

*An internship report submitted in fulfilment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**Computer Science and Engineering**

**By**

**Snehil Jha**

**Roll No: DC2021BTE0015**

***Under Supervision of***

**K.VELAYUDHAM, GM, AS, SCOF & MAFS/RO**

**SDSC-SHAR, SRIHARIKOTA, ANDHRA PRADESH-524121**

(Duration: 4th July 2024 - 2nd August 2024)



**School of Technology**

**Assam Don Bosco University,**

**Airport Road, Azara, Guwahati, Assam-781017**

**ACKNOWLEDGEMENT**

I feel privileged and thankful to mention such esteemed dignitaries who aided me for successful completion of internship training.

My best regards to Shri G. GRAHADURAI, Deputy Director, RO, for giving permission for doing internship at Range Operation.

I express my sincere gratitude to Shri K. VELAYUDHAM, GM, AS/SCOF/RO whose incessant support and belief in me laid stepping stones in accomplishment of my internship program.

I would like to thank Mr. BHARATHI KANNAN R , SCIENTIST/ ENGINEER - SE , APP-SW, RO for sharing his Suggestions.

I express my sincere thanks and deep veneration to Dr. Bobby Sharma, HOD, Department of Computer Science and Engineering for giving me this opportunity for successful completion of internship training.

I express my sincere thanks to our beloved guide Mr. Thoudam Johnson Singh Assistant Professor Stage III Department of CSE School of Technology Assam Don Bosco University, for helping me with this opportunity and successful completion of internship training.

In conclusion, we shall remember our internship training and put an oath of presenting the training experience to prove our ability and work for the pride of the organization in all respects wherever we work.

### BONAFIDE CERTIFICATE

This is to certify that SNEHIL JHA, student of SCHOOL OF TECHNOLOGY, ASSAM DON BOSCO UNIVERSITY, GUWAHATI ASSAM, Reg-no. AZ-0-2021-000272 has completed his internship training successfully at SHAR COMPUTER FACILITIES (SCOF)/RO, SATISH DHAWAN SPACE CENTRE, SRIHARIKOTA, A.P-524124 from 04-07-2024 to 02-08-2024.

During the internship period his conduct was found to be .

K. VELAYUDHAM GM, AS, SCOF, RO

Internship Guide

|  |  |  |
| --- | --- | --- |
| **INDEX** | | |
| **S.No.** | **CONTENT** | **Page No.** |
| 1. | Acknowledgement | 2 |
| 2. | About Organization:   * 1. Introduction   2. Range Operations (RO)   3. SHAR Computer Facility (SCOF) Operations | 5-7 |
| 3. | Technology Used:   * 1. PYTHON   2. PYTORCH   3. DJANGO   4. HTML   5. CSS   6. JAVASCRIPT   7. OPENCV   8. REST-API | 8-11 |
| 4. | Project Title:   * 1. Multi-model ML Architecture for Automated Pulsar Detection | 12-30 |
| 5. | Conclusion | 31 |
| 6. | Reference | 32 |

## About Organization



##### INTRODUCTION

Indian Space Research Organization (ISRO) is an independent unit, under the Department of Space, Government of India. Satish Dhawan Space Centre SHAR, a rocket launch station is one of the best-known names among the Spaceports of the World today located in Sriharikota, India. Space vehicles fly from here giving an assured access to space for indigenous satellites as well as commercial satellites. Diverse kinds of Space missions with remote sensing, communications, navigation and scientific satellites are accomplished.

Sriharikota Island was chosen in 1969 for setting up a rocket launch station. It became operational on October 9, 1971, with a flight of Rohini- a small sounding rocket. Since then, the facilities here were gradually expanded to meet the growing needs of ISRO.

The activities at SDSC SHAR are grouped under Vehicle assembly and static test operations, Range Operations, Liquid storage and service facilities, Solid Propellant, Space Booster Plant and to launch the indigenously designed and developed launch vehicles like SLV, ASLV, PSLV, GSLV Mk-II and GSLV MK-III. Management services including program planning and human resources development. Systems reliability groups support the center. Sriharikota Common Facilities provide administrative and auxiliary support for the center.

##### RANGE OPERATIONS

Range Operations Entity is the focal point for control of launch operations for various missions of ISRO at SDSC SHAR.

This entity is entrusted to perform a diverse range of activities ranging from tracking, tele- commanding, real time systems for mission monitoring, deployment and maintenance of various networks including mission network, campus network, internet network, surveillance network, networking services with secure features, computerization of administrative activities, web and mobile application development, photography including still and video coverage, meteorology services providing a constant weather watch for clearing various launch campaign activities.

It has facilities such as the Mission Control Centre, Computer and Communications Centre, Multi Object Tracking Radar (MOTR) including mobile MOTR, SHAR Computer Facilities operations (SCOF), Network Operations Centre (NOC).



**SCOF**

###### SHAR Computer Facility (SCOF) Operations

SHAR Computer Facility caters to the computing needs of the Centre. It consolidates the hardware, software and networking needs of the Centre. It has isolated the Internet and Intranet as per the security guidelines.

* + 1. SCOF has established a Network Operations Centre as well as Locational Diversity of Servers. The Network Operations Centre also acts as a Data Centre.
    2. A state-of-the-art High-Performance Computing (HPC) servers are commissioned for catering to the Meteorological, MOTR as well as other needs of the Centre.
    3. It has established three real time systems as part of mission network, the Range Safety (RS) real time system, the Specialists’ Display System (SDS) real time system and Mission Control Centre (MCC) real time system.
    4. These systems cater to real time monitoring of the status of various ground stations prior to the launch as well as the filling phase of the liquid and cryogenic stages of the launch vehicle during the countdown phase and from lift off they help in monitoring the performance of vehicle subsystems.

# TECHNOLOGY USED

##### 3.1 PYTHON

Python is the primary programming language employed in this pulsar detection project due to its versatility, ease of use, and powerful ecosystem of libraries suited for data science and machine learning. Python's syntax is clean and readable, making it an ideal choice for rapid development and iterative testing. The language is extensively used for data preprocessing, such as loading .phcx files, extracting features, and preparing data for neural network models. Libraries like NumPy and pandas are crucial for handling numerical data efficiently, allowing for operations on large datasets and complex data transformations. Additionally, Python's support for various machine learning frameworks, including PyTorch and ONNX, enables seamless model development, training, and deployment. Its widespread adoption in the scientific community ensures extensive documentation and community support, facilitating troubleshooting and continuous improvement of the project.

##### 3.2 PYTORCH

PyTorch is the deep learning framework of choice in this project, providing a flexible and efficient platform for developing, training, and deploying neural network models. Known for its dynamic computation graph, PyTorch allows for real-time debugging and adjustment of model architectures, making it easier to experiment with different network configurations. The framework's comprehensive GPU support accelerates the training process, crucial for handling the extensive HTRU1 and HTRU2 datasets. PyTorch is utilized to build both the artificial neural network (ANN) for numerical feature analysis and the convolutional neural network (CNN) for image data classification. After training, the models are exported to the ONNX format to optimize inference performance. PyTorch's extensive library of pre-built modules and its active development community provide a robust foundation for developing state-of-the-art machine learning models tailored to pulsar detection.

##### 3.3 DJANGO

Django is a high-level Python web framework that promotes rapid development and clean, pragmatic design. It serves as the backbone of the web application, providing a robust and scalable environment for deploying the pulsar detection system. Django's powerful ORM (Object-Relational Mapping) simplifies database interactions, allowing for efficient data storage and retrieval. The framework's built-in security features, such as protection against SQL injection and cross-site scripting, ensure that the application is secure and reliable. Django REST framework (DRF) extends Django to facilitate the creation of RESTful APIs, which are essential for the project's backend services. These APIs handle requests for model predictions, data processing, and user interactions, enabling seamless communication between the frontend and the deep learning models. Django's modular architecture and extensive middleware options ensure that the application is maintainable and scalable, capable of handling increased load and future feature expansions.

##### 3.4 HTML

HTML (HyperText Markup Language) is the standard language for creating and structuring content on the web. In this project, HTML is used to develop the frontend of the pulsar detection system, defining the layout and structure of the web pages that users interact with. It ensures that the content is well-organized and accessible, providing a solid foundation for further styling and scripting. HTML forms are crucial for capturing user inputs, such as uploading .phcx files or entering numerical data for ANN predictions. By using semantic HTML elements, the web application achieves better accessibility and search engine optimization, improving the overall user experience. HTML's compatibility with various web technologies and its fundamental role in web development make it indispensable for creating a functional and user-friendly interface.

##### 3.5 CSS

CSS (Cascading Style Sheets) is used to control the presentation and layout of the HTML elements on the web pages. It allows developers to separate content structure from design, ensuring that the web application is visually appealing and easy to maintain. In this project, CSS is employed to style the frontend, enhancing the user experience with visually attractive and responsive designs. CSS frameworks like Bootstrap may be used to speed up the development process by providing pre-designed components and a responsive grid system. Custom CSS rules are also written to ensure that the application meets specific design requirements, such as consistent color schemes, typography, and layout adjustments for different screen sizes. The use of CSS ensures that the web application is not only functional but also aesthetically pleasing and user-friendly.

##### 3.6 JAVASCRIPT

JavaScript is a versatile scripting language that adds interactivity and dynamic behavior to the web application. It plays a crucial role in enhancing the user experience by enabling real-time data validation, asynchronous data fetching, and dynamic content updates without reloading the page. In this project, JavaScript is used to handle user interactions, such as submitting forms and displaying prediction results. It also facilitates communication with the Django backend via AJAX requests, ensuring that data is processed and returned efficiently. JavaScript frameworks and libraries like React or Vue.js can further enhance the frontend by providing a more structured and component-based approach to building user interfaces. By leveraging JavaScript's capabilities, the web application becomes more responsive and interactive, providing a seamless experience for users.

##### 3.7 OPENCV

OpenCV (Open Source Computer Vision Library) is a powerful tool for image processing and computer vision tasks. In the context of this project, OpenCV is used to handle the conversion of .phcx files to images that can be fed into the CNN model for pulsar detection. It provides a comprehensive set of functions for image manipulation, such as resizing, normalization, and format conversion, which are essential for preparing the images for model inference. OpenCV's capabilities extend to more advanced image processing tasks, such as feature extraction and image enhancement, which can further improve the accuracy of the CNN model. The library's extensive documentation and active community support make it a reliable choice for implementing computer vision functionalities, ensuring that the image data is processed efficiently and accurately.

##### 3.8 REST-API

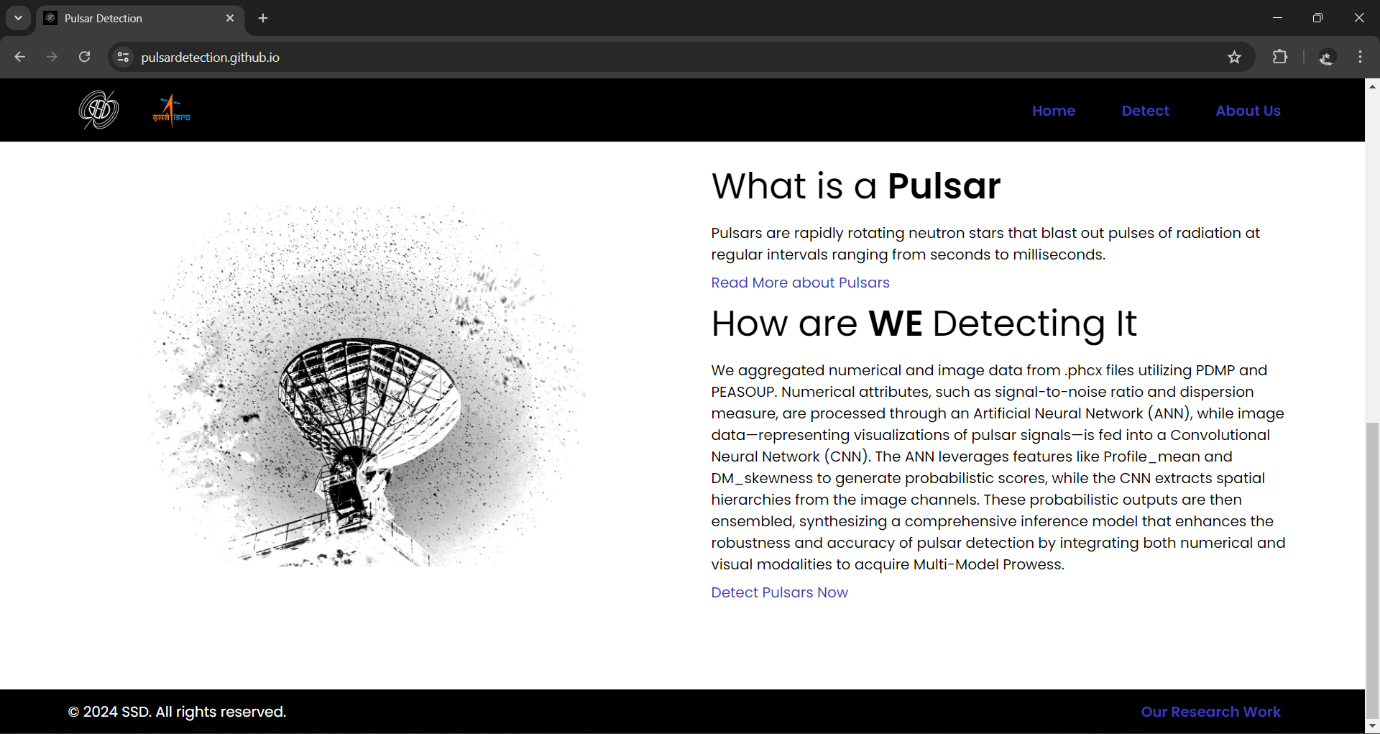
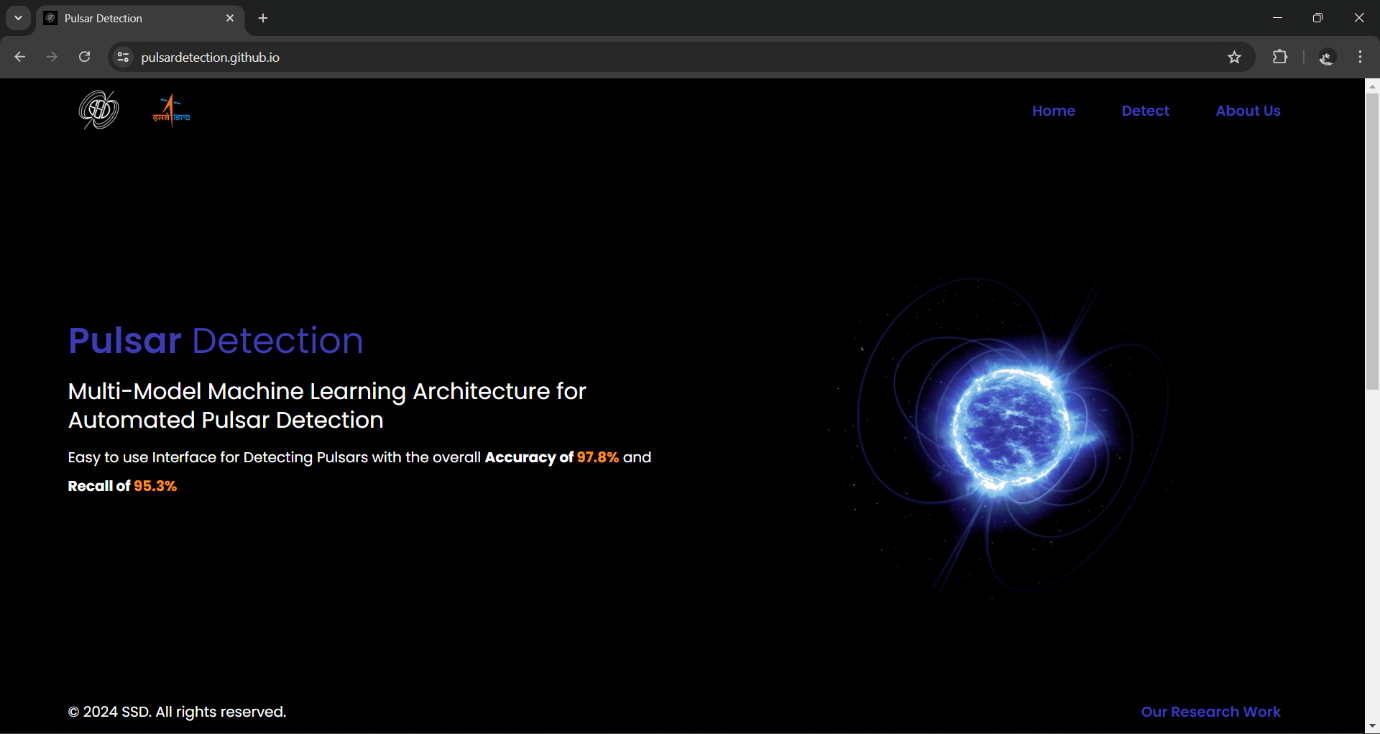
A REST-API (Representational State Transfer Application Programming Interface) is implemented to facilitate communication between the frontend and backend of the pulsar detection system. The REST API allows different components of the system to interact over HTTP, enabling the submission of data, execution of model inference, and retrieval of prediction results. Django REST framework (DRF) is used to create RESTful endpoints that handle requests for ANN predictions, CNN predictions, and combined model outputs. The API architecture ensures that the system is scalable, modular, and easy to integrate with other services. By adhering to REST principles, the API provides a standardized and stateless interface, making it easier to manage and extend the application. This approach ensures that the web application can handle multiple simultaneous requests, providing timely and accurate responses to users.

# PROJECT TITLE

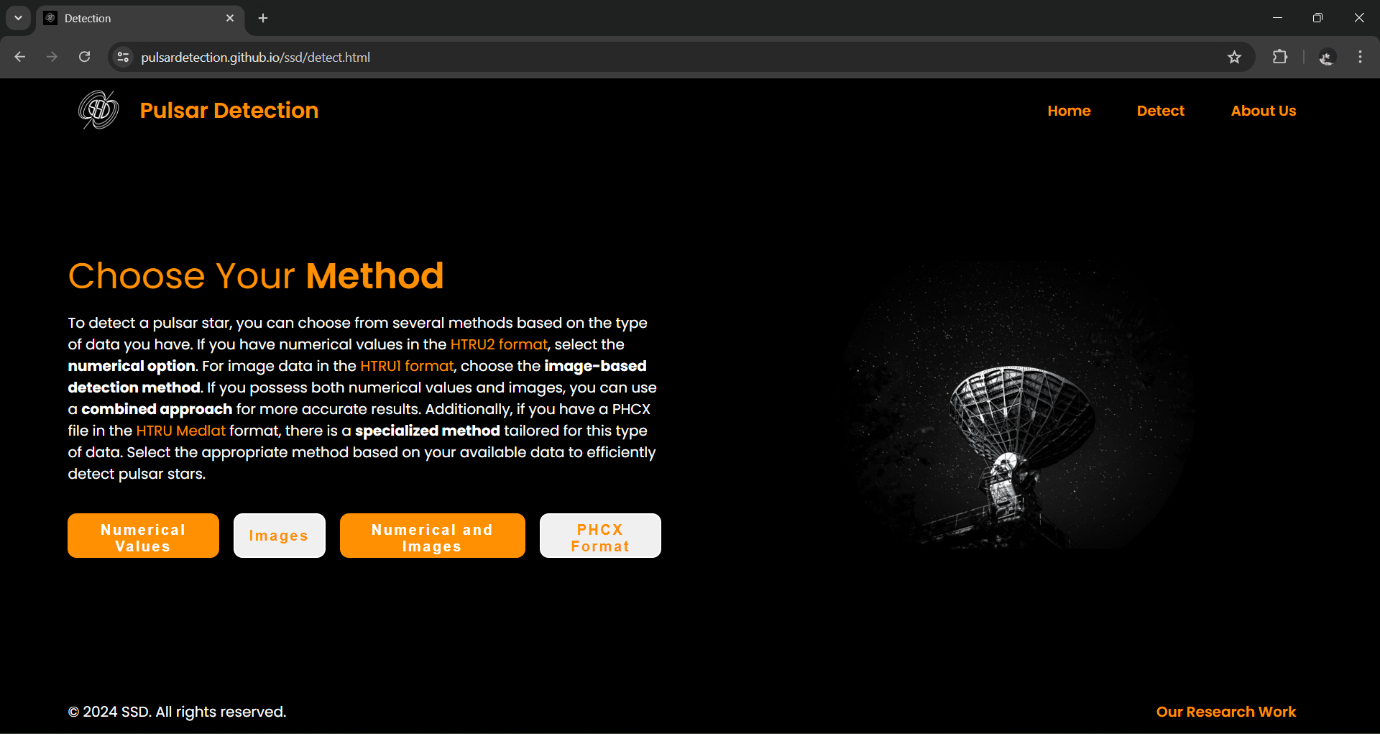
“Multi-model ML Architecture for Automated Pulsar Detection”

##### 4.1 Detailed Backend Functionality of the Pulsar Detection System

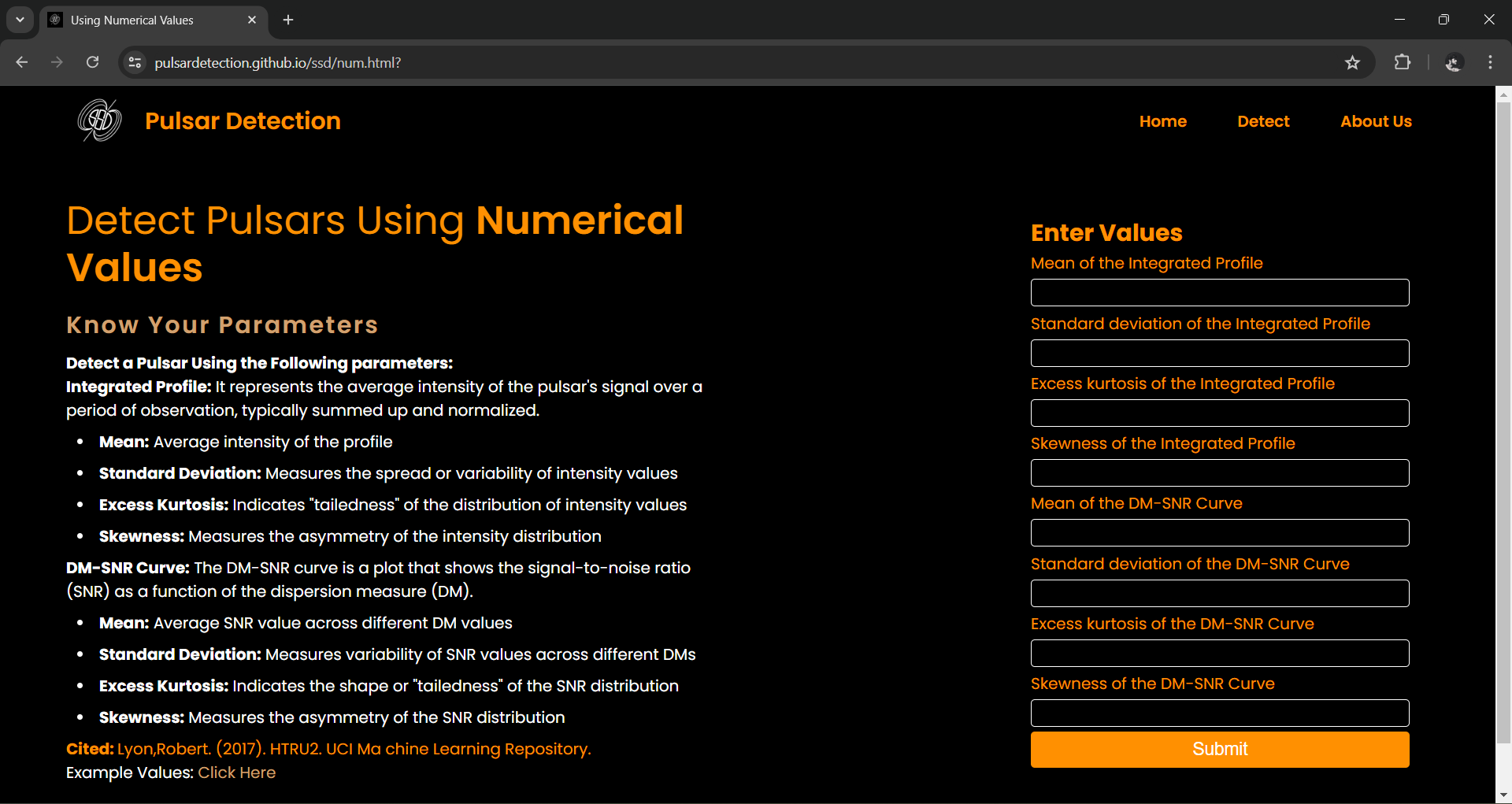
This is the Home page/Landing Page which has two sections – first one has the introduction and the last one has some description about pulsar and how are WE detectiong it. From Navbar anyone can go to the Detection page or About Us page. Our Github Repository is also linked here as Our Research work.



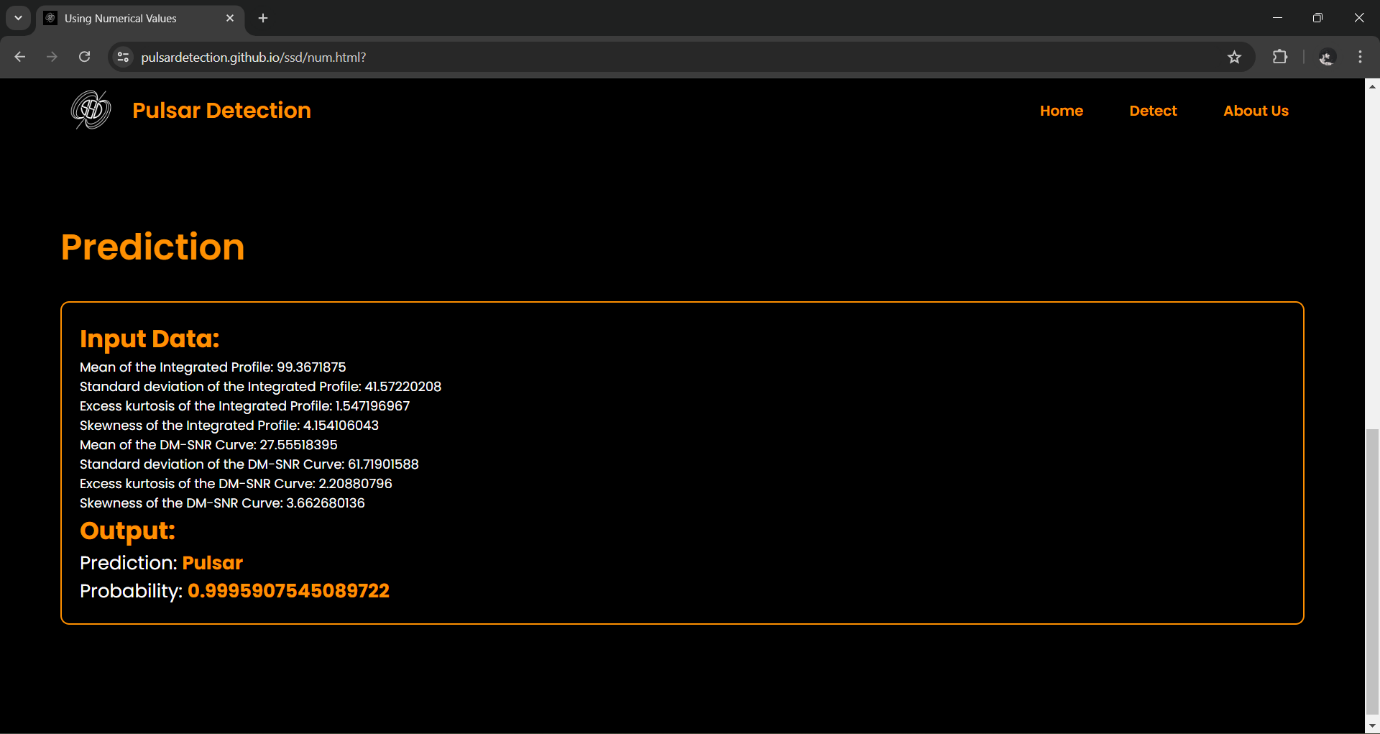
Now the next page is Detection page from where to detect a pulsar star, someone can choose from several methods based on the type of data they have. If they have numerical values in the HTRU2 format, they can select the numerical option. For image data in the HTRU1 format, they can choose the image-based detection method. If someone possess both numerical values and images, they can use a combined approach for more accurate results. Additionally, if someone has a PHCX file in the HTRU Medlat format, there is a specialized method tailored for this type of data. So, anyone can select the appropriate method based on their available data to efficiently detect pulsar stars.



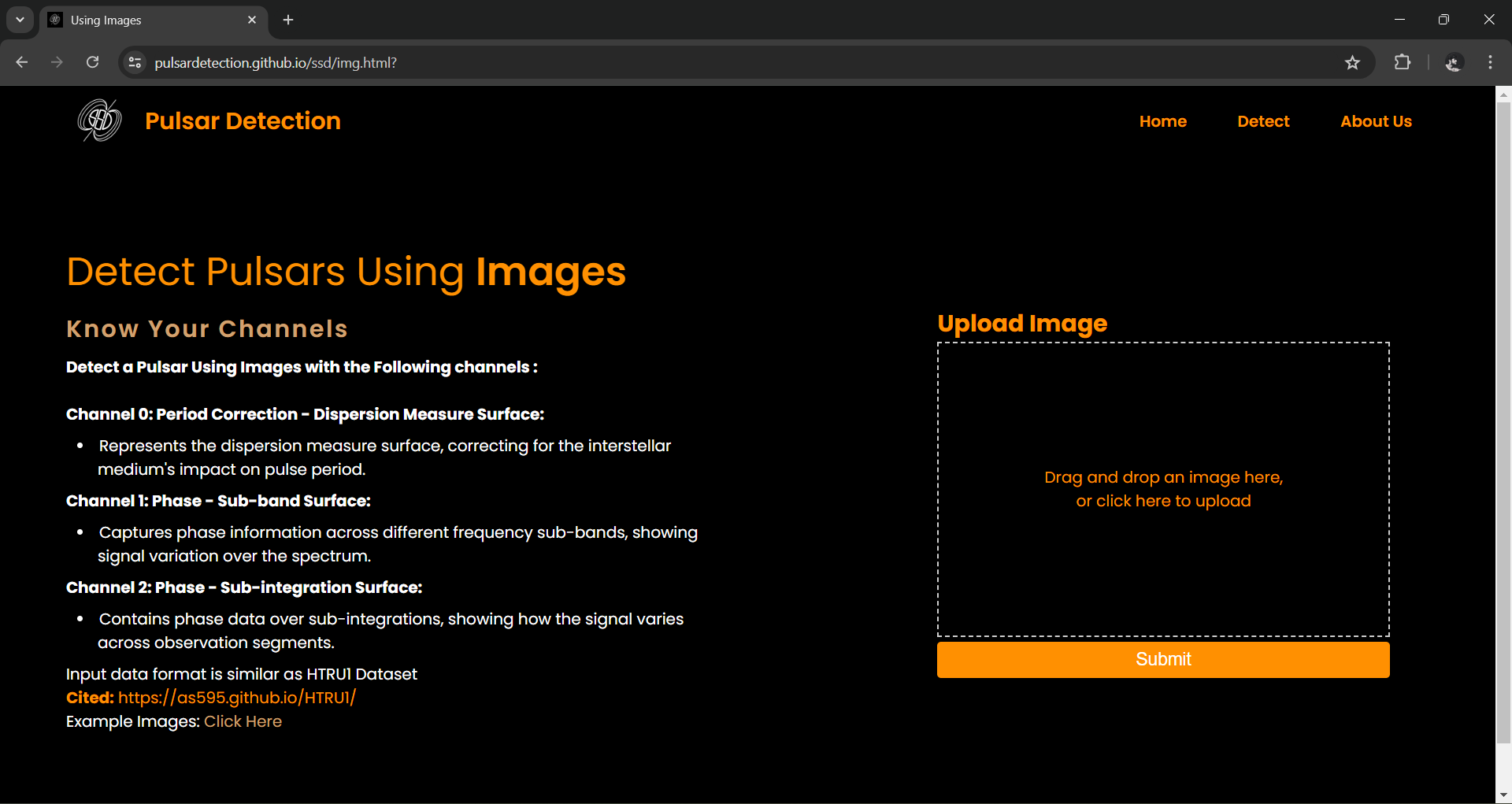
This is the page from where someone can detect a pulsar star by giving input of numerical values similar as HTRU2 format, which is given in the site.



Like this an output is printed, where the input data, prediction, and probability of that prediction is shown.

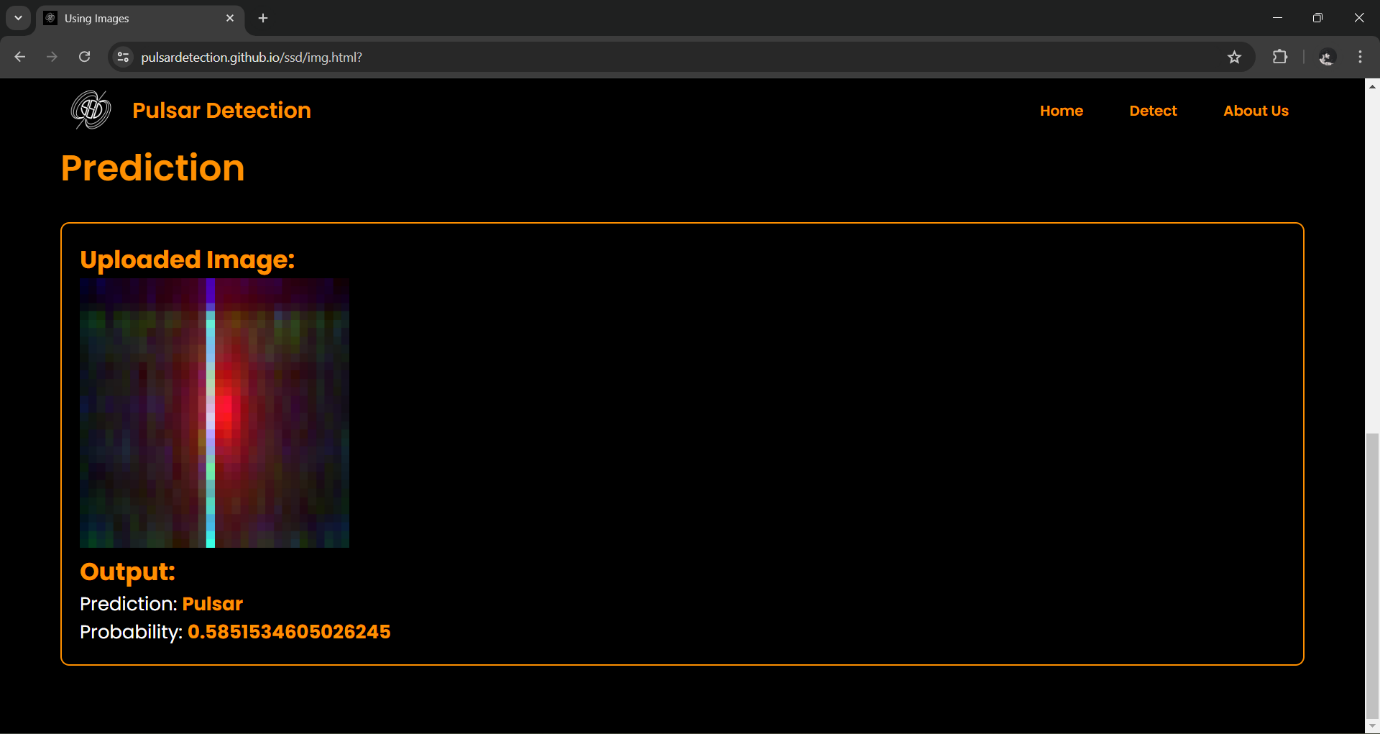


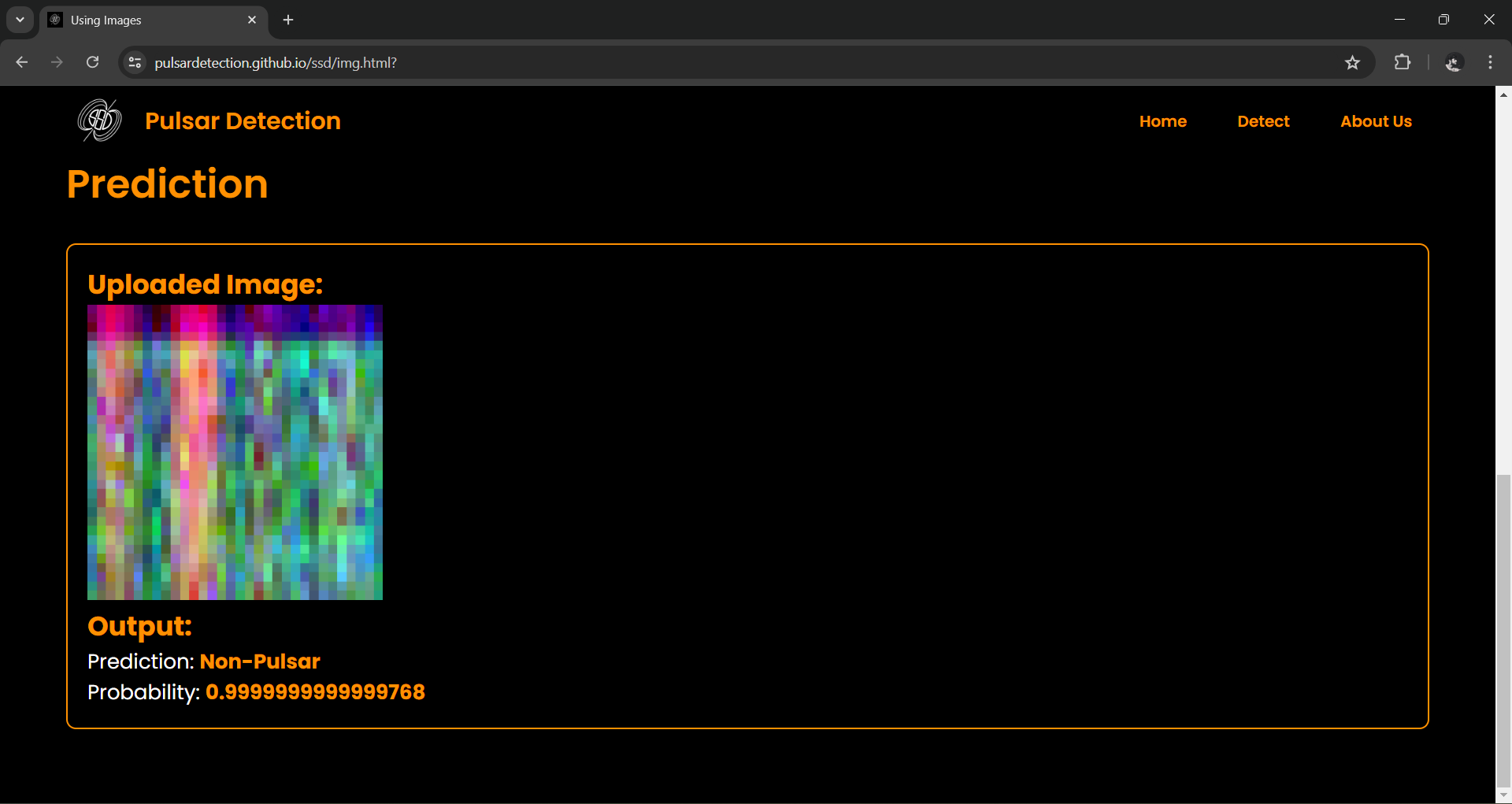
This is the page from where someone can detect a pulsar star by uploading an image having channels similar as HTRU1 format, which is given in the site.



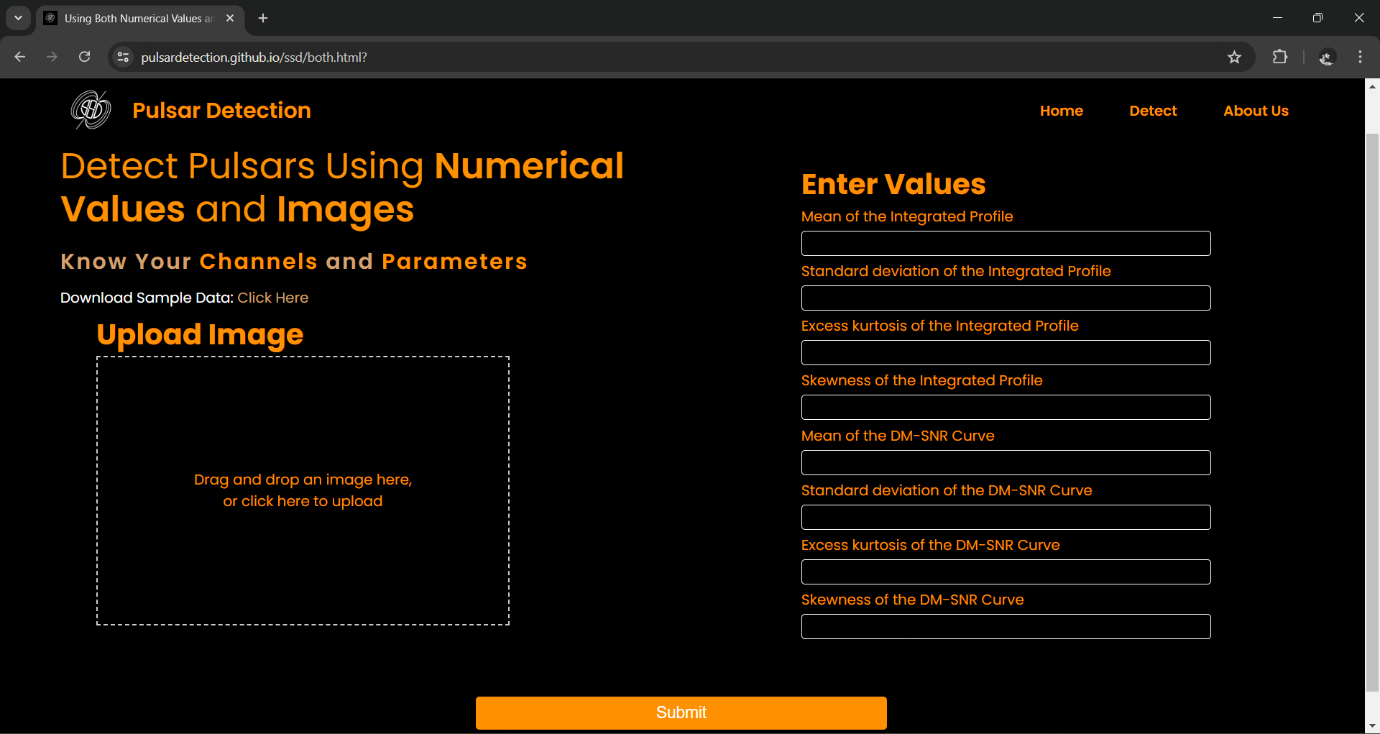
Like this an output is printed, where the uploaded image, prediction, and probability of that prediction is shown.

A pulsar -

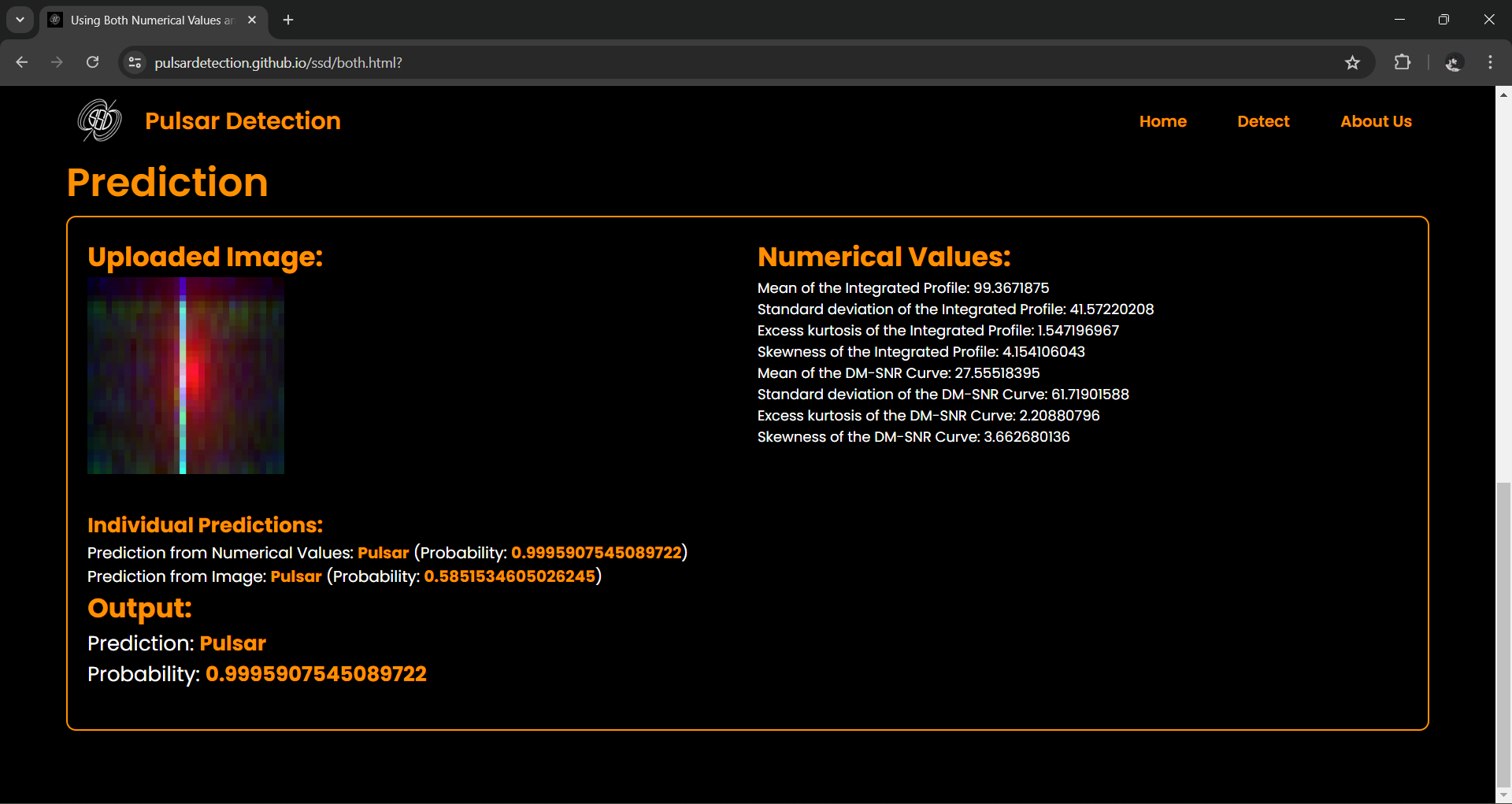


A non-pulsar - 

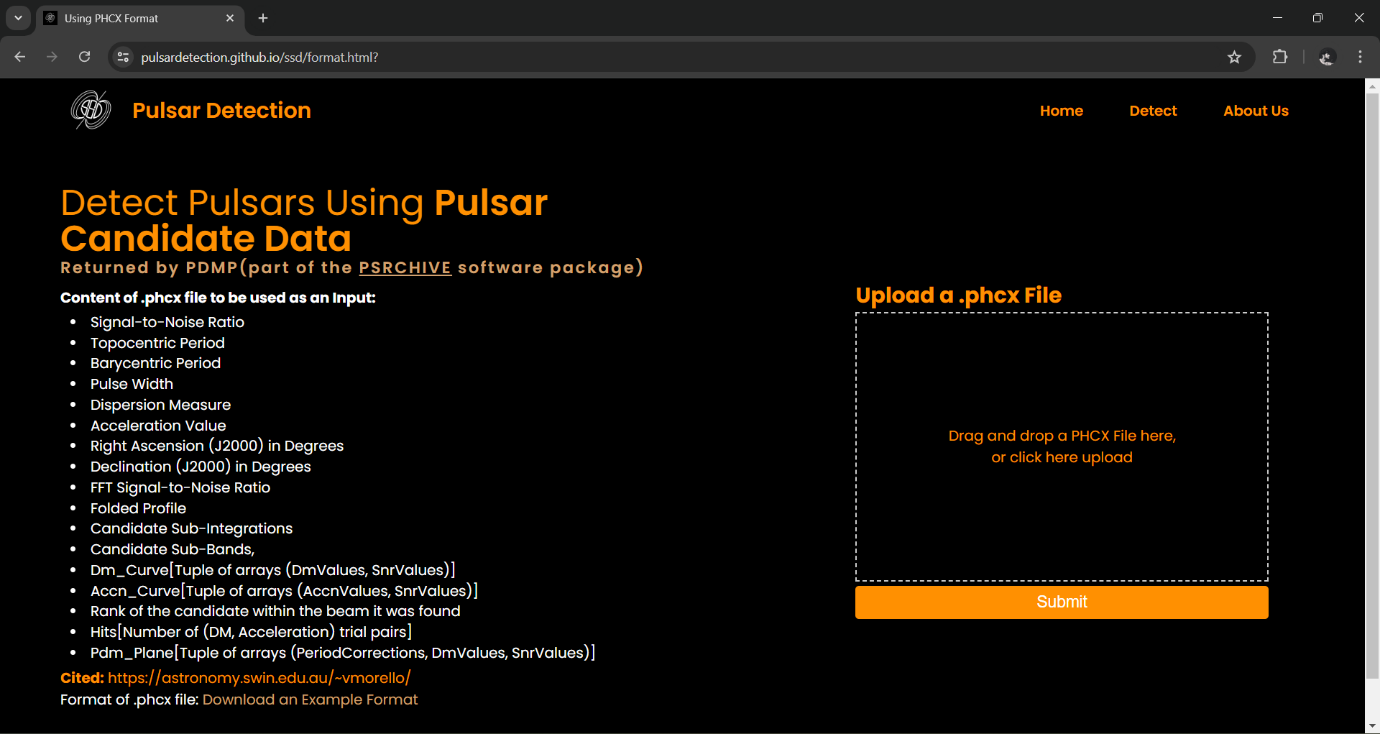
This is the page from where someone can detect a pulsar star if they have both of image having channels similar as HTRU1 format and numerical values similar as HTRU2 format, which is given in the site.



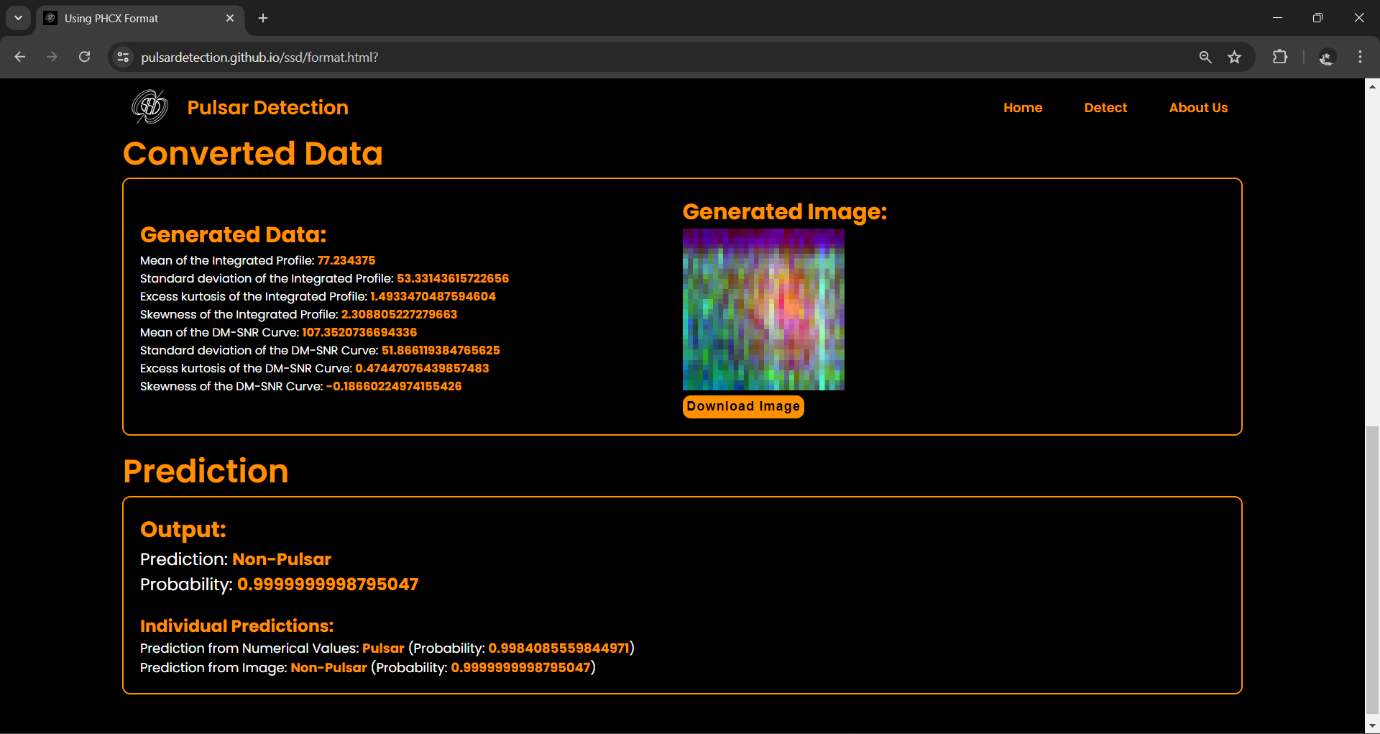
Like this an output is printed, where the uploaded image, input data, prediction from uploaded image and its probability, prediction from input data and its probability, overall prediction and probability of that prediction is shown.



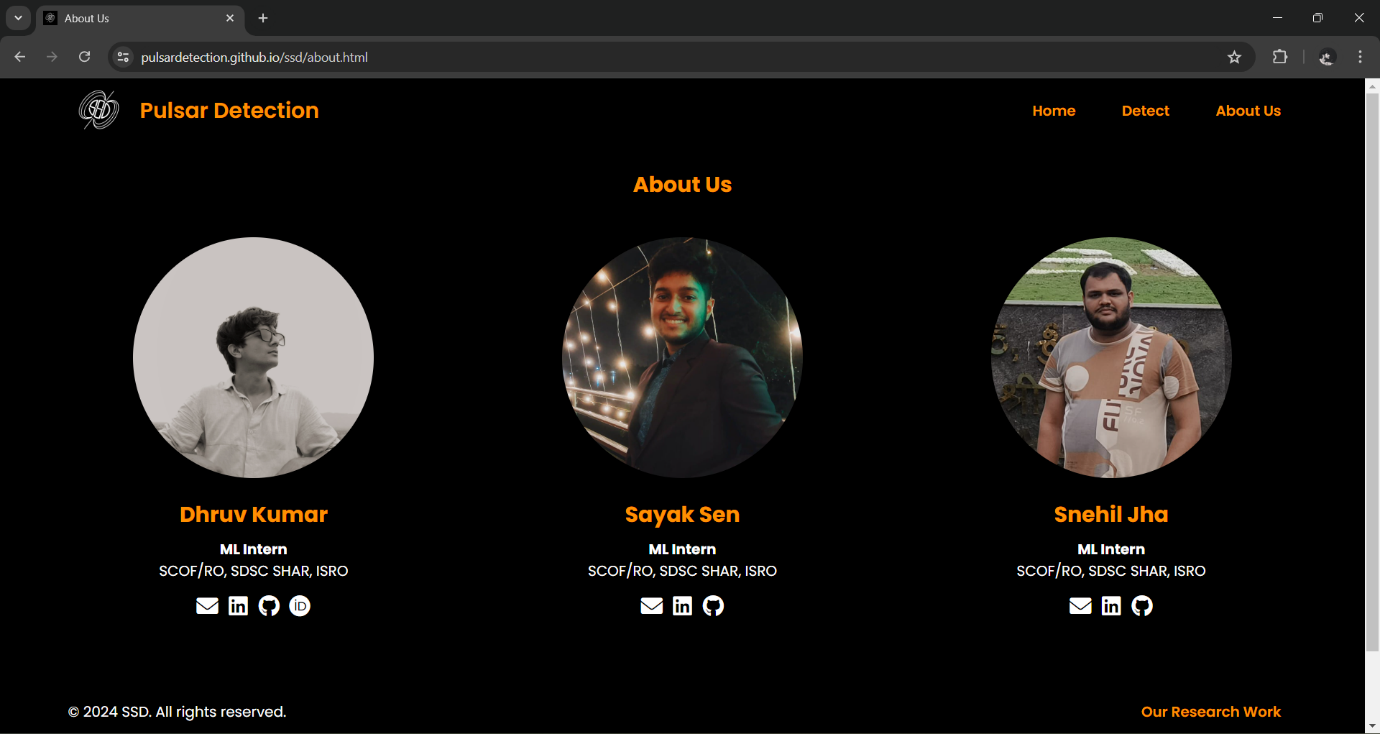
This is the page from where someone can detect a pulsar star if they have a .phcx document (similar as HTRU medlat datapoints) returned by PDMP(part of the PSRCHIVE software package) which is given in the site.



Like this an output is printed, where the generated numerical values from the .phcx document, generated image from the .phcx document, prediction from generated image and its probability, prediction from generated numerical values and its probability, overall prediction and probability of that prediction is shown.



Lastly, this is the About Us page where all the details of the people who have worked on this project are shown.



##### 4.2 Detailed Backend Functionality of the Pulsar Detection System

##### Overview

The backend of the pulsar detection system is built using Django, a high-level Python web framework. It handles the core functionalities, including processing user requests, managing data, interacting with machine learning models, and providing APIs for predictions. This section details how the backend functions, focusing on key components and their roles in the system.

##### Project Structure

A Django Backend Architecture is organized into several components, each serving a distinct purpose:

**Project Directory**: Contains settings, URL configuration, WSGI/ASGI application definitions, the model scripts and all the tools required to scale, convert and transform the raw data sent to the backend from the front-end and convert it into the predictions and probabilities leveraging the model’s capabilities.

**Applications**: Modular components within the project that handle specific functionalities, such as predictions, file conversions, image scaling, numerical data scaling, etc..

**Views**: Handle the logic for processing requests and returning responses.

**URLs**: Route incoming requests to the appropriate views.

##### Request Handling and Routing

When a user interacts with the pulsar detection web application, a series of HTTP requests are generated. Django handles these requests through a structured process:

**URL Routing**: Incoming HTTP requests are matched against predefined URL patterns in the urls.py file. Each pattern is associated with a specific view function or class.

**View Handling**: The matched view processes the request. Views execute the core logic, such as reading data sent from the Front End, interacting with machine learning models, or structuring responses.

**Response Generation**: After processing the request, the view returns an HTTP response, which is a JSON object which is sent via the Django REST framework.

##### Django REST Framework (DRF)

For the pulsar detection system, Django REST framework (DRF) is used to create RESTful APIs that handle interactions between the frontend and backend. DRF simplifies API development by providing tools and conventions for:

**Serializers**: Convert complex data types, such as querysets and model instances, to native Python data types that can be rendered into JSON or XML.

**Viewsets**: Define common behavior for multiple views, reducing boilerplate code.

**Routers**: Automatically generate URL patterns for viewsets, streamlining the process of defining API endpoints, mainly used in this project to redirect traffic from base Django URL to APIs app.

##### Machine Learning Model Integration

Django interfaces with machine learning models developed using PyTorch and exported to ONNX format. The following steps outline this integration:

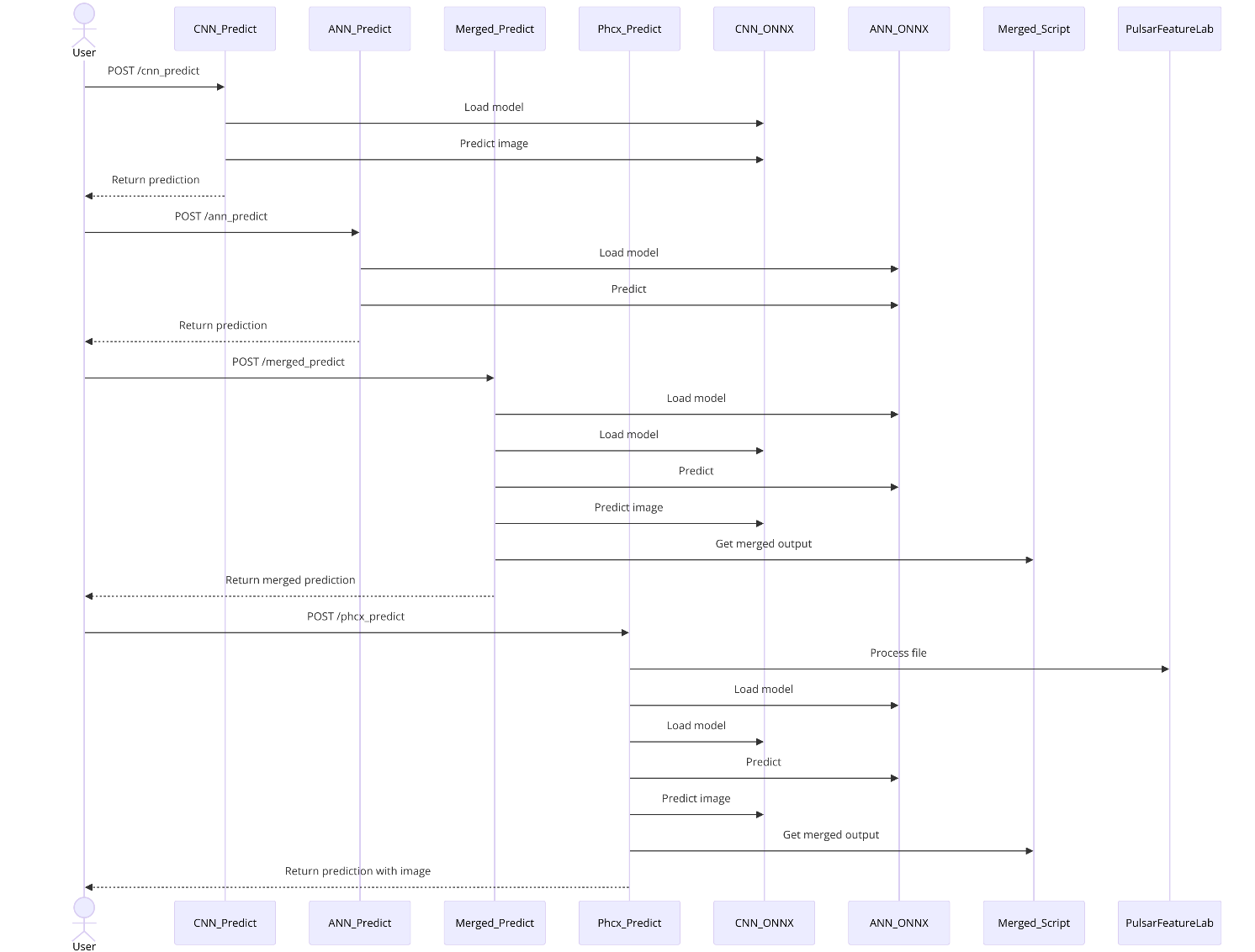
**Model Loading**: ONNX models are loaded into memory using onnxruntime when the server starts or on-demand when a prediction request is made.

**Data Preprocessing**: Input data is preprocessed to match the expected format of the models. For numerical data, it involves scaling using a standard scaler. For image data, it involves resizing, normalization, and tensor conversion.

**Inference**: The preprocessed data is fed into the loaded models to perform inference, which returns predictions and probabilities.

**Response Construction**: The predictions are packaged into a structured response and sent back to the client.

The following diagram represents all the API endpoints and interactions:

`

### Machine Learning Model Architecture and Development

#### **Introduction**

The discovery and study of pulsars, which are highly magnetized, rotating neutron stars emitting beams of electromagnetic radiation, are crucial for advancements in astrophysics. Detecting these celestial objects requires analyzing vast amounts of data collected from radio telescopes. Traditional methods of pulsar detection are time-consuming and often lack the precision needed to sift through the noise and identify true pulsar signals. This project aims to automate the detection process using advanced machine learning techniques. By leveraging both Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), we can analyze numerical data and image data, respectively, to enhance the accuracy and efficiency of pulsar detection. An ensemble learning approach further improves the reliability of predictions by combining the strengths of both models.

##### 4.3 Model Architecture

##### ****4.3.1 Artificial Neural Network (ANN)****

#### **Architecture**

The Artificial Neural Network (ANN) is meticulously designed to handle the numerical data associated with pulsar detection. The architecture is composed of three primary layers:

**Input Layer**: This layer receives 8 numerical features, which are the processed outputs of pulsar signal data.

**Hidden Layers**:

**First Hidden Layer**: Comprising 16 neurons, this layer employs ReLU (Rectified Linear Unit) activation functions. The ReLU activation introduces non-linearity into the model, enabling it to learn complex patterns within the data.

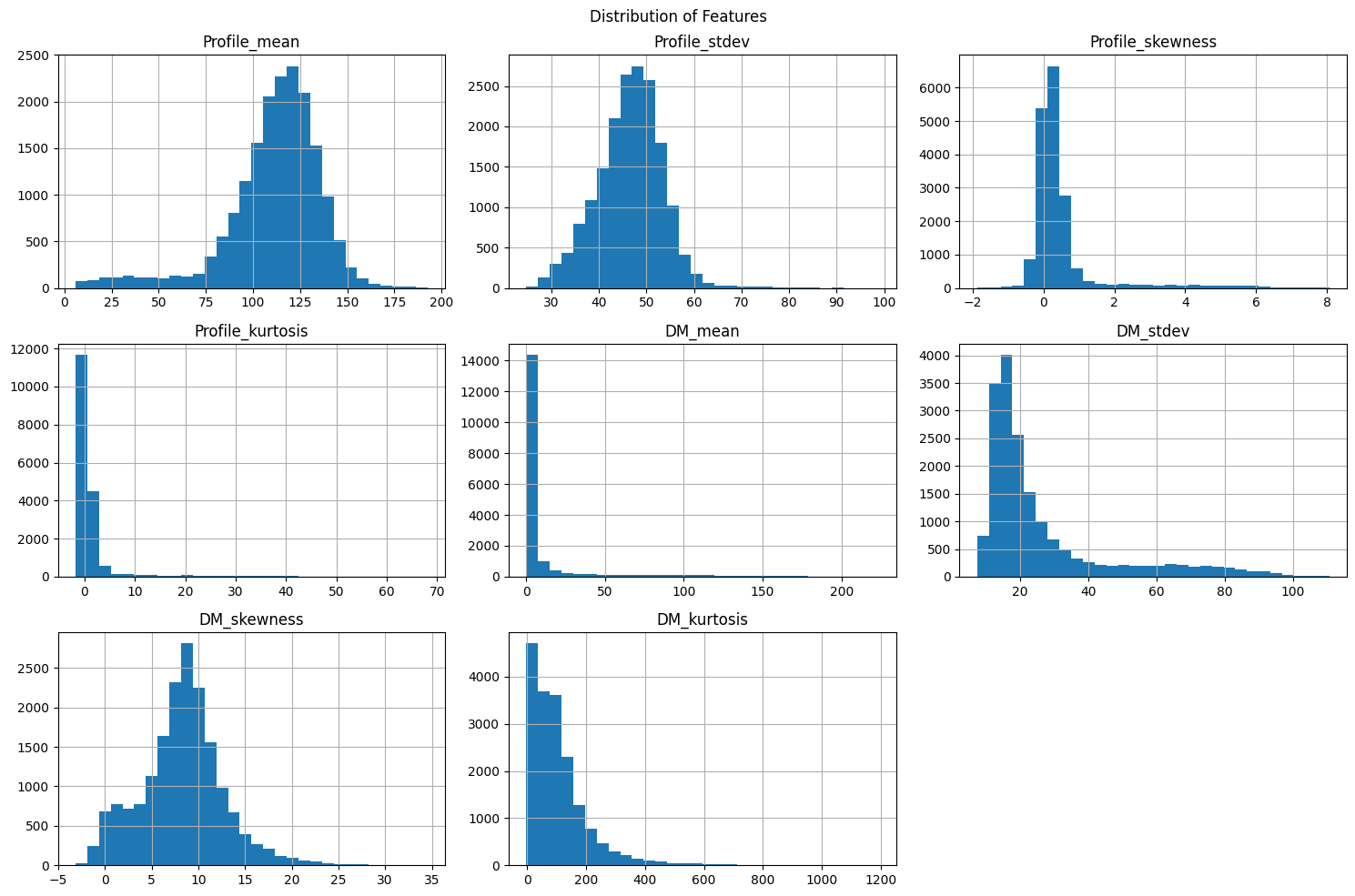
**Second Hidden Layer**: Another set of 16 neurons, also activated by ReLU functions, further refines the feature extraction process, allowing the network to build upon the representations learned in the first layer.

**Output Layer**: This final layer has a single neuron with a Sigmoid activation function, which outputs a probability score ranging between 0 and 1. This score determines the likelihood of the presence of a pulsar, facilitating binary classification.

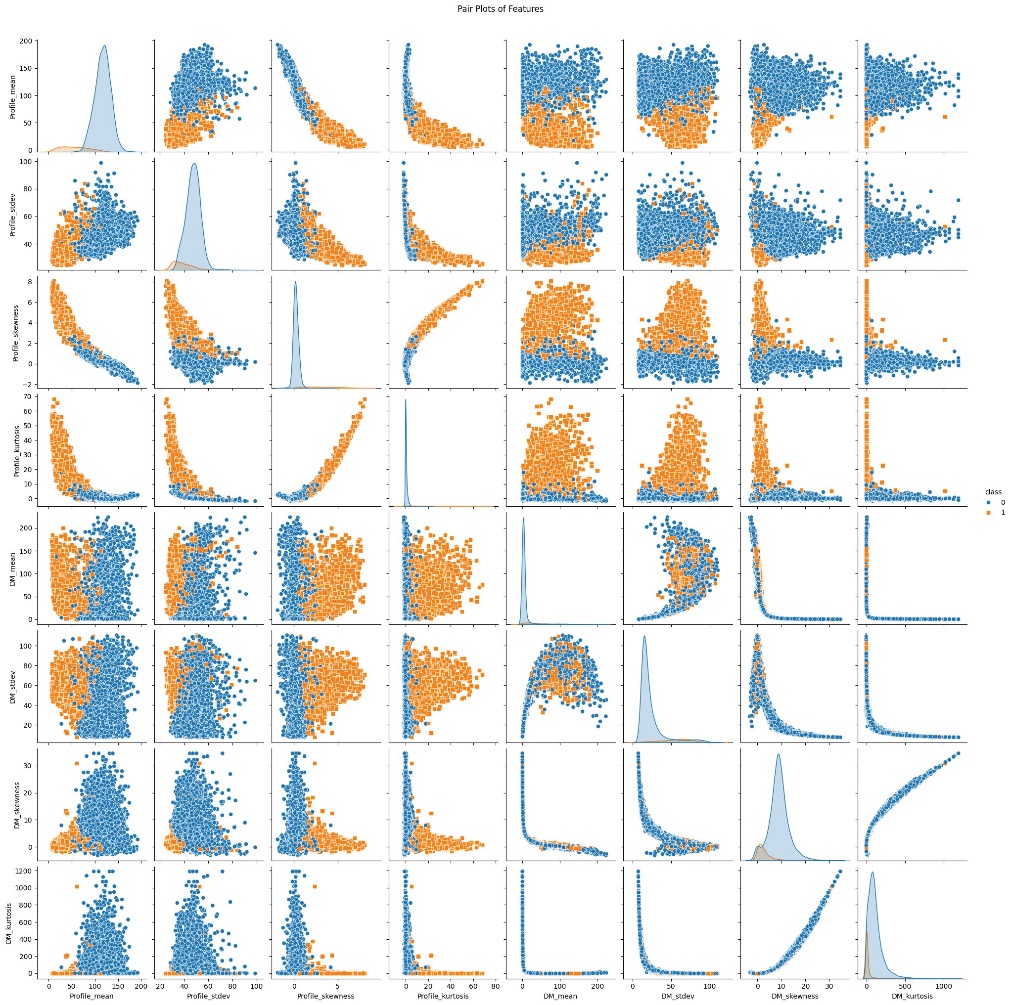
1. class ANN(nn.Module):
2. def \_\_init\_\_(self):
3. super(ANN, self).\_\_init\_\_()
4. layers = []
5. layers.append(nn.Linear(8, 16))
6. layers.append(nn.ReLU())
7. layers.append(nn.Linear(16, 16))
8. layers.append(nn.ReLU())
9. layers.append(nn.Linear(16, 1))
10. layers.append(nn.Sigmoid())
11. self.network = nn.Sequential(\*layers)
12. def forward(self, x):
13. return self.network(x)

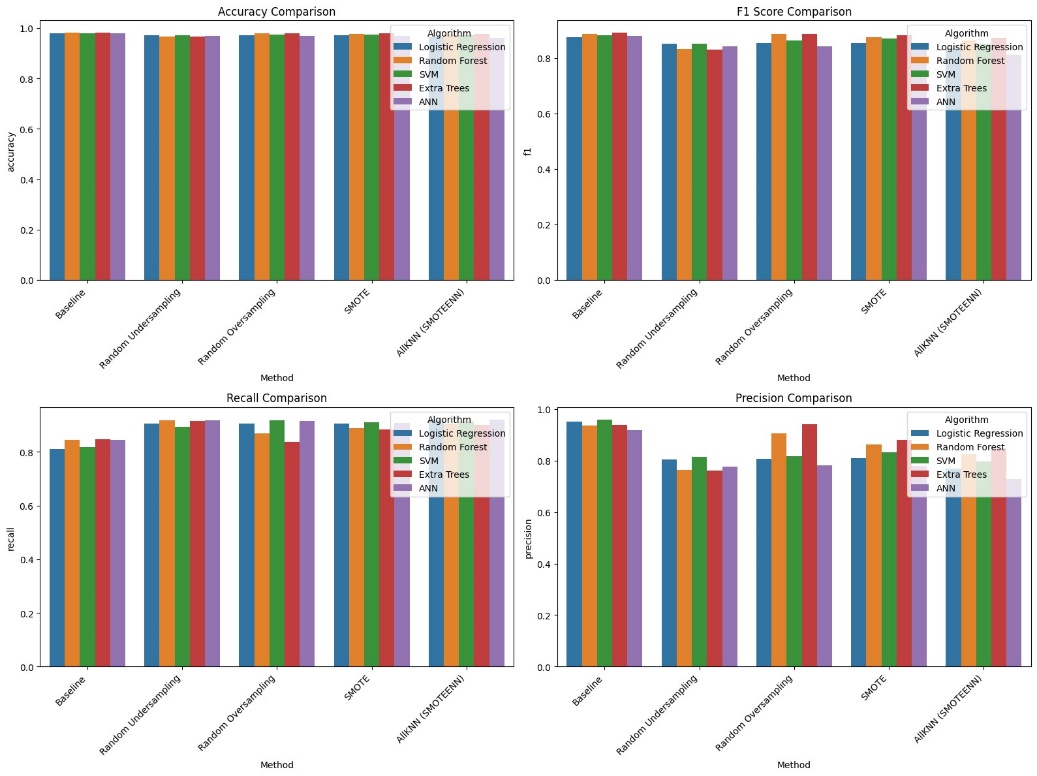
#### **Development Process**

* **Data Preprocessing**: The initial step in developing the ANN involves thorough preprocessing of the numerical data. This includes normalization, which scales the data to ensure each feature contributes equally during the training process. The normalized data is then converted into tensors, the preferred format for input into the ANN.



Distribution plot



* **Model Training**: The training process is a critical phase where the model learns to map the input features to the correct output (pulsar or not). This is achieved through:
* 

**Loss Function**: Binary Cross-Entropy Loss is utilized, which is well-suited for binary classification tasks. It measures the performance of the model by comparing the predicted probability with the actual class labels.

**Optimizer**: The Adam (Adaptive Moment Estimation) optimizer is employed to adjust the model's weights iteratively, minimizing the loss function. Adam is favored for its efficiency and ability to handle sparse gradients on noisy problems.

**Training Loop**: The training data is fed into the model in multiple epochs, with each epoch representing a complete pass through the entire dataset. During each epoch, the model's weights are updated based on the gradient of the loss function, progressively improving the model's accuracy.

**Model Evaluation**: After training, the ANN's performance is rigorously evaluated using a validation dataset. This step is crucial to ensure the model generalizes well to unseen data. Key performance metrics include:

**Accuracy**: The proportion of correct predictions out of the total predictions.

**Precision**: The ratio of true positive predictions to the total predicted positives, indicating the model's accuracy in identifying pulsars.

**Recall**: The ratio of true positive predictions to the total actual positives, reflecting the model's ability to detect all pulsar instances.

**F1 Score**: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

#### **Significance and Challenges**

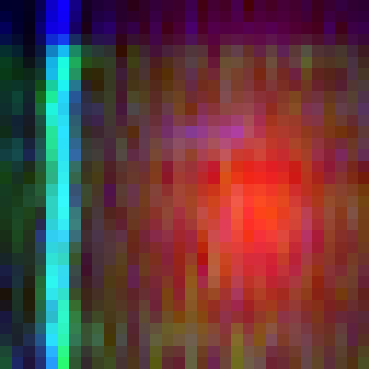
The ANN's role in pulsar detection is pivotal due to its capability to handle complex numerical data and uncover intricate patterns that might be missed by traditional methods. However, developing an effective ANN model involves challenges such as selecting appropriate hyperparameters, avoiding overfitting, and ensuring sufficient training data to achieve high accuracy.

### ****4.3.2. Convolutional Neural Network (CNN)****

#### **Architecture**

The Convolutional Neural Network (CNN) is tailored to process the image data related to pulsar detection. Its architecture includes several key components:

1. **Input Layer**: The network starts by receiving image inputs, which are resized to 32x32 pixels for consistency.

Pulsar Image Input Non Pulsar Image Input

1. **First Convolutional Layer**: This layer applies 32 filters to the input image. The filters, or kernels, detect various features such as edges and textures. The output of this layer is passed through a ReLU activation function, introducing non-linearity into the model. Max-pooling is then applied to reduce the spatial dimensions of the feature maps, retaining the most significant features while reducing computational complexity.
2. **Second Convolutional Layer**: Similar to the first, this layer applies 64 filters, followed by ReLU activation and max-pooling. This layer further refines the feature extraction process, capturing more complex patterns and details in the images.
3. **Fully Connected Layer**: After the convolutional layers, the feature maps are flattened into a single vector, which is then passed through a fully connected layer with 128 neurons. This layer, also activated by ReLU, integrates the features learned by the convolutional layers, preparing the data for the final classification step.
4. **Output Layer**: The final layer consists of a single neuron with a Sigmoid activation function. This neuron outputs a probability score indicating the likelihood of the image containing a pulsar.
5. class SimpleCNN(nn.Module):
6. def \_\_init\_\_(self):
7. super(SimpleCNN, self).\_\_init\_\_()
8. self.conv1 = nn.Conv2d(3, 32, kernel\_size=3, padding=1)
9. self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, padding=1)
10. self.fc1 = nn.Linear(64 \* 8 \* 8, 128)
11. self.fc2 = nn.Linear(128, 1)
12. def forward(self, x):
13. x = torch.relu(self.conv1(x))
14. x = torch.max\_pool2d(x, 2)
15. x = torch.relu(self.conv2(x))
16. x = torch.max\_pool2d(x, 2)
17. x = x.view(-1, 64 \* 8 \* 8)
18. x = torch.relu(self.fc1(x))
19. x = torch.sigmoid(self.fc2(x))
20. return x

#### **Development Process**

* **Data Preprocessing**: Image data undergoes several preprocessing steps before being fed into the CNN:
  + **Resizing**: Each image is resized to a standard dimension of 32x32 pixels to ensure uniformity.
  + **Normalization**: The pixel values of the images are normalized to a range suitable for the CNN, typically between -1 and 1, improving the convergence of the training process.
  + **Conversion to Tensors**: The preprocessed images are converted into tensors, the preferred format for input into PyTorch models.
* **Model Training**: Training the CNN involves several crucial steps:
  + **Data Augmentation**: Techniques such as rotation, flipping, and scaling are applied to the training images to artificially increase the size and variability of the dataset. This helps prevent overfitting and improves the model's generalization capability.
  + **Loss Function**: Similar to the ANN, the Binary Cross-Entropy Loss is used, which is appropriate for binary classification tasks. It measures the difference between the predicted probability and the actual class label.
  + **Optimizer**: The Adam optimizer is employed to iteratively update the model's weights, minimizing the loss function. Adam is chosen for its efficiency and ability to handle noisy gradients.
  + **Training Loop**: The training data is passed through the network in multiple epochs. During each epoch, the model's weights are updated based on the gradients of the loss function, progressively improving its accuracy.
* **Model Evaluation**: The trained CNN is evaluated using a validation dataset to ensure its performance on unseen data. Key performance metrics include accuracy, precision, recall, and F1 score, providing a comprehensive assessment of the model's capabilities.

##### ****Significance and Challenges****

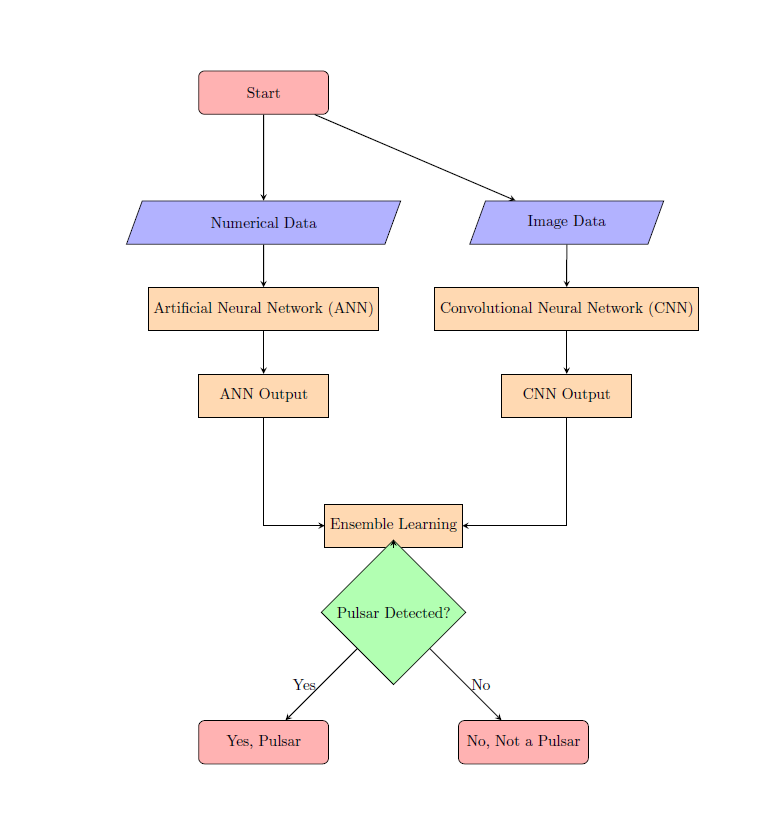
The CNN's ability to handle and process image data is crucial for the pulsar detection project, as it can identify visual patterns and features indicative of pulsars. However, developing an effective CNN involves challenges such as selecting appropriate filter sizes, layer depths, and handling large datasets to avoid overfitting while ensuring sufficient training data.

##### Multi-Model Ensemble

The multi-model ensemble approach integrates the predictions from both the Artificial Neural Network (ANN) and the Convolutional Neural Network (CNN) to enhance the accuracy and reliability of pulsar detection. This strategy capitalizes on the strengths of each model, allowing for a more comprehensive analysis of the data.

##### Architecture and Development

The ensemble model operates by combining the outputs of the ANN and CNN, leveraging their respective strengths in processing numerical and image data. The development process includes the following key steps:



Multi-Model Architecture

1. **Probability Extraction**: After training, both models output probability scores indicating the likelihood of the presence of a pulsar. These scores are crucial for the ensemble decision-making process.
2. **Decision Rules**: A set of predefined rules is applied to combine the probability scores from both models to arrive at a final prediction. The decision rules are designed to account for the confidence levels of each model, ensuring that the most reliable output is selected.

**Ensemble Strategy**

The ensemble strategy can be categorized into specific cases based on the probability outputs from the ANN and CNN:

* **Case 1**: If both models output a probability of 0.5, the ensemble predicts the presence of a pulsar. This case indicates a neutral stance, where neither model shows strong confidence, yet they agree on a baseline detection.
* **Case 2**: If both models provide probabilities greater than or equal to 0.5, the ensemble confidently predicts the presence of a pulsar. This scenario reflects a strong consensus between the models.
* **Case 3**: If both models output probabilities less than 0.5, the ensemble predicts the absence of a pulsar. This indicates a clear lack of evidence for pulsar presence.
* **Case 4 and 5**: In situations where the models produce mixed probabilities (one model predicts a pulsar while the other does not), the ensemble decision is influenced by the model with the higher confidence score. This approach ensures that the final prediction is weighted towards the more reliable model, enhancing overall decision accuracy.

##### Significance and Challenges

The multi-model ensemble approach significantly improves the robustness of pulsar detection systems. By combining the outputs of the ANN and CNN, it mitigates the risk of false positives and negatives that may arise from relying on a single model. However, challenges remain in optimizing the decision rules and ensuring that the ensemble effectively balances the contributions of both models. In summary, the multi-model ensemble leverages the unique capabilities of ANN and CNN to provide a comprehensive solution for automated pulsar detection, enhancing both accuracy and reliability in identifying these celestial objects.

##### **CONCLUSION**

The development of the multi-model ML architecture for automated pulsar detection represents a significant advancement in the field of astrophysics, leveraging the power of both artificial neural networks (ANN) and convolutional neural networks (CNN). The integration of these models into a cohesive system has resulted in a robust and highly accurate method for detecting pulsars.

On the frontend, the application is designed to be user-friendly and accessible. Utilizing HTML for structure, CSS for styling, and JavaScript for interactivity, the interface ensures that users can easily upload data, select detection methods, and view results. The seamless user experience is further enhanced by incorporating responsive design principles, ensuring the application is accessible across various devices.

The backend, powered by Django, provides a secure and scalable environment for the application. Django's robust framework supports efficient data handling and storage through its ORM, and the Django REST framework (DRF) facilitates smooth communication between the frontend and backend via RESTful APIs. These APIs manage the requests for model predictions and data processing, ensuring efficient and accurate delivery of results.

The core of the project lies in the model architecture, where the ANN and CNN work together to analyze numerical and image data respectively. PyTorch is employed to develop and train these models, capitalizing on its dynamic computation graph and GPU acceleration to handle extensive datasets. The ensemble strategy combines the strengths of both models, resulting in an overall **accuracy of 97.8%** and **recall of 95.3%**, making it a reliable tool for automated pulsar detection.

In conclusion, this project successfully integrates advanced machine learning techniques with a well-designed web application to provide an efficient and accurate pulsar detection system. The synergy between the frontend, backend, and model components demonstrates the potential of interdisciplinary approaches in solving complex scientific problems. This work not only advances the field of pulsar detection but also sets a foundation for future enhancements and applications in astronomical research.

**REFERENCE**

1. R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles, "Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach," Monthly Notices of the Royal Astronomical Society, vol. 459, no. 1, pp. 1104–1123, 2016.
2. T. P. Ranaivomanana, "Building a Deep Generative Model for Surveyed Pulsar Classes," M.Phil. thesis, Faculty of Science and Engineering, Univ. of Manchester, Manchester, U.K., 2021.
3. E. Lee, F. Rustam, W. Aljedaani, A. Ishaq, V. Rupapara, and I. Ashraf, "Predicting Pulsars from Imbalanced Dataset with Hybrid Resampling Approach," Journal of Healthcare Engineering, vol. 2021, Article ID 6625345, 10 pages, 2021.
4. M. J. Keith, A. Jameson, W. van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. D. Bates, N. D. R. Bhat, M. Burgay, S. Burke-Spolaor, N. D’Amico, L. Levin, P. L. McMahon, S. Milia, and B. W. Stappers, "The High Time Resolution Universe Pulsar Survey I: System configuration and initial discoveries," Monthly Notices of the Royal Astronomical Society, vol. 409, no. 2, pp. 619-627, 2010.