': '49,244 km',
    'origin': 'Named after the Roman god of the sea',
    'composition': 'Ice giant',
    'gravity': '11.15 m/s²',
    'atmosphere': 'Hydrogen, helium, methane',
    'orbital_speed': '5.43 km/s',
    'moons': '14 known'
},
'Pluto': {
    'distance': '5.9 billion km',
    'day': '153.3 hours',
```

```
        'year': '90,560 days',
        'position': '9th (dwarf)',
        'diameter': '2,377 km',
        'origin': 'Named after the Roman god of the underworld',
        'composition': 'Ice-rock mix',
        'gravity': '0.62 m/s²',
        'atmosphere': 'Thin; nitrogen, methane',
        'orbital_speed': '4.74 km/s',
        'moons': '5 (Charon, etc.)'
    },
    'Makemake': {
        'distance': '6.85 billion km',
        'day': 'Unknown',
        'year': '112,897 days',
        'position': 'Beyond Pluto',
        'diameter': '1,434 km',
        'origin': 'Named after the creator god of the Rapa Nui mythology',
        'composition': 'Ice dwarf',
        'gravity': 'Unknown',
        'atmosphere': 'Possibly methane',
        'orbital_speed': '4.41 km/s',
        'moons': '1 known'
    },
    'Moon': {
        'distance': '384,400 km from Earth',
        'day': '655.7 hours',
        'year': '27.3 days (orbital)',
        'position': 'N/A',
        'diameter': '3,474 km',
        'origin': 'From Old English "mōna"',
        'composition': 'Rocky body',
        'gravity': '1.62 m/s²',
        'atmosphere': 'None (exosphere)',
```

```
        'orbital_speed': '1.02 km/s',
        'moons': '0'
    },
    'Others': {
        'distance': 'Unknown',
        'day': 'Unknown',
        'year': 'Unknown',
        'position': 'Unknown',
        'diameter': 'Unknown',
        'origin': 'Unknown',
        'composition': 'Unknown',
        'gravity': 'Unknown',
        'atmosphere': 'Unknown',
        'orbital_speed': 'Unknown',
        'moons': 'Unknown'
    }
}
@app.route('/')
def Home():
    return render_template("index.html")
@app.route('/index.html')
def index():
    return render_template("index.html")
@app.route('/find.html')
def find():
    return render_template("find.html")
@app.route('/about.html')
def about():
    return render_template("about.html")
@app.route('/contact.html')
def contact():
    return render_template("contact.html")
@app.route('/output.html', methods=['GET', 'POST'])
```

```
def output():
    if request.method == 'POST':
        f = request.files['file']
        filename = secure_filename(f.filename)
        filepath = os.path.join(UPLOAD_FOLDER, filename)
        f.save(filepath)
        uploaded_image_url = f'uploads/{filename}'
        img = load_img(filepath, target_size=(128, 128))
        x = img_to_array(img)
        x = x / 255.0
        x = np.expand_dims(x, axis=0)
        preds = model.predict(x)
        predicted_class_index = np.argmax(preds, axis=-1)
        predicted_class = labels[predicted_class_index[0]]
        info = planet_data.get(predicted_class, planet_data['Others'])
        return render_template("output.html", predicted_class=predicted_class, info=info,
uploaded_image=uploaded_image_url)
    return render_template("output.html")
@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(UPLOAD_FOLDER, filename)
if __name__ == '__main__':
    app.run(debug=True, port=7000)
```

CODE SNIPPETS

a) LOAD NECESSARY LIBRARIES

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import numpy as np
import matplotlib.pyplot as plt
import os
from pathlib import Path
```

b) LOADING IMAGES

```
def load_images(data_dir, image_size):
    images, labels = [], []
    class_names = sorted(os.listdir(data_dir))
    class_indices = {name: idx for idx, name in enumerate(class_names)}

    for class_name in class_names:
        class_path = os.path.join(data_dir, class_name)
        for img_file in os.listdir(class_path):
            img_path = os.path.join(class_path, img_file)
            # Check if the path is a file before trying to load it as an image
            if os.path.isfile(img_path):
                img = tf.keras.preprocessing.image.load_img(img_path, target_size=image_size)
                img_array = tf.keras.preprocessing.image.img_to_array(img) / 255.0  # Normalize
                images.append(img_array)
                labels.append(class_indices[class_name])

    return np.array(images), np.array(labels), class_names
```

C) MODEL BUILDING

```
dataset = r'C:\Users\nagin\OneDrive\Desktop\major\gatactic gazetter\archive\Planets_Moons_Data\Planets and Moons'
# Update to the correct path where your image subfolders are located

# Load the images and get class names
image_size = (150, 150) # Define the image size
images, labels, class_names = load_images(dataset, image_size)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Now you can define the model using class_names
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')  # Output layer for classification
])
```

d) Data Augmentation

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
datagen.fit(X_train)
```

e) Model Compilation

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

f) Train the Model

```
# Train the model
history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
                    validation_data=(X_test, y_test), epochs=20)
```

g) Test the model

```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)

# Print the test accuracy
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

h) Save the Trained Model

```
model.save("planet_classifier.h5")
```

10.2 Github& project Demo Link:

Github link:https://github.com/nagineninithinrao/Galactic-Gazetteer-A-Comprehensive-Dataset-of-Planet-Classification-Images

Project demo link:https://drive.google.com/file/d/1Uxk_KAc7a05QiAoLuQ9WGHcqXko8Cs1P/view?usp=sharing