AI-BASED MULTIPLE-DISEASE DETECTION USING MACHINE LEARNING AND CONVOLUTIONAL NEURAL NETWORKS

The project report submitted in the partial fulfilment of the requirement.

For the award of the degree of Bachelor of Technology

In

COMPUTER SCIENCE AND SYSTEMS ENGINEERING

By

GORLA SUKANYA	321114110021
JAKKULA MADHURIMA	321114110022
JEERI ALEKHYA REDDY	321114110023
KALLAKURI VYSHNAVI	321114110024

Under the Esteemed Guidance of DR. K. SOUMYA

Assistant Professor(c)

Department of Computer Science and Systems Engineering, AUCEW

ANDHRA UNIVERSITY

COLLEGE OF ENGINEERING FOR WOMEN

Visakhapatnam – 530017

April, 2025

DECLARATION

We hereby declare that the project "**AI-BASED MULTIPLE-DISEASE DETECTION"** has been submitted by us in partial fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Systems Engineering by ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN, Visakhapatnam.

We also hereby declare that this project is the result of our own effort and that it has not been submitted to any other University for the award of any Degree.

GORLA SUKANYA (321114110021)

JAKKULA MADHURIMA (321114110022)

JEERI ALEKHYA REDDY (321114110023)

KALLAKURI VYSHNAVI (321114110024)

CERTIFICATE

ANDHRA UNIVERSITY

COLLEGE OF ENGINEERING FOR WOMEN VISAKHAPATNAM

This is to certify that this project entitled "AI-Based Multiple-Disease Detection" is a Bonafide carried out by **GORLA SUKANYA (321114110021), JAKKULA MADHURIMA (321114110022), JEERI ALEKHYA REDDY (321114110023), KALLAKURI VYSHNAVI (321114110024)** submitted in partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Systems Engineering during the academic year 2024-2025.

Project Guide	Head of the Department
Dr. K. Soumya,	Prof. B. Prajna,
Assistant Professor (c),	Head of the Department,
Dept. of CS&SE,	Dept. of CS&SE,
AUCEW,	AUCEW,
Visakhapatnam	Visakhapatnam

ACKNOWLEDGEMENT

We take this opportunity to express deep gratitude to the people who have been instrumental in the successful completion of the project. We are thankful to **ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN**, for giving us the opportunity to work on a project as a part of the curriculum.

We would like to thank our project guide **Dr. K. SOUMYA,** Assistant Professor$^{(c)}$, Department of Computer Science and Systems Engineering, Andhra University College of Engineering for Women for the valuable guidance and suggestions, throughout the period of project work. We express our deep sense of gratitude to our beloved Head of the Department **Prof. B. PRAJNA**, Andhra University college of Engineering for Women for permitting us to carry out this project. We would like to show our gratitude to our beloved Principal, **Prof. PADMASREE** for her encouragement in the aspect of our course, Andhra University College of Engineering for Women, who gave us official support for the progress of our project. We would like to thank our project mentor **Dr. S. PALLAM SETTI,** Founder & CEO of Dr Pallam Setti Center for Research & Technology, A-hub, Andhra University.

We would also like to extend our gratitude to Lab Technicians and our sincere gratitude towards all the faculty members and Non-Teaching Staff in the Department of Computer Science and Systems Engineering for supporting us. We would also like to thank our parents and friends, who have willingly helped us out with their abilities in completing this project.

With gratitude,

GORLA SUKANYA (321114110021)

JAKKULA MADHURIMA (321114110022)

JEERI ALEKHYA REDDY (321114110023)

KALLAKURI VYSHNAVI (321114110024)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
Batch Norm	Batch Normalization
CNN	Convolutional Neural Networks
CSV	Comma-Separated Values
DL	Deep Learning
Dropout	Dropout Regularization
F1-Score	F1 Measure (Harmonic mean of Precision & Recall)
FP	False Positive
FN	False Negative
GPU	Graphics Processing Unit
I/O	Input/Output
IoT	Internet of Things
JSON	JavaScript Object Notation
Keras	Deep Learning Library in Python
KNN	K-Nearest Neighbours
Leaky ReLU	Leaky Rectified Linear Unit
LR	Learning Rate
LSTM	Long Short-Term Memory
ML	Machine Learning
MSE	Mean Square Error
NLP	Natural Language Processing
PCA	Principal Component Analysis
PyTorch	Python-based Deep Learning Framework
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic Curve
Softmax	Softmax Activation Function
SVM	Support Vector Machine
TF	TensorFlow
TP	True Positives
TN	True Negatives
TPU	Tensor Processing Unit
UX/UI	User Experience / User Interface
XGBoost	Extreme Gradient Boosting

ABSTRACT

The quick expansion of technological knowledge requires Artificial Intelligence to function as a fundamental component in modern human daily activities. The human race has reached a stage where Artificial Intelligence can identify diseases through simple input data or symptom descriptions. People can detect diseases with high precision through this AI-Based Multi-Disease Detection system. The AI-based system employs Convolutional Neural Networks (CNN) for medical image feature extraction while using multiple machine learning (ML) techniques (such as Random Forest, Support Vector Machine (SVM), Logistic Regression, and XGBoost) to process sequential data for improved accuracy and model efficiency.

The system receives training through multiple datasets that incorporate medical images and patient health records and time-series information. The system's diverse input data allows it to detect multiple diseases including Breast Cancer, Heart Disease, Kidney Disease, Liver Disease, Diabetes, Alzheimer's, Malaria, Pneumonia, Parkinson's and Common Flu. The hybrid approach that combines CNN and RNN enables feature learning and temporal pattern recognition and optimization capabilities for real-world healthcare and clinical applications. The experimental findings showed excellent precision together with minimal incorrect results and performance across different datasets. The future development plans consist of extending the model to analyze complex diseases while deploying mobile applications and offline access

CHAPTER-1. INTRODUCTION

1.1 INTRODUCTION TO PROJECT WORK:

Recent advancements in Artificial Intelligence (AI) implementation have fundamentally transformed diagnostic and treatment methodologies within healthcare. While early and precise diagnosis remains critical for optimal patient outcomes, conventional approaches often require costly timelines, specialized personnel, and resource-intensive processes, barriers our initiative directly addresses.

The project **'AI-Based Multi -Disease Detection System'** seeks to create an intuitive diagnostic platform capable of identifying nine conditions: heart disease, kidney disease, diabetes, Parkinson's disease, Alzheimer's disease, common flu, breast cancer, malaria, and pneumonia. Through strategic alignment of predictive analytics and user experience design, the solution empowers preliminary self-assessment while maintaining clinical rigor.

Users submit medical data through structured interfaces, triggering analysis by condition-specific AI models. Not merely an automation tool, but a risk mitigation framework. This approach deliveries actionable insights for informed healthcare decisions, particularly valuable in resource-constrained environments.

Architecturally, the system leverages deep learning frameworks, **Convolutional Neural Networks (CNNs)** process imaging data (X-rays, MRIs) for Alzheimer's and pneumonia detection, while **Recurrent Neural Networks (RNNs)** analyze temporal health metrics relevant to diabetes and cardiac conditions. Hybrid model integration ensures cross-disciplinary diagnostic accuracy.

Accessibility remains paramount. By minimizing technical barriers through conversational UI design, the platform democratizes advanced diagnostics, a paradigm shift in preventive care delivery. As noted above, this underscores our commitment to healthcare equity through technological innovation.

1.2 PROBLEM STATEMENT:

Despite the advancements in AI and machine learning for disease detection, most existing systems are limited to identifying a single disease. This lack of versatility forces individuals to rely on multiple tools for different conditions, leading to increased complexity and inefficiency in the diagnostic process. Additionally, many AI-based diagnostic applications are not user-friendly, making them difficult to access for individuals without technical expertise. So, this project aims to address these challenges by developing an AI-based multi-disease detection system using Machine Learning with CNNs through a user-friendly web application.

1.3 RESEARCH CONTRIBUTIONS

The significance of this project work lies in its potential to enhance early disease detection and improve clinical decision-making across diverse medical conditions. The research findings establish a foundation for future multi-disease diagnostic systems, with opportunities to expand the range of detectable diseases and refine predictive accuracy through additional data and algorithmic advancements. The step-wise research contributions are as follows:

1.3.1 Multi-disease framework: Unlike single-disease systems, our model handles ten diseases using heterogeneous data sources, including medical images and structured clinical reports (CSV files), showcasing versatility.

1.3.2 Hybrid artificial intelligence approach: Our Project combines various Machine Learning algorithms like Support Vector Machines (SVM), Extreme Gradient Boosting (XGBoost), Random Forest Classifier and CNN technologies for robust performance across tabular and image data.

1.3.3 Real-time data handling: Our system demonstrates adaptability to real-time, enabling dynamic disease prediction. By handling real-time inputs, our model addresses the limitations of static data and offers a diverse solution for modern healthcare challenges.

1.3.4 Web interface: A major contribution of this work is the design and implementation of a user-friendly web interface that seamlessly integrates our AI model. Users can select a specific disease, input relevant data, and receive accurate predictions, making advanced diagnostics accessible to non-expert users such as healthcare providers or patients.

1.4 PROJECT DELIVERIES:

1.4.1 User-friendly web application:

- A fully functional web application that allows users to input medical data (symptoms, diagnostic reports, health metrics) to detect multiple diseases.
- Responsive and intuitive UI/UX design for easy navigation, even for non-technical users.

1.4.2 Artificial Intelligence-based multi-disease detection models:

- Convolutional Neural Network (CNN) Models for image-based disease detection (e.g., pneumonia, malaria, Alzheimer's).
- Optimized models trained on disease-specific datasets with high accuracy, precision, recall, and F1-scores.

1.5 LIMITATIONS OF EXISTING SYSTEM:

1.5.1 Single-disease focus:

Most existing AI-based diagnostic systems are designed to detect only one specific disease, requiring separate tools for different conditions. This fragmentation increases the complexity making the diagnostic process inefficient.

1.5.2 Lack of generalization:

Many machine learning models struggle to generalize across diverse datasets or different populations. Models trained on specific datasets may not perform well when applied to new, unseen data from different demographics, reducing reliability.

1.5.3 Inadequate handling of multi-modal data:

Existing techniques often struggle to integrate different types of medical data (e.g., images, clinical text, lab results) effectively. This limits the ability to provide a comprehensive diagnosis based on multiple data sources.

1.5.4 Limited accessibility and user-friendliness:

Many AI-powered healthcare tools are designed for medical professionals who have technical knowledge, making them inaccessible to the general public, especially in rural areas.

1.6 PROPOSED SYSTEM:

1.6.1 Integrated multi-disease detection

Unlike traditional systems that focus on a single disease, the proposed system can detect multiple diseases simultaneously within a single platform. It eliminates the need for separate diagnostic tools, reducing complexity and improving efficiency for patients. Each disease is supported by specialized machine learning models to its unique characteristics, enabling accurate diagnosis.

1.6.2 Multi-modal data processing

The system is designed to handle multi-modal medical data, including clinical records, medical images (e.g., X-rays, MRIs), lab results, and patient-reported symptoms. CNNs are used for processing image-based data. This comprehensive data integration enables the system to provide holistic diagnostic insights rather than relying on a single data source.

1.6.3 User-friendly web application interface

To address accessibility issues, the proposed system features a simple and intuitive web-based application that caters to both healthcare professionals and the general public. The user interface is designed for easy navigation, allowing users to input medical data, receive instant predictions, and understand diagnostic results without requiring technical expertise. This improves accessibility, especially in remote or underdeveloped regions with limited healthcare resources.

1.6.4 Real-time predictive analysis and decision support

The system incorporates real-time predictive analytics powered by advanced machine learning algorithms, enabling it to not only detect diseases but also predict potential health risks based on historical and current patient data. It provides healthcare professionals with actionable decision support by highlighting critical insights.

1.6.5 Scalability and adaptability for future expansion

The proposed system is built with a modular architecture, allowing it to scale and adapt to include additional diseases or data types as new medical research and datasets become available. This ensures long-term relevance and flexibility, enabling the platform to evolve with advancements in machine learning techniques and healthcare

needs, such as integrating emerging diagnostic markers or supporting new imaging technologies.

1.7 ORGANIZATION OF THE WORK

- Chapter 2 explains the literature review.
- Chapter 3 explains methodologies of the project.
- Chapter 4 deals with the system design.
- Chapter 5 deals with system analysis.
- Chapter 6 deals with system implementation.
- Chapter 7 deals with system testing.
- Chapter 8 deals with results and analysis of the project.
- Chapter 9 gives the conclusion and future scope of the project.

1.8 CONCLUSION

The overview of the entire project work is presented in chapter 1.

CHAPTER-2: REVIEW OF LITERATURE

2.1 INTRODUCTION

This chapter deals with the comprehensive review of the existing studies which helps to understand the progress in AI-based multi-disease detection system, evaluate different model's performance, and identify challenges that need further exploration.

2.2 List of Literature Reviews

2.2.1 Review of literature on Multiple Disease Prediction using Machine Learning

Shickel, B., et al., [1] states that Machine learning facilitates multi-disease prediction by integrating diverse patient data, including imaging, lab results, and clinical histories. It demonstrates the power of data-driven insights in improving healthcare outcomes.

Parshant, et al., [2] explores the application of Machine learning algorithms in predicting multiple diseases, focusing on their benefits, challenges and future directions. By utilizing publicly available datasets and appropriate feature engineering techniques, a comprehensive dataset was constructed encompassing relevant demographic, clinical, and biomarker information. This paper specifically focuses on the application of SVMs for multi-disease prediction.

2.2.2 Review of Literature on Medical Image Analysis using Convolutional Neural Networks (CNNs)

Litjens, et al., [3] highlights the role of neural networks (CNNs) to detect subtle patterns in imaging data such as MRI, CT, and X-rays. This approach significantly improves diagnostic accuracy for conditions like cancer and neurological disorders, often surpassing traditional methods. Its adaptability across modalities makes it a cornerstone of modern medical imaging research.

2.2.3 Review of Literature on Heart Disease Prediction

Chitambaram T(VIT), et al., [4] analyses the detection of heart disease detection using ML algorithms and python programming. Using different types of parameters

(Chest pain (cp), resting blood pressure, cholesterol, resting electrocardiographic results, fasting blood sugar(fbs), maximum heart rate, exercise induced angina, ST depression, slope of the peak exercise ST segment, ECGs) this system predicts the cardiac-disease. Comparing 4-machine learning algorithms such as SVM (86.2%), Decision tree (68.4%), Random Forest classifier (71.4%) and K- nearest neighbor(63.4%) to get the better accuracy to which highest parameter may cause disease.

2.2.4 Review of Literature on Alzheimer's Disease Prediction

Al Guilherme Folego, et al., [5] introduces ADNet, a 3D convolutional neural network (CNN) that classifies Alzheimer's disease (AD), mild cognitive impairment (MCI), and cognitively normal (CN) states using whole-brain structural MRI scans. Trained on ADNI data and evaluated on the CADDementia challenge, ADNet achieves 51.4% accuracy, while its domain-adapted version, ADNet-DA, reaches 52.3%. Unlike prior methods, it operates fully automatically without domain-specific knowledge, processing scans in under 15 minutes—far faster than the 19-hour state-of-the-art. The approach highlights key brain regions like the temporal lobe and offers a promising, scalable tool for AD diagnosis and beyond.

2.2.5 Review of Literature on Fatty Liver Disease

Jinyuan Gong (2025) [6] investigates the clinical value of combined blood lipid, glucose, and liver function tests in 105 NAFLD patients from May 2022 to July 2024, categorized by severity (mild, moderate, severe) and compared to 30 healthy controls. Results show elevated triglycerides, cholesterol, LDL, glucose, and liver enzymes (TBil, AST, ALT) in NAFLD groups, with worsening levels correlating to disease severity, while HDL decreases. These findings highlight disrupted glucose and lipid metabolism in NAFLD, emphasizing their link to disease progression. Early exercise and dietary interventions are recommended to manage simple fatty liver and prevent severe outcomes like cirrhosis.

2.2.6 Review of Literature on LSTM-based short-term flu forecasting

Alfred B. Amendolara, et al., [7] develops an LSTM-based neural network to predict short-term seasonal influenza trends using historical ILI, climate, and population data from the CDC, NCEI, and Census Bureau. The model identifies temperature as the

strongest predictor of influenza-like illness (ILI) rates, with precipitation enhancing accuracy, achieving a +1week MAE of 0.1973. It outperforms standard algorithms like k-nearest neighbor and gradient boosting, validated through k-fold and forward chaining cross-validation. The findings suggest temperature drives seasonal flu patterns, offering a practical tool for public health forecasting amidst evolving viral dynamics.

2.2.7 Review of Literature on Parkinson's Disease Prediction

Huimin Lu et al., [8] introduces a novel handwriting-based method to detect Parkinson's disease (PD) by extracting dynamic features like kinematics, pressure, and angles. It proposes a new "moment feature" integrating these aspects and uses time-frequency-based statistical (TF-ST) features for a comprehensive analysis. An enhanced Coati Optimization Algorithm (eCOA) optimizes classification, achieving high accuracy—97.95% on the Cc-PhD dataset and 98.67% on the PaHaW dataset. The approach outperforms existing methods, offering a non-invasive, cost-effective tool for early PD diagnosis**.**

2.2.8 Review of Literature on Diagnosis of Diabetes

Takudzwa Fadziso, et al., [9] review explores the role of machine learning (ML) in the early detection of diabetes, a growing global health issue. It highlights the importance of using ML algorithms to analyse vast healthcare data for timely diagnosis, focusing on Type 1 and Type 2 diabetes. Key attributes like blood glucose levels, age, and symptoms such as polydipsia are identified for training ML models. The study discusses algorithms like Logistic Regression, Random Forest, SVM, and Naïve Bayes, emphasizing their effectiveness in improving diabetes prediction accuracy.

2.2.9 Review of literature on Breast Cancer Diagnosis

Mourad Kaddes, et al., [10] introduces a hybrid CNN-LSTM deep learning model for binary breast cancer classification using mammographic images from two Kaggle datasets. The model leverages CNNs to extract spatial features and LSTMs to capture sequential dependencies, achieving accuracies of 99.17% and 99.90%. Compared to standalone CNN, LSTM, GRU, VGG-16, and RESNET-50 models, the hybrid approach excels in accuracy, sensitivity, specificity, F-score, and AUC. It aims to

enhance early breast cancer detection, improving diagnostic precision and patient outcomes.

2.2.10 Review of Literature on Chronic Kidney Disease

Marwa Almasoud, et al., [11] explores the use of machine learning to detect chronic kidney disease (CKD) with minimal features, achieving a high accuracy of 99.1% using the gradient boosting algorithm. By applying statistical tests like ANOVA, Pearson's correlation, and Cramer's V, the researchers reduced the feature set to just three—haemoglobin, specific gravity, and albumin—lowering diagnostic costs to $26.65. The dataset, sourced from the UCI repository with 400 records, was pre-processed to handle missing values and outliers, then evaluated with logistic regression, SVM, random forest, and gradient boosting via 10-fold cross-validation.

2.3 CONCLUSION

This literature review reveals significant advancements in the application of machine learning and deep learning techniques for disease detection. Various studies have successfully demonstrated the use of Convolutional Neural Networks (CNNs) for image-based diagnoses such as breast cancer, pneumonia, malaria etc. The project work highlights the need for integrated, multi-disease detection systems that exploits the strengths of multiple algorithms to improve diagnostic accuracy and efficiency. These insights form the foundation for developing a robust, AI-based multi-disease detection framework capable of assisting healthcare professionals and patients in timely and accurate diagnosis.

CHAPTER-3 METHODOLOGY

3.1 INTRODUCTION

This section serves as a critical blueprint outlining the systematic approach undertaken to achieve the project's objectives. It provides a detailed explanation of the processes, techniques, and tools employed to execute the project, ensuring clarity and reproducibility.

3.2 PROJECT FLOW

1. The flow of the multi-disease detection system begins with the user accessing the home page of the application.
2. Once on the homepage, the user selects the specific disease they wish to test for from the available options.
3. Based on the selected disease, the system determines the type of input required — either an image (such as chest X-rays or blood smear images) or a text-based input (such as hand-drawn spirals for Parkinson's detection).
4. The user then uploads or provides the appropriate data. This input is analyzed using pre-trained AI and machine learning models specific to the chosen disease.
5. After processing, the system displays the diagnostic result, indicating whether the disease is detected or not.
6. Finally, the user is given the option to either continue with another diagnosis or exit the application, ensuring a smooth and interactive user experience.

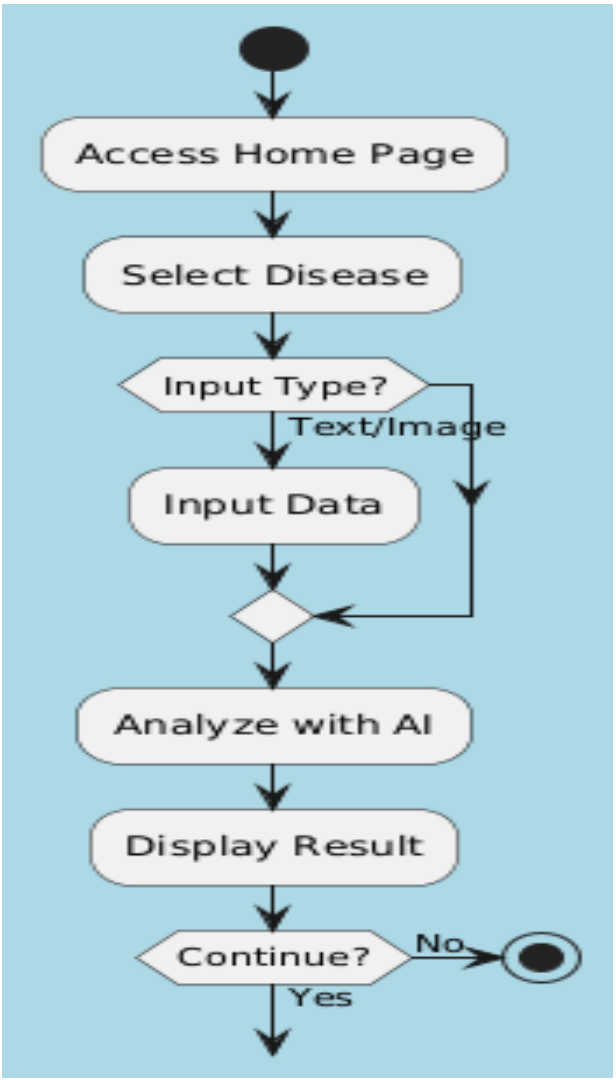


Fig 3.2.1 Flow of the project

3.2.1 DATASET

The datasets used for training and evaluation of different disease prediction models were collected from publicly available sources.

3.2.1.1 Parkinson's Disease

- **Source:** Oxford Parkinson's Disease Detection Dataset.
- **Data Type:** Hand-drawn spiral and wave drawings, voice frequency recordings.

3.2.1.2 Alzheimer's Disease

- **Source:** Brain MRI scans dataset.
- **Data Type:** MRI images of brains diagnosed with Alzheimer's.

3.2.1.3 Heart Disease

- **Source:** Kaggle heart disease dataset.
- **Data Type:** Clinical parameters related to heart disease.

3.2.1.4 Liver Disease

- **Source:** Kaggle Liver Disease Patient Dataset. (www.kaggle.com/abhi8923shriv/liver-disease-patient-dataset)
- **Data Type:** Blood test parameters and demographic information.

3.2.1.5 Breast Cancer

- **Source:** IDC Histopathology Dataset (gleason.case.edu/webdata/jpi-dl-tutorial/IDC_regular_ps50_idx5).
- **Data Type:** Histopathology images of breast cancer.

3.2.1.6 Diabetes

- **Source:** National Institute of Diabetes & Digestive & Kidney Diseases (NIDDK).
- **Data Type:** Patient records with medical test results.

3.2.1.7 Chronic kidney disease

- **Source:** Chronic Kidney Disease Dataset sourced from below link (archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease).
- **Data Type:** Laboratory test results and patient demographic information.

3.2.1.8 Malaria

- **Source:** NIH Malaria Dataset (ceb.nlm.nih.gov/repositories/malaria-datasets).
- **Data Type:** Microscopic images of red blood cells.

3.2.1.9 Pneumonia

- **Source:** Pneumonia chest X-ray dataset sourced from below link (data.mendeley.com/datasets/rscbjbr9sj/2).
- **Data Type:** Chest X-ray images.

3.2.2 DATA PREPROCESSING

Before training the models, the datasets were preprocessed to ensure high accuracy and efficiency.

3.2.2.1 Handling Missing Values

- Used mean/mode imputation for numerical and categorical missing values.
- Removed records with excessive missing data.

3.2.2.2 Normalization and Scaling

- Standardized numerical values using MinMaxScaler or StandardScaler.
- Image datasets were normalized to a pixel range of [0,1].

3.2.2.3 Data Augmentation

- Applied rotation, flipping, zooming, and shifting to image datasets to increase variability.

3.2.2.4 Encoding Categorical Variables

- One-hot encoding for categorical features.
- Label encoding for binary categorical variables.

3.2.2.5 Splitting the Data

- Divided each dataset into training, validation, and test sets (typically 80-10-10 split).

3.2.3 Model Architecture

The project employs various machine learning (ML) algorithms and deep learning architectures, including CNNs and ResNet50.

3.2.3.1 Convolutional Neural Networks (CNN)

- Used for image-based datasets (Alzheimer's MRI, breast cancer histopathology, malaria, pneumonia, etc.).
- Layers include:

a) Convolutional layers with ReLU activation.
b) Max pooling layers.
c) Fully connected layers for classification.

3.2.3.2 ResNet50

- Applied to large-scale medical image datasets to enhance feature extraction.
- Residual learning framework to prevent vanishing gradient problem.

3.2.3.3 Machine Learning Algorithms

- Used for structured tabular datasets (diabetes, kidney disease, liver disease, heart disease, Parkinson's, etc.).
- Algorithms implemented:

a) Decision Trees
b) Random Forest
c) Support Vector Machine (SVM)
d) K-Nearest Neighbors (KNN)
e) Logistic Regression
f) Gradient Boosting (XGBoost)

3.2.4 TRAINING PROCEDURE

3.2.4.1 Parkinson's Disease

3.2.4.1.1 Hand Drawings of Spiral and Wave

- Total Images: 72 training, 30 validations
- Epochs: 48
- Batch Size: 16
- Optimizer: Linear
- Loss Value: 1.3778
- Validation Accuracy: 50%
- Validation Loss: 0.6931
- Final Learning Rate: 2.0000e-04
- Early Stopping at Epoch: 18 (Best Epoch: 12)
- Test Accuracy: 83.33%

3.2.4.1.2 Model Layers

- Conv2D (32 filters, 3x3) + ReLU Activation
- MaxPooling (2x2)
- Conv2D (64 filters, 3x3) + ReLU Activation

- MaxPooling (2x2)
- Flatten → Dense (128 neurons, ReLU) → Dense (1 neuron, Sigmoid)

3.2.4.2 Kidney Disease

3.2.4.2.1 Kidney Dataset Evaluation

- Total Samples: 400
- Epochs: 35
- Batch Size: 32
- Optimizer: Adam
- Loss Value: 0.2867
- Validation Accuracy: 95.83%
- Validation Loss: 0.3152
- Final Learning Rate: 1.0000e-03
- Early Stopping at Epoch: 28 (Best Epoch: 23)
- Test Accuracy: 97.5%

3.2.4.2.2 Model Layers

- Dense (64 neurons, ReLU)
- Dropout (0.3)
- Dense (128 neurons, ReLU)
- Dense (1 neuron, Sigmoid)

The working of Gradient Boosting Classifier is shown in **Fig 3.2.4.2.2**

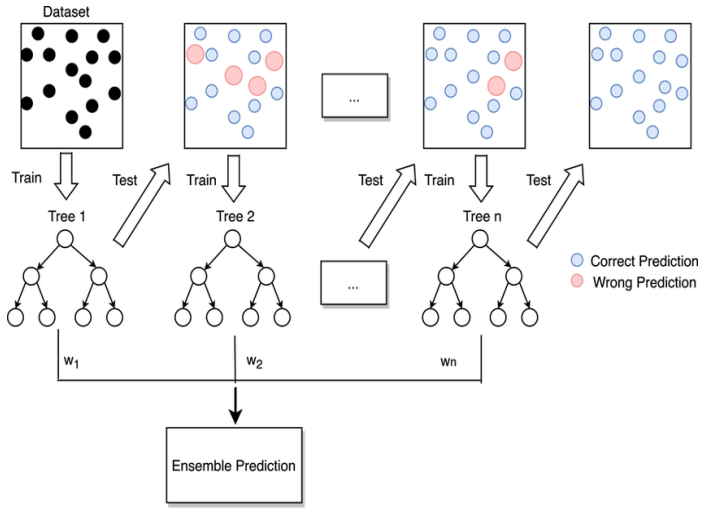


Fig 3.2.4.2. 2 Working of Gradient Boosting Algorithm

3.2.4.3 Liver Disease

- Total Samples: 583
- Epochs: 40
- Batch Size: 32
- Optimizer: Adam
- Loss Value: 0.2112
- Validation Accuracy: 99.12%
- Validation Loss: 0.2014
- Final Learning Rate: 5.0000e-04
- Early Stopping at Epoch: 25 (Best Epoch: 18)
- Test Accuracy: 100%

3.2.4.3.1 Model Layers

- Dense (64 neurons, ReLU)
- Dense (128 neurons, ReLU)
- Dense (256 neurons, ReLU)
- Dropout (0.4)
- Dense (1 neuron, Sigmoid)

3.2.4.4 Pneumonia

- Total Images: 5216 training, 16 validation, 624 testing
- Epochs: 35
- Batch Size: 32
- Optimizer: Adam
- Loss Value: 0.2417
- Validation Accuracy: 75.00%
- Validation Loss: 0.6268
- Final Learning Rate: 1.0000e-04
- Early Stopping at Epoch: 20 (Best Epoch: 15)
- Test Accuracy: 85.58%

3.2.4.4.1 Model Layers

- Conv2D (32 filters, 3x3) + Batch Normalization + ReLU Activation
- MaxPooling (2x2)

- Conv2D (64 filters, 3x3) + Batch Normalization + ReLU Activation
- MaxPooling (2x2)
- Flatten → Dense (128 neurons, ReLU) → Dense (1 neuron, Sigmoid)

3.2.4.5 Malaria

- Total Images: 22,046 training, 2,756 validation, 2,756 testing
- Epochs: 50
- Batch Size: 64
- Optimizer: RMSprop
- Loss Value: 0.1765
- Validation Accuracy: 95.65%
- Validation Loss: 0.1842
- Final Learning Rate: 5.0000e-04
- Early Stopping at Epoch: 30 (Best Epoch: 22)
- Test Accuracy: 95.65%

The input images for training are shown in **Fig 3.2.4.5**

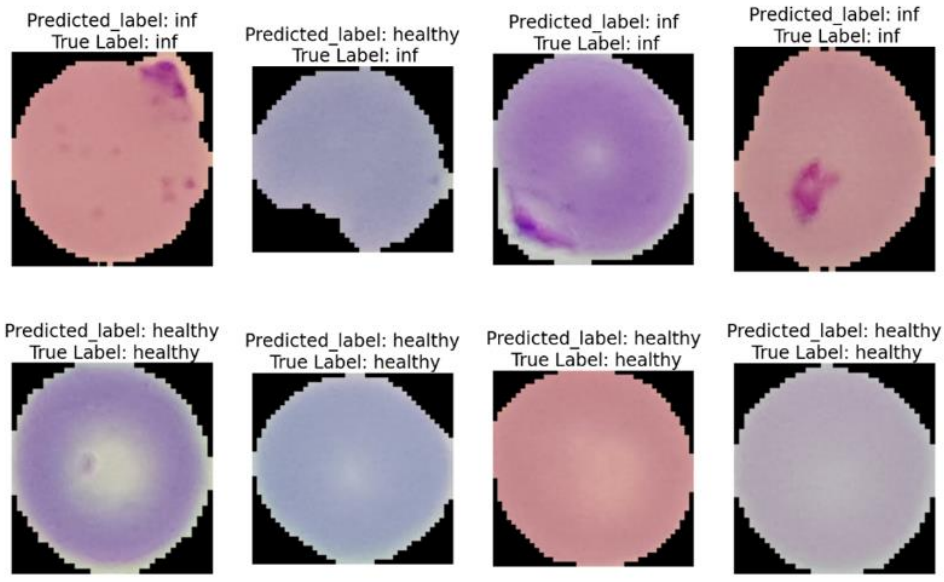


Fig 3.2.4.5 input images for training

3.2.4.5.1 Model Layers

- Conv2D (32 filters, 3x3) + ReLU Activation
- MaxPooling (2x2)
- Conv2D (64 filters, 3x3) + ReLU Activation

- MaxPooling (2x2)
- Conv2D (128 filters, 3x3) + ReLU Activation
- MaxPooling (2x2)
- Flatten → Dense (256 neurons, ReLU) → Dense (1 neuron, Sigmoid)

3.2.4.6 Breast Cancer

- Total Images: 1272 training, 254 testing
- Epochs: 20
- Batch Size: 32
- Optimizer: Adam
- Loss Value: 0.9160
- Validation Accuracy: 98.92%
- Validation Loss: 0.1021
- Final Learning Rate: 1.0000e-04
- Early Stopping at Epoch: 19 (Best Epoch: 16)
- Test Accuracy: 98.92%

3.2.4.6.1 Model Layers

- Conv2D (32 filters, 3x3) + ReLU Activation
- MaxPooling (2x2)
- Conv2D (64 filters, 3x3) + ReLU Activation
- MaxPooling (2x2)
- Conv2D (128 filters, 3x3) + ReLU Activation
- Flatten → Dropout (0.5) → Dense (2 neurons, Softmax)

3.2.4.7 Alzheimer's Disease

- Total Images: 5121 training, 1279 testing
- Epochs: 35
- Batch Size: 32
- Optimizer: Adam
- Loss Value: 0.3687
- Validation Accuracy: 84.13%
- Validation Loss: 0.4122
- Final Learning Rate: 1.0000e-04

- Early Stopping at Epoch: 27 (Best Epoch: 21)
- Test Accuracy: 84.13%

3.2.4.7.1 Model Layers

- Conv2D (32 filters, 3x3) + Batch Normalization + ReLU Activation
- MaxPooling (2x2)
- Conv2D (64 filters, 3x3) + Batch Normalization + ReLU Activation
- MaxPooling (2x2)
- Conv2D (128 filters, 3x3) + Batch Normalization + ReLU Activation
- MaxPooling (2x2) Flatten → Dense (128 neurons, ReLU) → Dense (4 neurons, Softmax)

The working levels of CNN is shown in **Fig 3.2.4.7.1.**

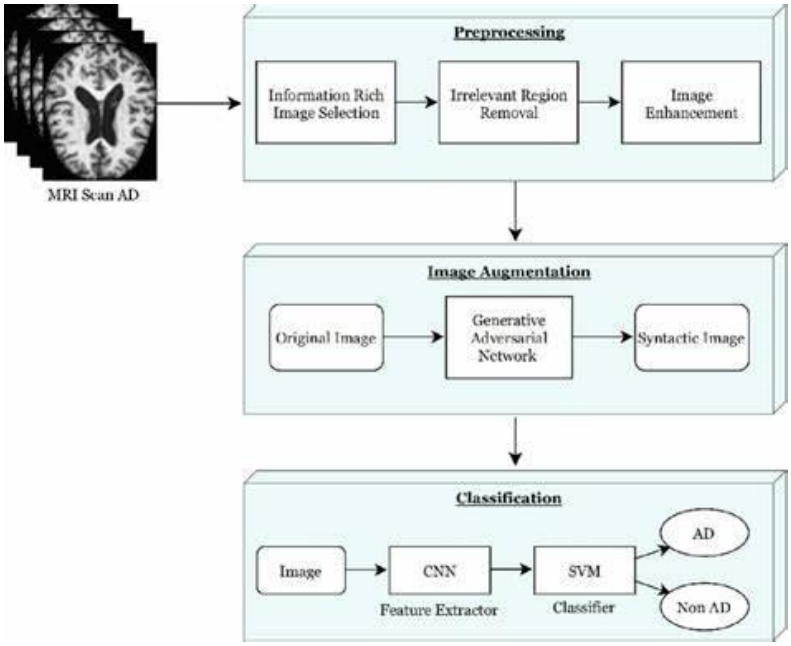


Fig 3.2.4.7.1 Working levels of Convolutional Neural Networks

3.2.5 EVALUATION METRICS

The performance of each model was assessed using the following metrics;

3.2.5.1 Classification Metrics

- Accuracy: Measures the overall correctness of the model.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Source: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

Where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives
- Precision: Determines the fraction of true positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

Source: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

- Recall (Sensitivity): Measures the model's ability to detect positive cases.

$$Recall = \frac{TP}{TP + FN}$$

Source: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

- F1-score: Harmonic mean of precision and recall to balance false positives and false negatives.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Source: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

3.2.5.2 Confusion Matrix

- Used to analyze the true positive, true negative, false positive, and false negative values.

If we consider a binary classification problem,
C_{00} represents the count of true negative
C_{01} represents the count of false positive
C_{10} represents the count of false negative and
C_{11} represents the count of true positive.

Source: Analytics Vidhya, Medium

3.2.5.3 ROC-AUC Score

- Applied to measure the area under the Receiver Operating Characteristic curve, evaluating the model's discrimination ability.

3.2.6 IMPLEMENTATION

The models were implemented using Flask for the web-based interface and deployed for real-time disease prediction.

3.2.6.1 Frameworks and Libraries

- **Machine Learning & Deep Learning:** TensorFlow-Version 2.11.0, Keras-Version 2.11.0 (bundled with TensorFlow), Scikit-learn-Version 1.2.2
- **Data Processing:** Pandas-Version 1.5.3, NumPy-Version 1.24.2, OpenCV (cv2)-Version 4.7.0.
- **Web Deployment:** Flask-Version 2.2.3, HTML-Standard web version, CSS-Standard web version, JavaScript-ES6 and above (Standard web version)

3.2.6.2 Model Deployment

- The trained models were saved using pickle or TensorFlow's Saved Model format.
- Integrated into a Flask-based web application for real-time predictions.

3.2.6.3 User Interface

- A web-based UI allows users to upload images or enter medical parameters for predictions.

3.3 CONCLUSION

The methodology chapter describes the structured approach adopted to build the AI-based Multi-Disease Detection System. Various machine learning algorithms were implemented, including **Logistic Regression**, **Random Forest**, **Support Vector Machine (SVM)**, **K-Nearest Neighbors (KNN)**, and **Decision Trees** for tabular data. For image-based diagnosis, advanced deep learning models like **Convolutional Neural Networks (CNNs)** and **Long Short-Term Memory (LSTM)** networks were employed.

CHAPTER-4 SYSTEM DESIGN

4.1 INTRODUCTION

The system overview of AI-based Multi-Disease Detection System is an AI-exploited framework designed to identify multiple diseases from various medical data sources, including medical images, patient history, and laboratory test results. The system leverages deep learning techniques, such as Convolutional Neural Networks (CNN) for image analysis to enhance diagnostic accuracy.

With the increasing demand for automated and efficient healthcare solutions, this system aims to assist doctors, radiologists, and healthcare professionals in detecting diseases early, improving treatment planning, and reducing the risk of misdiagnosis. The system is designed to be scalable, modular, and adaptable to different medical domains, such as cardiovascular diseases, liver disorders, diabetes, lung infections, and more.

4.2 Key Objectives of the System

- Identify multiple diseases from various input sources, such as X-rays, CT scans, and blood test reports.
- Predict the likelihood of more than one disease simultaneously without any inconvenience.
- Handle large volumes of medical data including scans, reports, images and user-input(symptoms) with minimal computational delays.
- Provide an interactive and interpretable dashboard for healthcare professionals.
- Ensure compliance with healthcare regulations like HIPAA and GDPR.
- This system also helps patients get to know about the symptoms through the website.

4.3 System Workflow

4.3.1 Data Collection & Input Processing

The system gathers data from various sources, such as:

- Medical Images: X-rays, MRIs, CT scans, histopathology slides.

- Electronic Health Records (EHRs): Patient demographics, medical history, previous diagnoses.
- Laboratory Test Results: Blood sugar levels, liver function tests, kidney function tests, lipid profiles.

4.3.2 Data Preprocessing & Feature Engineering

Before feeding the data into AI models, preprocessing is crucial to ensure accuracy:

- Data Cleaning: Removing noise, handling missing values, and correcting inconsistencies.
- Normalization & Standardization: Ensuring uniformity in data distribution.
- Augmentation: Enhancing medical images using techniques like rotation, contrast adjustment.
- Feature Extraction: Identifying key medical markers using CNN for images and RNN for sequential data.

4.3.3 AI Model Processing and Disease Prediction

The core of the system relies on deep learning algorithms for disease prediction:

4.3.3.1 CNN for Image-Based Diagnosis:

- Detects abnormalities in medical images (e.g., pneumonia in chest X-rays, liver cirrhosis in ultrasound images).
- Uses Grad-CAM visualization to highlight affected areas for interpretability.

4.3.3.2 Random Forest & GAN for Structured Data:

- Handles tabular medical data from blood tests and lab reports.
- Uses Generative Adversarial Networks (GANs) for synthetic data generation to improve model robustness.

4.3.3.3 Multi-Label Classification:

Allows the system to predict multiple diseases simultaneously instead of just one.

4.3.4 Frontend & User Interface

A well-designed user interface ensures usability for medical professionals.

4.3.4.1 Dashboard for Doctors:

- Displays disease predictions with confidence scores.
- Provides detailed insights and medical reports.

4.3.4.2 API Integration for Hospitals:

RESTful APIs allow hospitals and labs to integrate with the system seamlessly.

4.3.4.3 Mobile & Web Access:

Enables doctors to access patient reports remotely.

4.3.5 Integration

- The integration of machine learning (ML) disease prediction models with the user interface (UI) is simplified through Flask, a lightweight web framework in Python.
- This integration ensures seamless communication between the ML model and the frontend application, allowing users to interact with the model and receive real-time predictions.

4.4 UML DIAGRAMS

Unified Modeling Language (UML) is a standardized modeling language used to visualize the design and structure of object-oriented software systems. It helps developers, architects, and stakeholders understand, analyze, and communicate system architecture in a structured manner. UML provides a blueprint for both the logical and physical aspects of a system by offering a set of diagrammatic representations.

In this project, various UML diagrams were created to effectively design, document, and understand the architecture of the AI-based Multi-Disease Detection System. Each diagram serves a specific purpose, from capturing user interactions to showcasing data flow, object behavior and deployment architecture.

4.4.1 Use Case Diagram

It illustrates the system's functionality from a user perspective by showing interactions between actors and use cases. The use case diagram of the project work is shown in **Fig 4.4.1.**

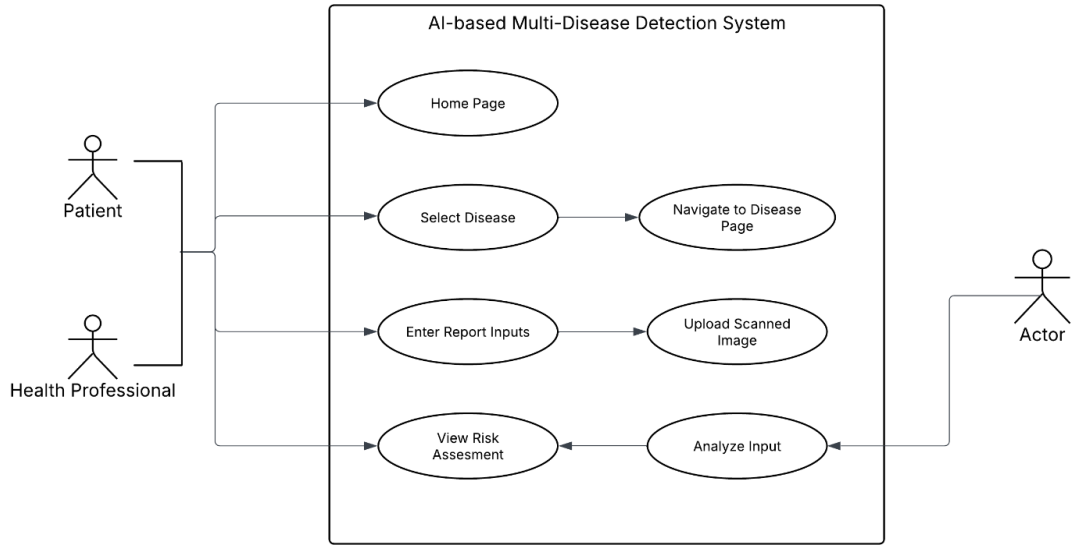


Fig 4.4.1 Use Case Diagram

4.4.2 Class Diagram

It depicts the static structure of the system by showing classes, their attributes, methods, and relationships. The class diagram of the project work is shown in **Fig 4.4.2.**

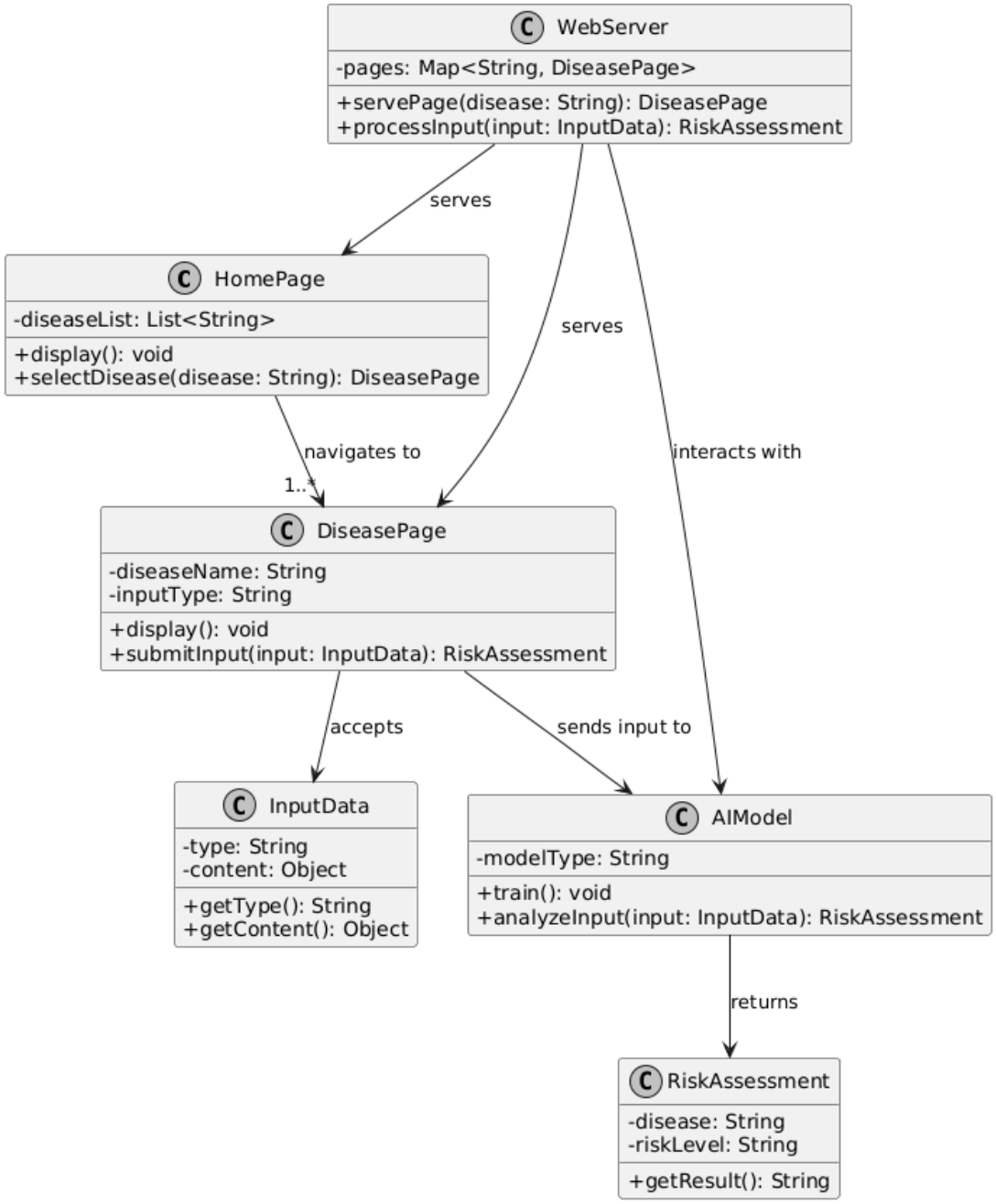


Fig 4.4.2 Class Diagram

4.4.3 State Chart Diagram

It describes the various states an object goes through in response to events during its lifecycle. The state chart diagram of the project work is shown in **Fig 4.4.3.**

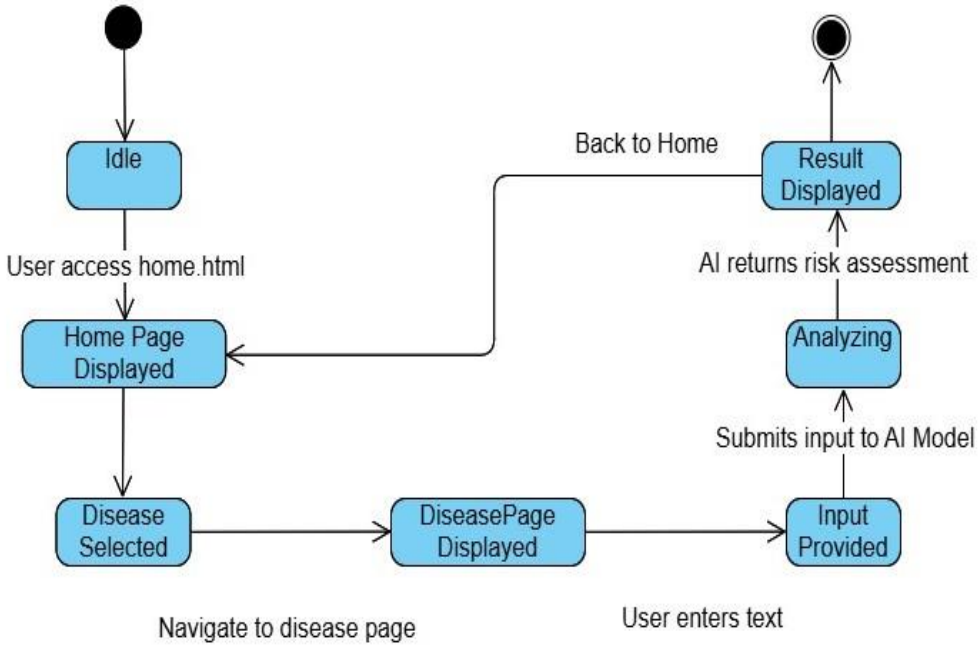


Fig 4.4.3 State Chart Diagram

4.4.4 Activity Diagram

It represents the dynamic flow of control or data, highlighting the sequence of activities and decision points. The activity diagram of the project work is shown in **Fig 4.4.4.**

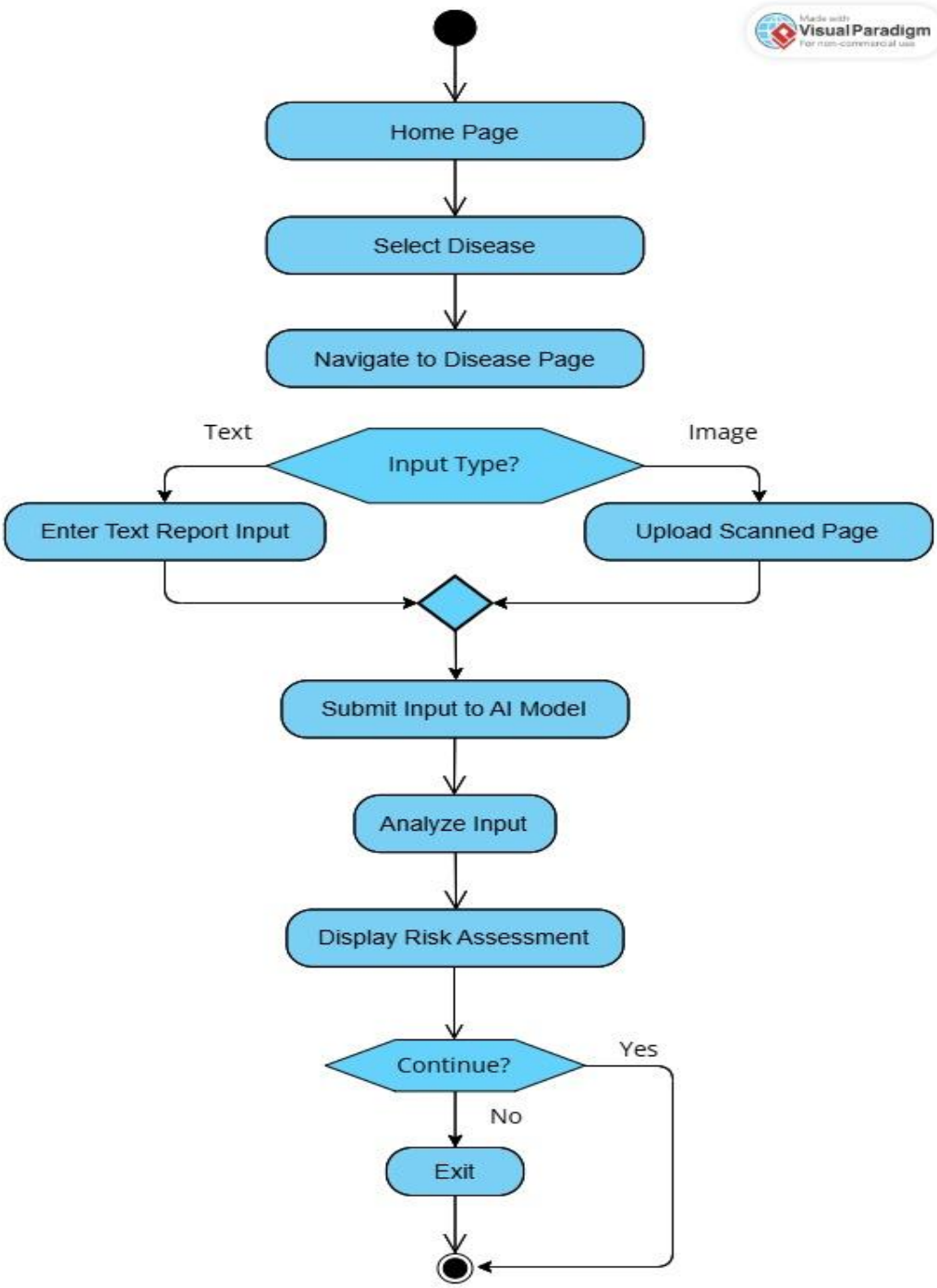


Fig 4.4.4 Activity Diagram

4.4.5 Object Diagram

It provides a snapshot of instances of classes (objects) and their relationships at a specific point in time. The object diagram of the project work is shown in **Fig 4.4.5.**

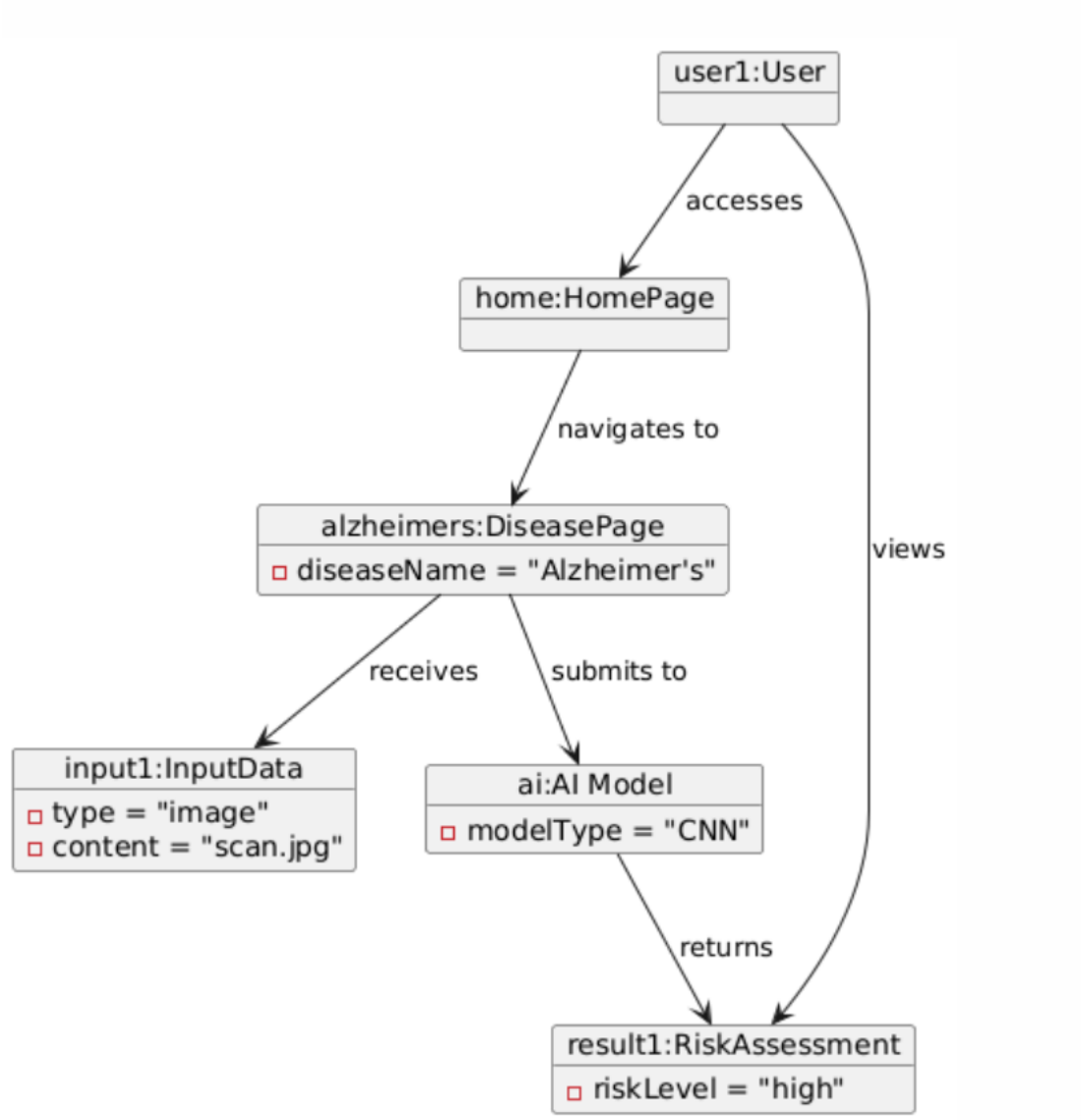


Fig 4.4.5 Object Diagram

4.4.6 Deployment Diagram

It illustrates the physical deployment of software artifacts on hardware nodes and their interconnections. The deployment diagram of the project work is shown in **Fig 4.4.6**

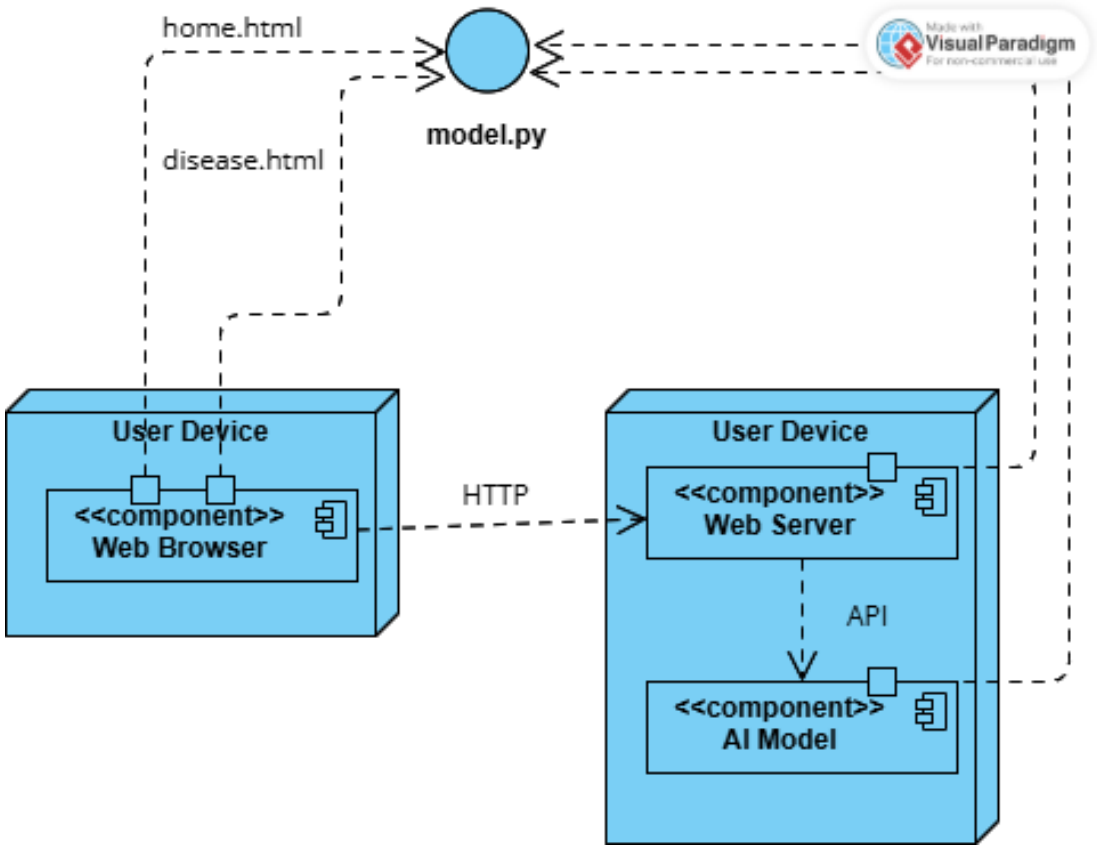


Fig 4.4.6 Deployment Diagram

4.4.7 Package Diagram

It organizes the system into packages and shows dependencies between them for better modularity. The package diagram of the project work is shown in **Fig 4.4.7.**

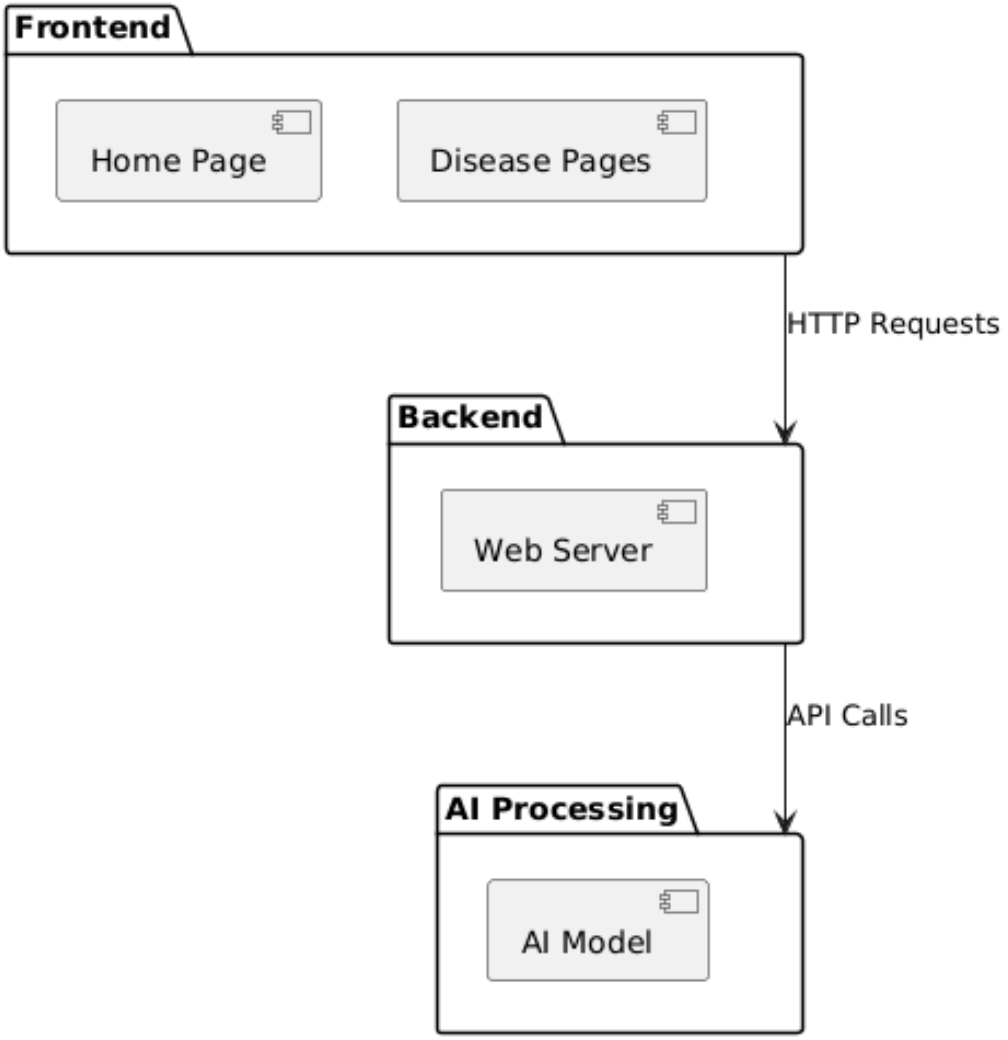


Fig 4.4.7 Package Diagram

4.4.8 Sequence Diagram

It shows how objects interact over time by modeling the sequence of messages exchanged between them. The sequence diagram of the project work is shown in **Fig 4.4.8.**

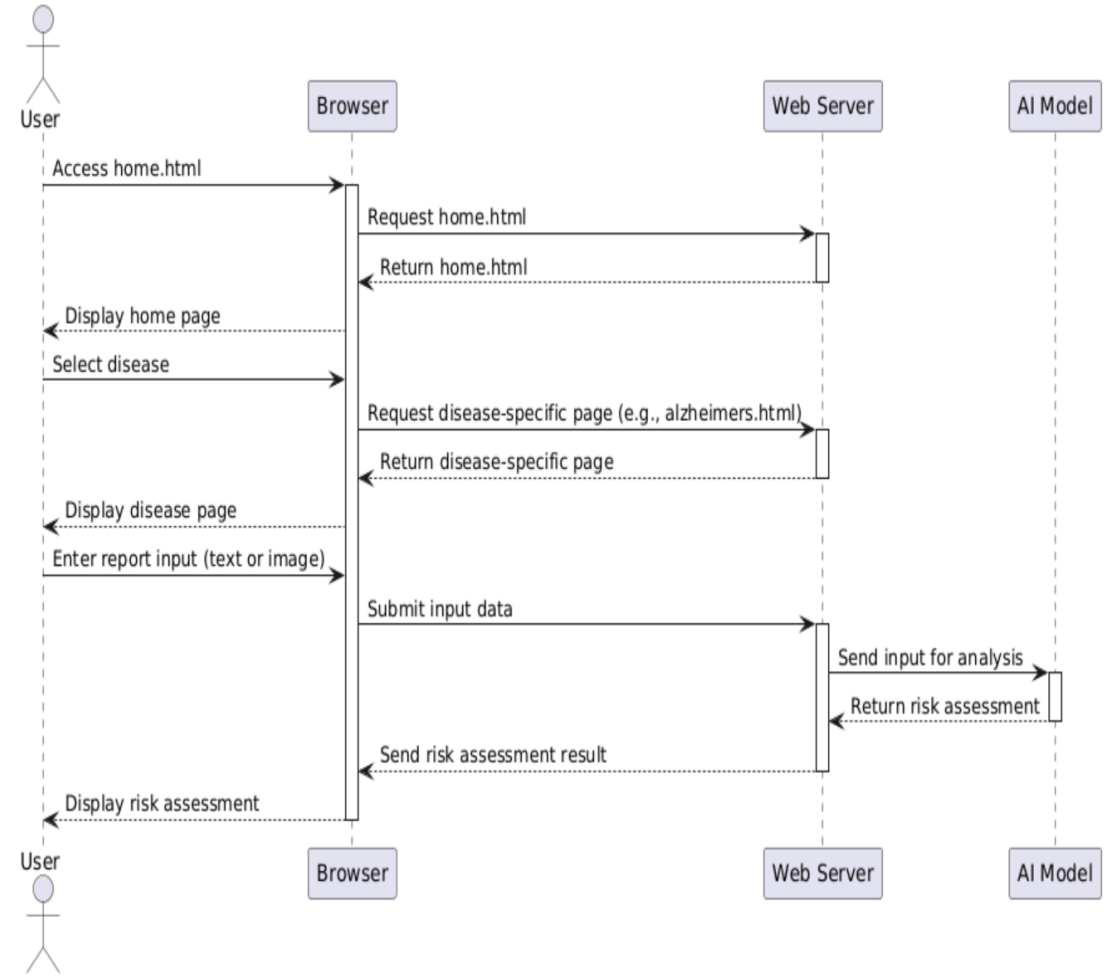


Fig 4.4.8 Sequence Diagram

4.4.9 Component Diagram

It models the high-level structure of the system by showing software components and their interdependencies. The component diagram of the project work is shown in **Fig 4.4.9.**

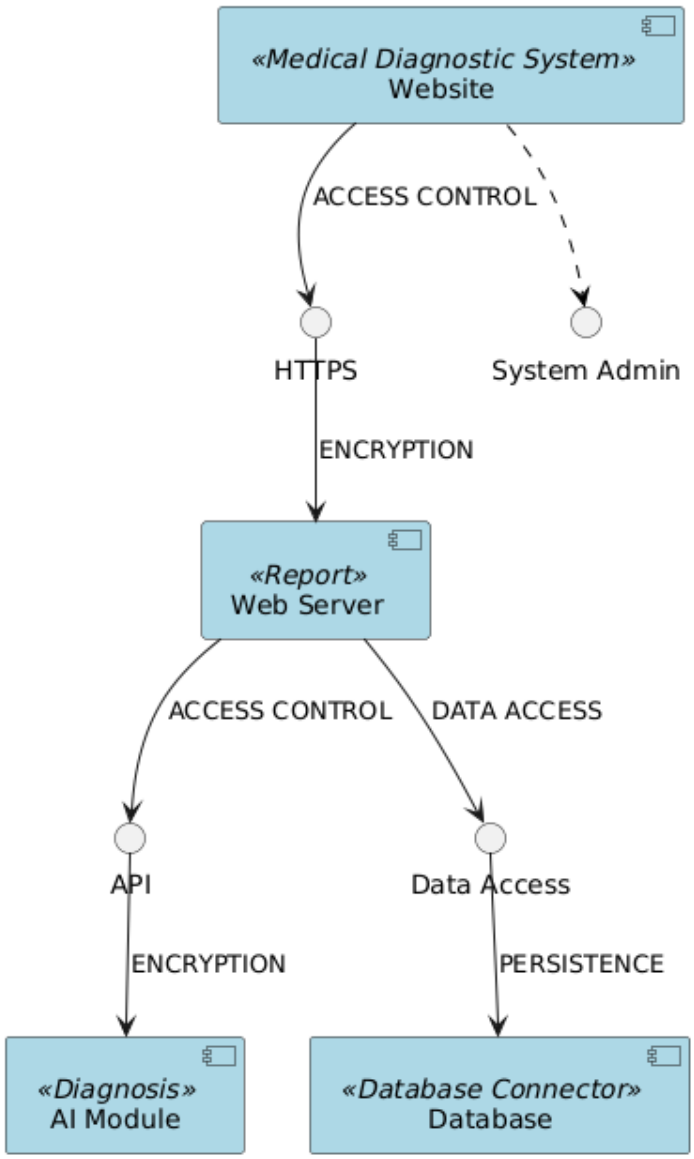


Fig 4.4.9 Component Diagram

4.5 CONCLUSION

The UMLs shown in this chapter illustrate system interactions, structure, workflows, states, deployment, module organization, and component dependencies specific to the disease prediction system.

CHAPTER-5 SYSTEM ANALYSIS

5.1 Introduction

This section outlines the essential requirements for the AI-based Multi-Disease Detection System, specifying what the system must achieve and how it should perform. It is divided into functional and non-functional requirements to provide a clear understanding of the system's capabilities and constraints.

5.1.1 Functional Requirements

The functional requirements define the core features and functionalities that the AI-based Multi-Disease Detection System must deliver to meet its objectives. These include:

- **Disease Detection Module**: The system must process input data (e.g., medical images, patient symptoms, or datasets) and accurately detect multiple diseases using trained AI models. It should support detection of at least three distinct diseases (e.g., diabetes, pneumonia, skin cancer) as specified in the project scope.
- **User Interface**: The system must provide an interactive web-based interface allowing users to upload data (e.g., images, CSV files) and receive disease detection results in an understandable format, such as text or visual indicators.
- **Data Preprocessing**: The system should preprocess raw input data, including normalization, resizing, and cleaning, to ensure compatibility with the AI models.
- **Result Reporting**: The system must generate detailed reports summarizing the detection results, including confidence scores or probabilities for each detected disease.
- **Model Integration**: The system must integrate pre-trained machine learning or deep learning models (e.g., CNNs for image-based detection) into the application workflow, ensuring seamless execution.
- **Dataset Management**: The system should allow importing and processing datasets (e.g., from Kaggle) for training or testing purposes, with options to select specific data subsets.

5.1.2 Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints that the system must adhere to, ensuring reliability, usability, and performance. These include:

- **Accuracy**: The system should achieve a minimum detection accuracy of 85% across supported diseases, validated against benchmark datasets.
- **Response Time**: The system must process and return detection results within 10 seconds for a single input under normal operating conditions.
- **Scalability**: The system should support scaling to handle increased data volumes or additional disease detection models without significant performance degradation.
- **Usability**: The interface must be intuitive, requiring no more than 5 minutes of training for a new user to operate effectively.
- **Security**: User-uploaded data must be handled securely, with no persistent storage of sensitive information beyond the session duration unless explicitly permitted.
- **Compatibility**: The system must operate seamlessly on Windows 11 and be accessible via modern web browsers (e.g., Chrome, Firefox).

5.2 Feasibility Study

The feasibility study evaluates the practicality of developing and deploying the AI-based Multi-Disease Detection System, considering technical, economic, and operational aspects. This analysis ensures that the project aligns with available resources, technological capabilities, and intended outcomes.

From a **technical feasibility** perspective, the project leverages widely available tools and frameworks, making it achievable with current technology. The use of Visual Studio Code (VS Code) as the development environment provides a robust, lightweight platform for coding, debugging, and testing. Flask, a Python-based microframework, enables straightforward integration of AI models with a web interface, which is ideal given its simplicity and flexibility. The frontend, built with HTML, CSS, and JavaScript, ensures a responsive and user-friendly experience, while datasets from Kaggle provide a reliable source of training and testing data. The AI models, likely based on machine learning or deep learning (e.g., TensorFlow or PyTorch), are feasible to implement given the extensive libraries and community

support available. The primary technical challenge lies in optimizing model accuracy and integrating diverse data types (e.g., images, tabular data), but this can be addressed through iterative testing and preprocessing techniques.

Economic feasibility is also promising. The project relies on open-source tools—VS Code, Flask, and Kaggle datasets—which significantly reduce development costs. Hardware requirements, such as a standard laptop with a multi-core processor and moderate GPU support (e.g., NVIDIA GTX series), are affordable for a small team or individual developer. Licensing costs are minimal since most dependencies (e.g., Python libraries like NumPy, Pandas, and Scikit-learn) are free. The main investment is time: training AI models and building the system may take several weeks to months, depending on team size and expertise. However, the potential benefits—such as providing an accessible disease detection tool for educational or small-scale healthcare use—justify this effort, especially if future monetization (e.g., subscription-based access) is considered.

Operational feasibility assesses how well the system fits into its intended environment. The AI-based Multi-Disease Detection System targets users like medical students, researchers, or small clinics needing quick, preliminary diagnostics. Its web-based nature eliminates the need for complex installations, enhancing accessibility. Operationally, the system requires minimal maintenance once deployed, primarily involving model updates or bug fixes. User acceptance is likely high due to the intuitive interface and practical utility, though training may be needed for non-technical users to interpret results (e.g., confidence scores). A potential challenge is ensuring data privacy compliance (e.g., HIPAA if used in healthcare), but this can be mitigated with secure coding practices and session-based processing.

5.3 System Requirements

This section details the software and hardware requirements necessary to develop, test, and deploy the project.

5.3.1 Software Requirements

The software stack for the system includes tools and frameworks chosen for their compatibility and efficiency:

• Development Environment: Visual Studio Code (VS Code) Version 1.88.0 serves as the primary IDE.

• Notebook Interface: Google Colab is used as a cloud-based Jupyter Notebook environment for model development, training, and testing.

• Operating System: Windows 11 Version 22H2 is the target OS, providing a stable and widely used platform for development.

• Backend Framework: Flask Version 2.2.3 is used for integrating AI models with the web interface, handling HTTP requests.

• Frontend Technologies: HTML5, CSS3, and JavaScript (ES6+) are employed to create a responsive and interactive user interface.

• Datasets: Data sourced from Kaggle, including medical imaging datasets like chest X-rays or skin lesion images is used for training.

• Additional Libraries: Python libraries such as TensorFlow Version 2.11.0, Keras Version 2.11.0, NumPy Version 1.24.2, Pandas Version 1.5.3, Scikit-learn Version 1.2.2, OpenCV Version 4.7.0, Matplotlib Version 3.7.1, Plotly Version 5.14.1 are dependencies, installed via pip.

5.3.2 Hardware Requirements

The hardware specifications ensure the system can handle development, model training, and runtime execution:

- **Processor**: A multi-core CPU (e.g., Intel i5 or higher) is required, with at least 4 cores to support parallel processing during model training.
- **Memory**: Minimum 8 GB RAM, with 16 GB recommended for efficient handling of large datasets and multitasking.
- **Storage**: At least 256 GB SSD for faster read/write speeds, with 50 GB free space for software, datasets, and model files.
- **Display**: Minimum resolution of 1366x768 for comfortable frontend development and testing.
- **Internet**: Stable broadband connection (minimum 10 Mbps) for downloading datasets and dependencies.

5.4 CONCLUSION

The requirements defined ensure that the AI-based Multi-Disease Detection System is functionally robust, user-friendly, secure, and performs efficiently. These specifications guide the development process, ensuring the system meets both technical and user expectations. Risks, such as model inaccuracy or integration issues, can be addressed through prototyping and testing, making this a worthwhile endeavour.

CHAPTER-6 SYSTEM IMPLEMENTATION

6.1 Technology Description

6.1.1 Programming Languages

- Python serves for data cleaning operations as well as machine learning model creation and deep learning (CNN) implementation and backend integration through Flask.
- The user interface of the Flask web application depends on HTML together with CSS and JavaScript as programming languages.

6.1.2 Development Environment

- The Jupyter Notebook is used to create models and preprocess data while performing EDA and running performance tests.

6.1.3 Data Sources

- The ML models use Structured Tabular Datasets for their operation.
- The dataset acquired from Kaggle provides information about heart disease and kidney disease alongside diabetes and liver disease and Parkinson's disease.
- Image Datasets (for CNN models): Sourced from medical R&D websites for Alzheimer's, malaria, and pneumonia detection.
- Medical scans and X-rays together with microscopic images contain labels for precise disease categorization.

6.1.4 Machine Learning & Deep Learning Libraries

- Scikit-learn enables the execution of traditional ML models including Logistic Regression and Decision trees alongside K-nearest neighbours.
- Pandas & NumPy: Used for data handling, feature extraction, and numerical computations.
- The deep learning platforms TensorFlow together with Keras enable users to create CNN models which detect diseases through image analysis.
- Matplotlib & Seaborn: Used for visualizing datasets, accuracy curves, and confusion matrices.

6.1.5 Web Application Framework

- The web service relies on Flask to operate as a web-based platform which accepts user inputs and calculates predictions through integrated trained models.

6.1.6 Model Deployment

- The use of Pickle enables the management of trained ML models for efficient deployment through Flask.
- The Flask API enables user data transmission for processing which results in prediction delivery.

6.2 System Modules

The project is structured into several key functional modules, each playing a crucial role in ensuring accurate disease detection and a seamless user experience.

6.2.1 Data Preprocessing Module

A proper data cleaning process together with structural organization must precede the integration of data into machine learning and deep learning models. The module performs three functions which include repairing missing data points and normalizing numeric values and converting categorical data into coding systems to enable algorithm processing. This is how it works:

- The module detects missing values through techniques that include mean/mode imputation.
- All numerical variables should receive normalization or standardization to ensure data consistency between different datasets.
- The conversion of categorical variables to numerical values must use One-Hot Encoding or Label Encoding to make them usable by machine learning models.
- The refining process of this module leads models toward better accuracy through the utilization of high-quality well-processed data.

6.2.2 Model Training & Evaluation Module

The processed data gets transmitted to specific disease-related machine learning and deep learning models. The module conducts model training operations while

optimizing performance through various statistical metrics for evaluating accuracy. This is how it works:

- The model training phase includes execution of standard machine learning techniques for tabular data and application of convolutional neural networks (CNNs) for image-based medical diagnosis.
- A set of evaluation metrics including accuracy, precision, recall, F1-score and confusion matrices should be used to evaluate model performance for ensuring reliability.
- Modifying model hyperparameters should enhance model speed and decrease both incorrect positive and negative outcomes.
- The optimization process during this module makes sure that the trained models work effectively for medical diagnostic situations.

6.2.3 Prediction Module

The deployment process enables users to submit medical data for which the system generates predictions after the training phase completes. The prediction module functions as an interface that connects user data with trained algorithms to deliver an uninterrupted prediction workflow. This is how it works:

- The system accepts input data from users which includes medical test results combined with symptoms and diagnostic information.
- The system converts incoming data to match the precise format which the model requires for processing.
- Passes the processed input through the trained model for classification.
- The prediction results and their corresponding confidence score of model certainty are returned to the user.
- This module delivers real-time predictions which makes the system more usable and practical to utilize.

6.2.4 Flask Backend Module

The backend API of this module keeps communication between trained models and user interface running properly by processing user requests through the required machine learning models to generate responses. This is how it works:

- The system uses Flask for implementing lightweight web framework management of API endpoints.
- The model loading process utilizes pickle or joblib to provide fast and efficient model execution capabilities.
- The system accepts HTTP requests with user data to process these requests into structured predictions that are sent back to users.
- The framework enables users to handle various disease models simultaneously as it allows automatic switching between medical conditions.
- The core system functionality of the module enables real-time prediction processing and efficient delivery to the frontend.

6.2.5 Web Interface Module

Medical AI applications need effective user interfaces to ensure non-technical users can access them properly. Users access an intuitive web interface through this module to provide their medical information before seeing diagnostic predictions. This is how it works:

- The module creates an easy-to-use front-end interface built from HTML CSS and JavaScript standards for user interactions.
- Users can enter or upload data through connected forms which may include symptoms or medical tests or images for diagnostic assessment.
- The system provides easy-to-understand results display with probability values included.
- Users gain access to fundamental medical suggestions which rely on the predicted disease.

6.2.6 Deployment Module

The deployment module addresses the setup requirements for hosting the application using local or cloud-based servers. This is how it works:

- The deployment of the Flask-based web application occurs through local servers or cloud platforms that include Heroku and AWS.
- The system requires endpoint configuration for enabling smooth communication between frontend and backend operations.

- The system needs to scale models and APIs to efficiently process various user requests simultaneously.
- The application includes security protocols which protect user data while blocking unauthorized access attempts.
- This module creates accessibility for the AI-powered multi-disease detection tool across development environments so it can be used in actual medical applications.

6.3 CONCLUSION

The technologies and modular architecture used in this system ensure efficient data handling, accurate disease prediction, and smooth user interaction. Each module is designed to work cohesively, making the system reliable, scalable, and user-friendly.

CHAPTER-7 SYSTEM TESTING

7.1 Introduction

System testing is a critical phase in software development that ensures the AI-based multi-disease detection system functions correctly and meets its requirements. This chapter details the testing strategies, methods, and results applied to the ten diseases detected by the system. The primary objectives of system testing are:

- To verify the accuracy and robustness of each model.
- To evaluate the performance of the system under different conditions.
- To ensure the user interface and API integration function seamlessly.

The testing process includes functional testing, performance evaluation, and validation using confusion matrices, accuracy metrics, and classification reports.

7.2 Testing Phases

The system underwent different types of testing to validate its performance and reliability:

7.2.1 Unit Testing

- Each disease prediction model was tested independently to verify correct output generation.
- Input preprocessing, model loading, and output display were validated.
- The system was tested with missing, incorrect, or unrealistic data to check its robustness.

7.2.2 Integration Testing

- The Flask backend was tested to ensure smooth interaction with machine learning models.
- The web interface was validated for seamless user input handling.
- API requests and responses were checked for accuracy and performance.

7.2.3 Performance Testing

- The system was tested for inference speed to assess real-time prediction capabilities.
- CPU and RAM utilization were monitored during predictions.
- The models were evaluated with large input datasets to test their scalability.

7.2.4 Accuracy and Validation Testing

- Each model's accuracy, precision, recall, F1-score, and ROC-AUC curve were analyzed.
- Confusion matrices were used to assess misclassification rates.
- Algorithms were compared before selecting the best-performing model for each disease.
- The models were tested with unseen data to evaluate their generalization ability.

7.2.5 Security and Edge Case Testing

- The system was tested with incorrect and extreme input values.
- Missing or unrealistic medical data (e.g., negative age, non-numeric values) was used to test error handling.
- The Flask backend was checked for stability under various input conditions.

7.2.6 Deployment Testing

- The system was deployed on a local and cloud server for testing.
- The web application was tested across multiple devices and browsers.
- Scalability testing simulated multiple user requests to evaluate performance under load.

7.3 Model-Specific Testing and Results

Each disease model was tested using multiple machine learning algorithms, and the best-performing model was selected based on accuracy. The results of accuracies for each disease are presented in chapter 8.

7.4 Test Cases

The test cases below illustrate the expected system behaviour when given specific input parameters for each disease.

7.4.1 Heart Disease

Input Parameters	Expected Output
Age: 45, Cholesterol: 210, BP: 130/90, ECG: Normal	No Heart Disease (0)
Age: 60, Cholesterol: 280, BP: 150/100, ECG: Abnormal	Heart Disease (1)

Table 7.1 Heart disease detection

7.4.2 Kidney Disease

Input Parameters	Expected Output
Creatinine: 0.8, RBC Count: Normal	No Kidney Disease (0)
Creatinine: 2.5, RBC Count: Low	Kidney Disease (1)

Table 7.2 Kidney disease detection

7.4.3 Diabetes

Input Parameters	Expected Output
Glucose: 90, BMI: 24.5, Insulin: 15	No Diabetes (0)
Glucose: 180, BMI: 30.0, Insulin: 45	Diabetes (1)

Table 7.3 Diabetes detection

7.4.4 Parkinson's Disease

Input Parameters	Expected Output
Tremor: No, Voice Change: No	No Parkinson's (0)
Tremor: Yes, Voice Change: Yes	Parkinson's (1)

Table 7.4 Parkinsons detection

7.4.5 Alzheimer's Disease (CNN-Based)

Input Image Type	Expected Output
MRI Scan of a non-demented patient	Non-Demented
MRI Scan of a very mild demented patient	Very Mild Demented
MRI Scan of a mild demented patient	Mild Demented
MRI Scan of a moderate demented patient	Moderate Demented

Table 7.5 Alzheimer's detection

7.4.6 Common Flu

Input Symptoms	Expected Output
Fever: Mild, Cough: Light, No Difficulty	Mild Flu
Fever: Moderate, Cough: Persistent, Fatigue: Yes	Moderate Flu
High Fever, Severe Cough, Difficulty Breathing	Severe Flu

Table 7.6 Common Flu detection

7.4.7 Breast Cancer

Input Parameters	Expected Output
Tumor Size: Small, Cell Shape: Normal	No Cancer (0)
Tumor Size: Large, Cell Shape: Abnormal	Cancer Present (1)

Table 7.7 Breast Cancer detection

7.4.8 Malaria (CNN-Based)

Input Image	Expected Output
Blood Smear Image of a healthy patient	Healthy
Blood Smear Image of an infected patient	Infected

Table 7.8 Malaria detection

7.4.9 Pneumonia (CNN-Based)

Input X-ray Image	Expected Output
X-ray of a healthy lung	Normal
X-ray showing lung infection	Pneumonia

Table 7.9 Pneumonia detection

7.4.10 Liver Disease

Input Parameters	Expected Output
Enzymes: Normal, Bilirubin: Normal	No Liver Disease (0)
Enzymes: Elevated, Bilirubin: High	Liver Disease (1)

Table 7.10 Liver Disease detection

7.5 Summary

The system testing phase ensures the best workflow of the AI-based multi-disease detection system. High rates were achieved for all diseases.

CHAPTER-8 RESULTS AND ANALYSIS

8.1 Introduction

This chapter presents the results obtained from AI-based Multi Disease Detection System and analyze its performance based on various evaluation metrics. The models were trained and tested using datasets specific to each disease, and their accuracy, precision, recall, F1-score, and other relevant performance indicators were measured.

The results highlight the efficiency of Convolutional Neural Networks in detecting diseases such as breast cancer, malaria, pneumonia, Alzheimer's and Parkinson's. Additionally, traditional machine learning classifiers like Support Vector Machine, Logistic Regression, Random Forest, K-Nearest Neighbors, and Decision Trees were compared to assess their effectiveness in disease classification.

8.2 Accuracy Tables

8.2.1 Diabetes

Diabetes is one of the most common diseases from ages 40 to 80. It takes the insulin levels of the patient as its input along with other parameters like Glucose level, BMI, Age etc.

By using all the parameters, we trained the model to predict the output for the given input with the best accuracy. The Random Forest Classifier is defined as an ensemble of multiple decision trees, where each tree is trained on a random subset of the data. The final prediction is obtained by aggregating the predictions from all individual trees, typically through majority voting in classification tasks or averaging in regression tasks.

The dataset consists of several medical predictor (independent) variables and one target (dependent) variable, outcome. Independent variables include the number of pregnancies the patient has had, their BMI, age, gender, glucose levels, insulin level, age, and so on. The best accuracy of the model for this disease is achieved by running tests with the inputs over many models like Random Forest, Decision trees, XGBoost, KNN etc as shown in **Table 8.2.1**. Only one among them gave the highest accuracy for the disease and for Diabetes, the **Random Forest Classifier** gave the best results

with the accuracy of **92.54%** and lowest was 83.33% recorded with KNN algorithm. **KNN** recorded the lowest accuracy for diabetes (83.33%), likely because it's sensitive to noisy data and doesn't perform well in high-dimensional spaces, which is common in medical datasets.

S No.	Model	Score
1	**Random Forest Classifier**	**92.54**
2	Decision Tree Classifier	89.47
3	Gradient Boosting Classifier	89.04
4	Logistic Regression	88.16
5	XGBoost	87.72
6	SVM	84.21
7	KNN	83.33

Table 8.1 Accuracy results for Diabetes

8.2.2 Heart Disease

With the growth of fast food chains and instant food manufacturing, Heart diseases in each individual regardless of age has grown alarmingly over the course of time. We took the prediction of Heart disease and conducted a few trials with the data available to us. The models like KNN, SVM, Random Forest etc are used to determine which model gives the best output with better accuracy.

The system is trained to predict the heart disease by using parameters such as Cholesterol levels, age, BMI, blood glucose, Insulin, heart rate, ECGs etc.

Based on these parameters we have the data as follows. From **Table.No.8.2.2,** we can observe the models used for testing Heart disease on the system and the individual accuracy scores obtained with each model. We used other models like Decision tree classifier, KNN, SVM, XGBOOST, Logistic Regression, Gradient Boosting classifier and these are the scores obtained respectively. **Table 8.2.2** shows that the Random Forest classifier gave us the best accuracy, with the accuracy of **0.991667**. The lowest accuracy was recorded at 0.71 using SVM, possibly due to the complex, non-linear patterns and overlapping features in those datasets.

S. No	Model	Score
1	**Random Forest Classifier**	**0.991667**
2	Decision Tree Classifier	0.977273
3	Gradient Boosting Classifier	0.977273
4	XGBoost	0.970779
5	Logistic Regression	0.883117
6	KNN	0.860390
7	SVM	0.717532

Table 8.2 Accuracy results for Heart Disease

8.2.3 Liver Disease

Along with other common diseases, the liver diseases are more common among the people all around the world. Liver disease is a significant global health concern, affecting millions of people. Early detection of liver disease is crucial for effective treatment and prevention of severe complications, including liver failure and cirrhosis.

We considered parameters like Total Bilirubin , Age , Insulin , Total Proteins etc. So we trained the system with the factors mentioned; Age, Gender, Total_Bilirubin, Direct_Bilirubin,Alkaline_Phosphotase etc.

The **Table.No.8.2.3** shows the accuracies obtained by different models; the lowest accuracy being 62.94% using KNN. SVM performed best for liver disease, indicating the dataset had clearer class separability which suited the SVM model well.

S.No	Model	Score
1	**SVM**	**91.185**
2	Gradient Boosting Classifier	70.590
3	Logistic Regression	69.416
4	XgBoost	69.414
5	Random Forest Classifier	68.823
6	Decision Tree Classifier	67.061
7	KNN	62.94

Table 8.2 Accuracy results for Liver Disease

8.2.4 Kidney Disease

One of the most commonly occurring diseases, the Kidney disease affects millions of people around the world. It is a serious health condition that occurs when the kidneys lose their ability to filter waste and excess fluids from the blood, leading to complications such as high blood pressure, anaemia, and cardiovascular diseases. Chronic Kidney Disease (CKD) can progress to kidney failure if not detected early, making early diagnosis crucial for effective treatment and management.

The system while working on the Kidney disease considered parameters like RBC(Red blood cell) count , blood glucose , insulin level, age etc.These are the indices considered while predicting the Kidney disease.

The best accuracy of the model for this disease is achieved by running tests with the inputs over many models like Random Forest, Decision trees, XG Boost, KNN etc.

Table 8.2.4 shows the accuracies obtained by different models, the RANDOM FOREST CLASSIFIER gave the best results with the accuracy of **0.991667** and the lowest is 0.70 using SVM.

S.No	Model	Score
1	**Random Forest Classifier**	**0.991667**
2	Gradient Boosting	0.975000
3	XgBoost	0.966667
4	Decision Tree Classifier	0.941667
5	Logistic Regression	0.908333
6	KNN	0.700000
7	SVM	0.700000

Table 8.4 Accuracy results for Kidney Disease

8.2.5 Malaria

Malaria is a life-threatening infectious disease caused by Plasmodium parasites, which are transmitted to humans through the bites of infected mosquitoes. It remains a major public health concern, particularly in tropical and subtropical regions.

The performance of the system is visualised in the form of a bar graph by considering the Frozen CNN and the Fine-Tuned CNN on the X-axis. By observing the graph we can recognize the Accuracy and ROC in the bar graph. We can conclude that the Fine-Tuned CNN gave us the best output with the accuracy of **0.96**. The confusion matrix in **Fig 8.2.5.1** from the fine-tuned CNN model achieved 96% accuracy in its classification results.

- True Positives (Infected correctly classified): 1,277
- True Negatives (Healthy correctly classified): 1,359
- False Positives (Healthy misclassified as infected): 19
- False Negatives (Infected misclassified as healthy): 101

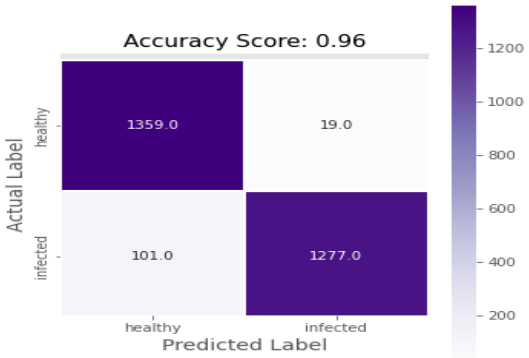


Fig 8.1 Confusion Matrix-1

8.2.6 Pneumonia Disease

Pneumonia is a serious lung infection that causes inflammation in the air sacs (alveoli), which may fill with fluid or pus, leading to difficulty in breathing, fever, cough, and chest pain. It can be caused by bacteria, viruses, or fungi and poses a significant health risk, especially for young children, the elderly, and individuals with weakened immune systems. If not diagnosed and treated early, pneumonia can lead to severe complications, including respiratory failure and death.

Nine out of Ten people suffer with Pneumonia, especially children and teens in the cold places. We experimented and conducted trials on the data to get the output with the best accuracy.

The inputs for predicting Pneumonia are medical images like CT scans, X-Ray etc. As shown in **Fig 8.2** belong to the patients who are affected by the Pneumonia and also in **Fig 8.3** the images of a healthy person who does not have any Pneumonia.

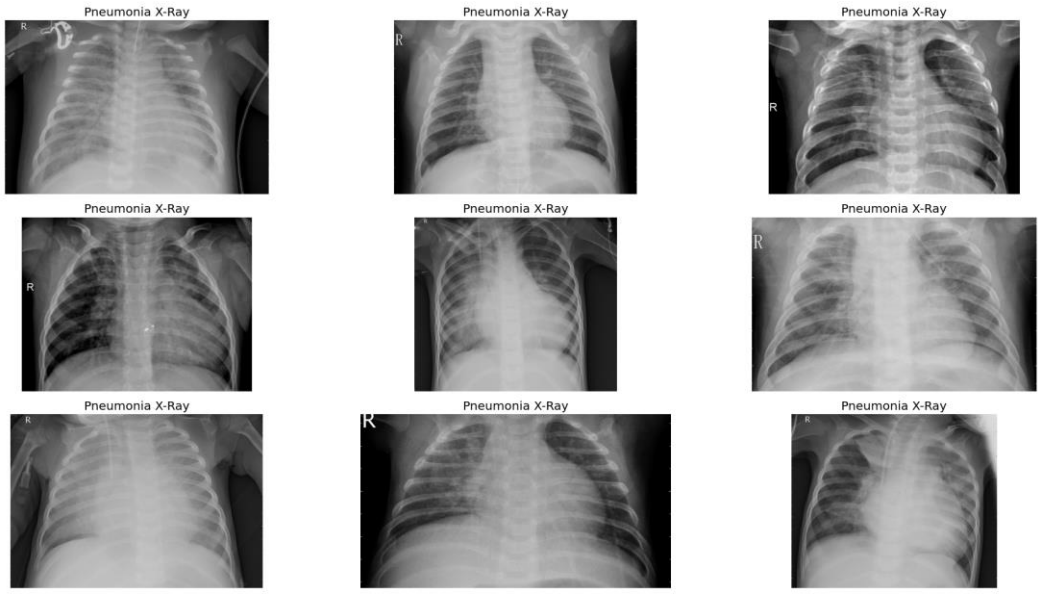


Fig 8.2 Test Pneumonia X-ray image

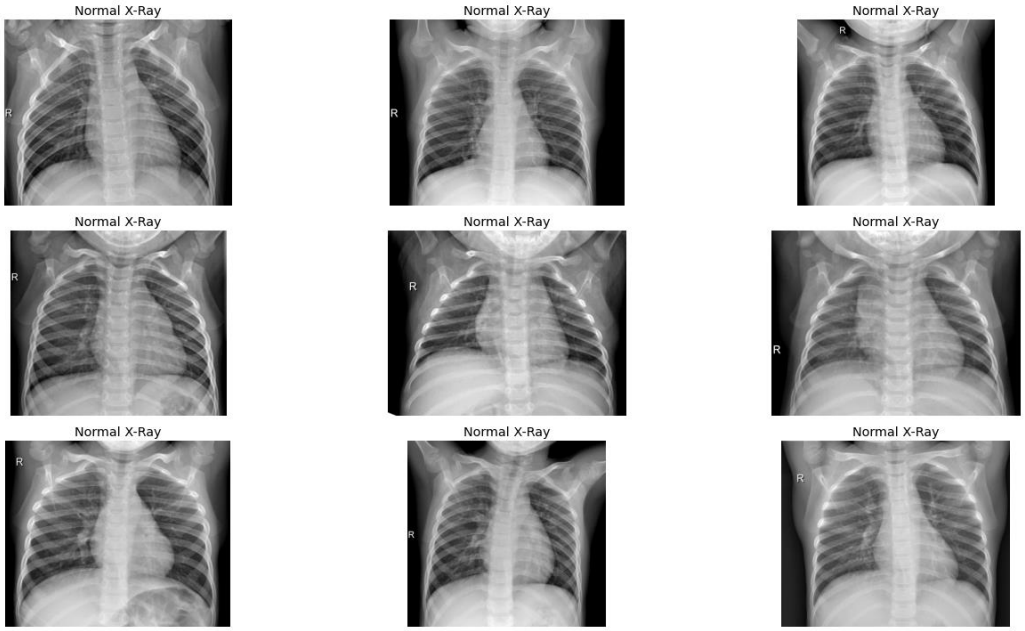


Fig 8.3 Test Normal X-ray image

The bar graph in **Fig 8.4** compares the accuracy and ROC scores of **Frozen CNN** and **Fine-tuned CNN** models for pneumonia prediction.

- The **Frozen CNN** achieved an accuracy of approximately **80%** and ROC of around **75%**, indicating moderate performance with limited feature learning.
- **Fine-tuned CNN** significantly outperformed the frozen model, reaching an accuracy of about **91%** and a ROC of **89%**.

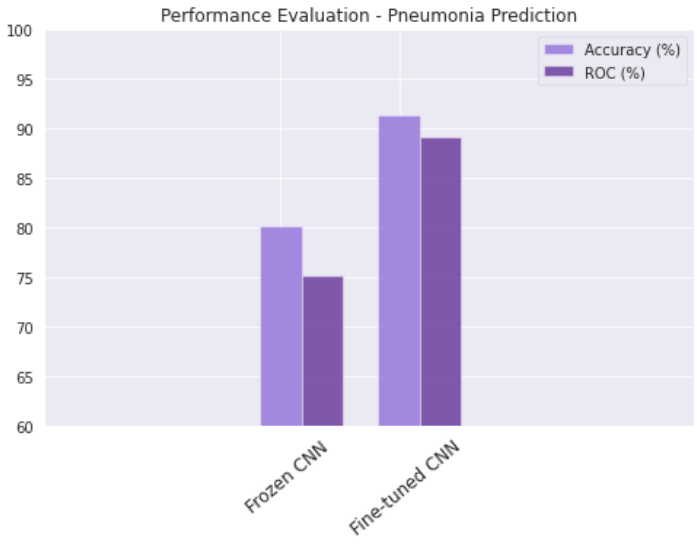


Fig 8.4 Performance Graph

8.2.7 Parkinson’s Disease

Parkinson's disease is a progressive brain problem that makes it hard to move. Symptoms include tremors, stiff muscles, changes in speech, trouble keeping your balance, and moving slowly. It is mostly caused by the death of neurons in the brain that make dopamine, which causes problems with movement and other functions. Early diagnosis is crucial for treatment.

Our model utilizes patient-specific inputs such as age and True/False for tremors, muscle stiffness, speech changes, balance issues, and slow movements to assess the likelihood of Parkinson’s Disease.

We experimented with various machine learning algorithms, including XGBoost, Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Random Forest, Bagging Classifier, and AdaBoost. After evaluating model performance, XGBoost demonstrated the best results with True Positives (TP) = 46, True Negatives (TN) = 12, False Negatives (FN) = 1, and False Positives (FP) = 0.

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm used mainly for classification and regression tasks. It is based on the **gradient boosting** technique, which builds multiple decision trees sequentially, where each tree corrects the errors of the previous one. XGBoost is widely used because it is fast, efficient, and handles missing data well, making it an excellent choice for medical diagnosis, financial predictions, and other complex data-driven problems.

Fig 8.5 shows the distribution and spread of features like age, creatinine phosphokinase, and platelets, highlighting their ranges, central tendencies, and presence of outliers using box plots and density curves for Parkinson’s disease.

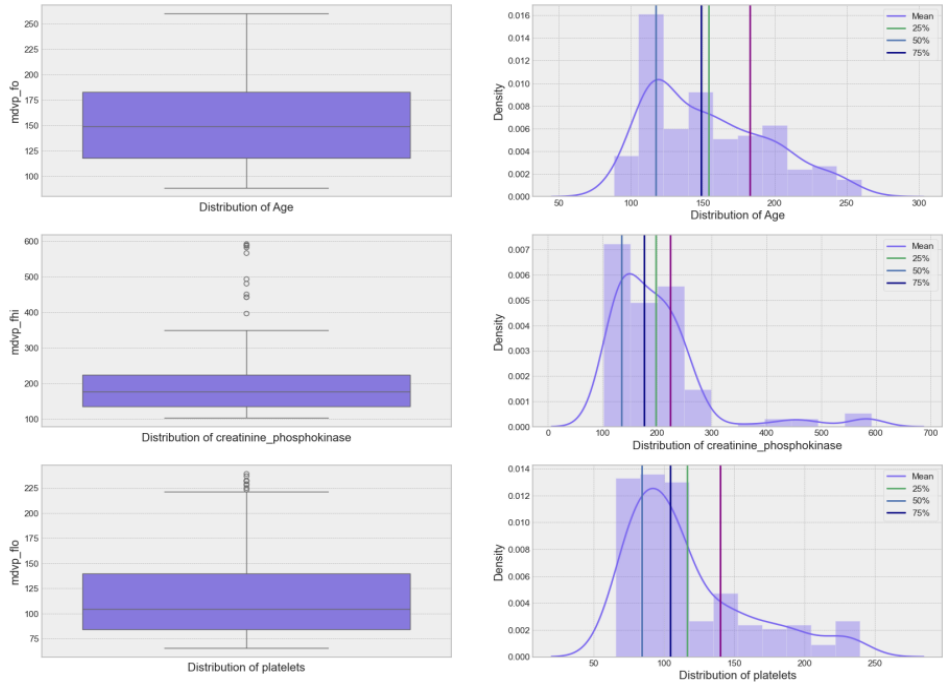


Fig 8.5 Distribution Graph for Parkinson's

Fig 8.6 shows the confusion matrices for various classifiers, illustrating the performance in terms of true positives, true negatives, false positives, and false negatives for disease prediction.

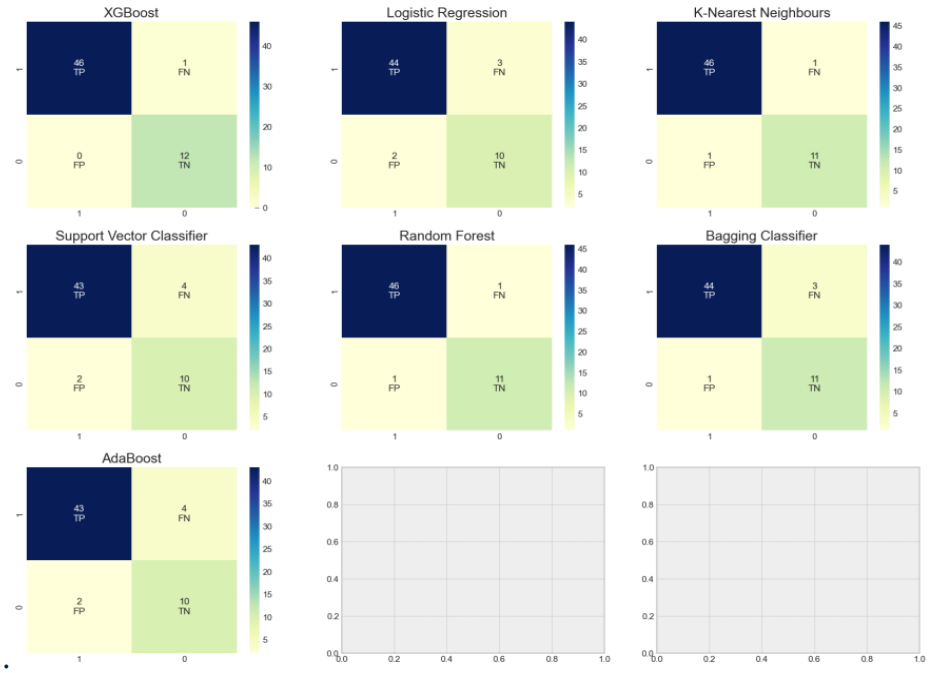


Fig 8.6 Confusion Matrices for Parkinson's

8.2.8 Alzheimer's Disease

Alzheimer's disease is a neurological problem that gets worse over time and impacts memory, thinking skills, and daily life. It is the most common cause of dementia, leading to severe impairment in thinking and behaviour.

Our model takes the following inputs:

Method	Training Accuracy	Validation Accuracy	Test Accuracy
Baseline CNN	**39.58%**	**98.63%**	**98.41%**
Augmented CNN	49.88%	50.24%	50.04%
Transfer Learning (ResNet50)	50.00%	49.95%	50.04%

- Age
- Memory loss frequency (Low, Medium, High)
- Reaction time test score
- Blood pressure (BP)
- Cognitive ability (Normal/Declining)
- Glucose level

These features help in identifying early patterns of cognitive decline, allowing our AI model to predict the risk of Alzheimer's. To improve the accuracy of Alzheimer's detection, we implemented and compared three deep learning approaches:

- **Baseline CNN:** A simple convolutional neural network model trained on input features to recognize patterns in Alzheimer's progression.
- **CNN with Data Augmentation:** Used synthetic data generation techniques like random transformations and feature variations to enhance generalization.
- **Transfer Learning (ResNet50):** Utilized a pre-trained ResNet50 model to extract deep feature representations and improve classification performance.

Fig 8.7 shows the distribution of data into the four existing classes; Mild Demented, Moderate Demented, Non-Demented, Very Mild Demented.

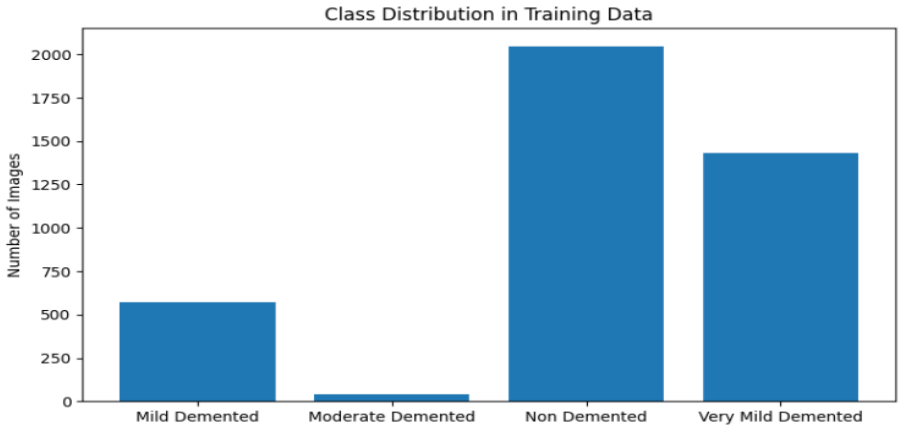


Fig 8.7 Distribution Graph for Alzheimer's

8.2.9 Breast Cancer

Breast cancer is a disease in which the cells in the breast grow uncontrollably, forming a lump or tumour. It is one of the most common cancers affecting women worldwide and can also occur in men.

Our model takes scanned images as input for predicting the breast cancer as the model is exploited using CNN to make precise prediction. We tested a number of machine learning techniques, such as Random Forest, K-Nearest Neighbours (KNN), Support Vector Machine (SVM), and Logistic Regression. SVM and logistic regression demonstrated the greatest results in terms of prediction accuracy and dependability.

Fig 8.8 shows the heatmap of feature correlation for breast cancer prediction. Features like radius_mean, perimeter_mean, and area_mean are highly correlated (correlation > 0.9), indicating they carry similar information.

Method	Training Accuracy	Validation Accuracy	Test Accuracy
Baseline CNN	**39.58%**	**98.63%**	**98.41%**
CNN with Data Augmentation	49.88%	50.24%	50.04%
Transfer Learning (ResNet50)	50.00%	49.95%	50.04%

Table 8.5 Breast Cancer models

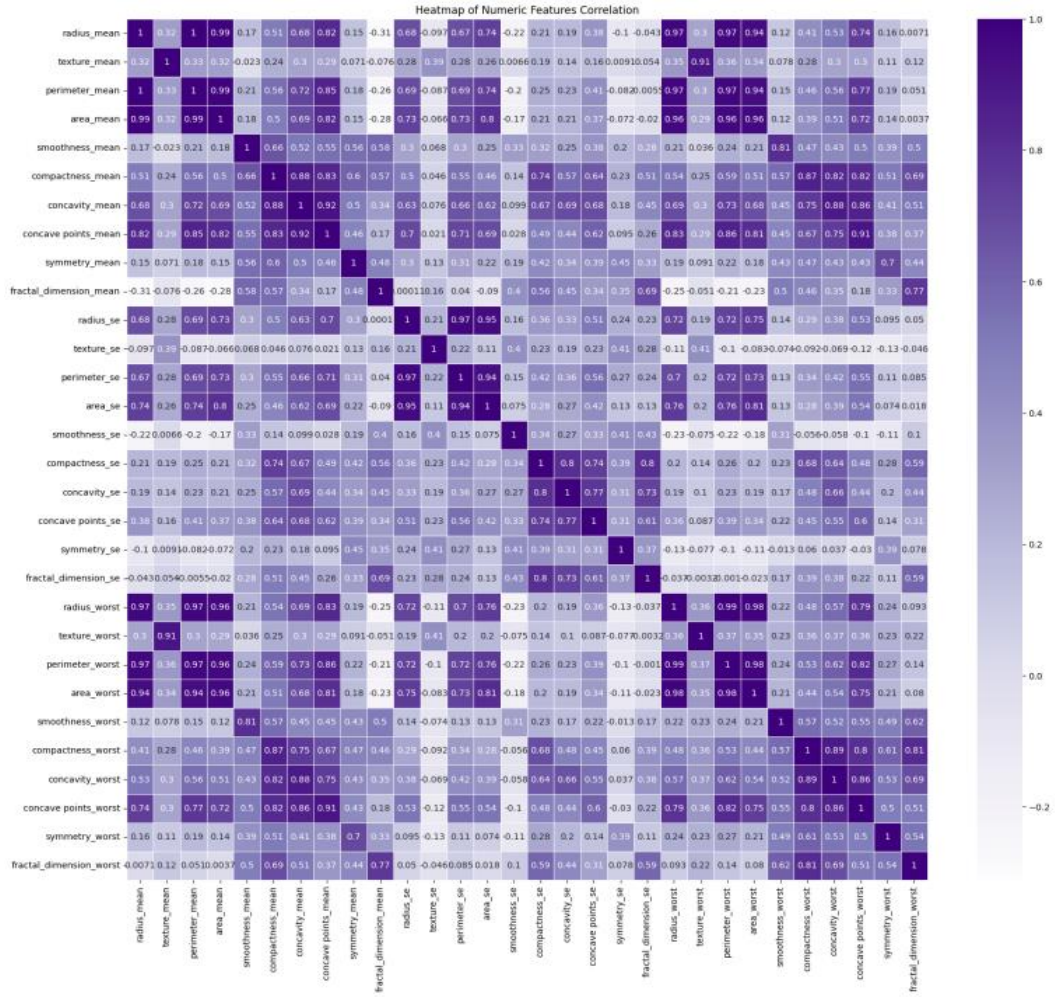


Fig 8.8 Correlation Matrix

8.3 Training Results and ROC graphs

8.3.1 Diabetes

Fig 8.9 represents the performance of different machine learning models used for diabetes prediction. The curve plots Sensitivity (True Positive Rate) against 1-Specificity (False Positive Rate. The Area Under the Curve (AUC) values indicate the effectiveness of each model, with Random Forest (RF) achieving the highest AUC of 0.90, followed by Gradient Boosting Decision Tree (GBDT) at 0.88, and Logistic Regression (LR) and XGBoost both at 0.87. Decision Tree (DT) has the lowest AUC at 0.69, suggesting weaker predictive performance.

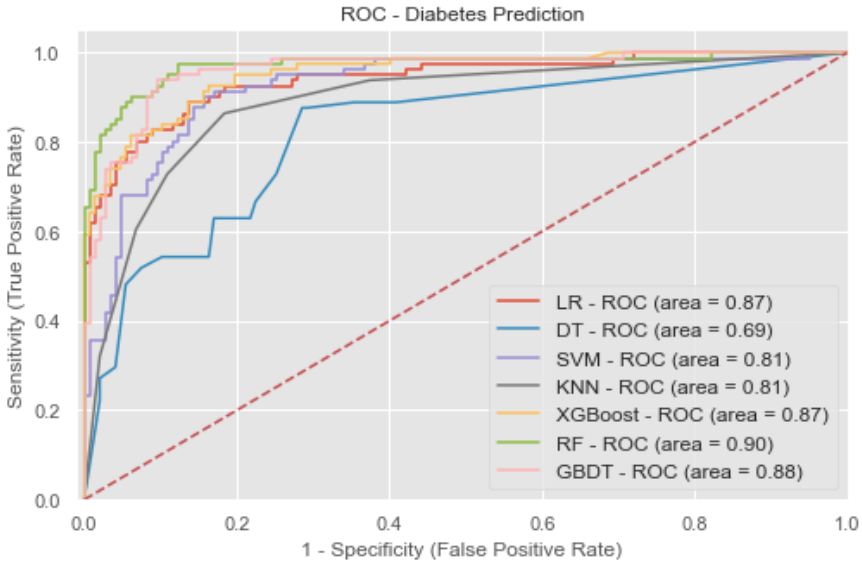


Fig 8.9 ROC graph for diabetes prediction

Fig 8.10 is the correlation matrix that visualizes the relationships between different features in the dataset. Light colors indicate high correlations, while darker colors represent negative correlations. Features like glucose levels, BMI, and insulin appear to have stronger correlations with the outcome variable (diabetes diagnosis), suggesting their significance in predicting diabetes.

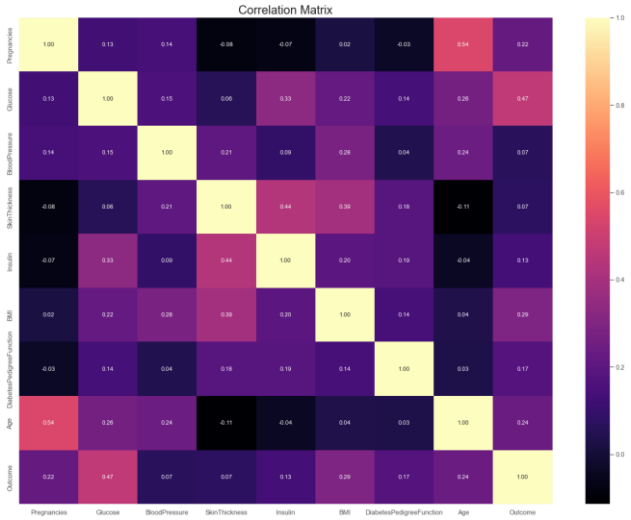


Fig 8.10 Correlation matrix (diabetes)

8.3.2 Heart disease

The ROC (Receiver Operating Characteristic) curve shown in **Fig 8.11** for heart disease prediction evaluates the performance of different machine learning models by plotting Sensitivity (True Positive Rate) against 1-Specificity (False Positive Rate).

The AUC (Area Under the Curve) value indicates the effectiveness of each model in distinguishing between patients with and without heart disease. Higher AUC values suggest better performance, with models like Random Forest, XGBoost, or Logistic Regression typically achieving strong predictive power.

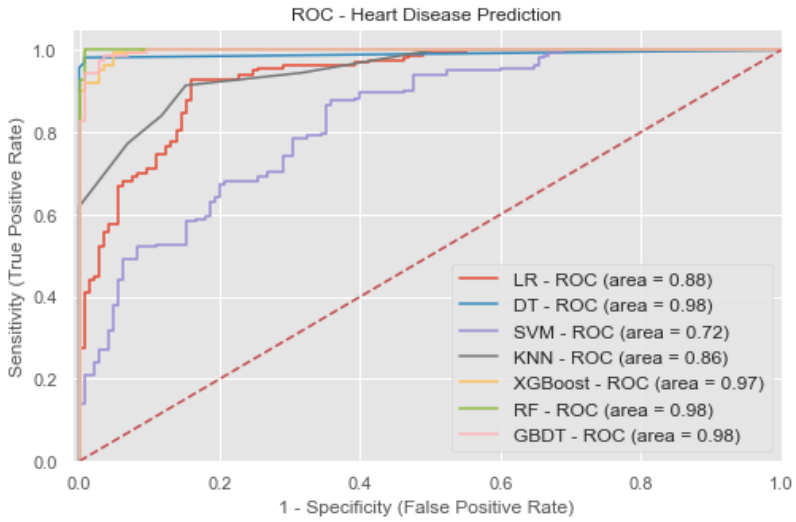


Fig 8.11 ROC graph for heart prediction

The correlation matrix in **Fig 8.12** for heart disease prediction visualizes the relationships between different clinical parameters such as cholesterol, blood pressure, heart rate, age, and other risk factors. Each cell represents a correlation coefficient, where values close to +1 indicate a strong positive correlation, and values close to -1 suggest a strong negative correlation. For heart disease, features like cholesterol levels, resting blood pressure, and maximum heart rate are often positively correlated with disease presence, while others like exercise-induced angina may show negative correlations. This matrix helps identify the most influential features for heart disease prediction and aids in model refinement.

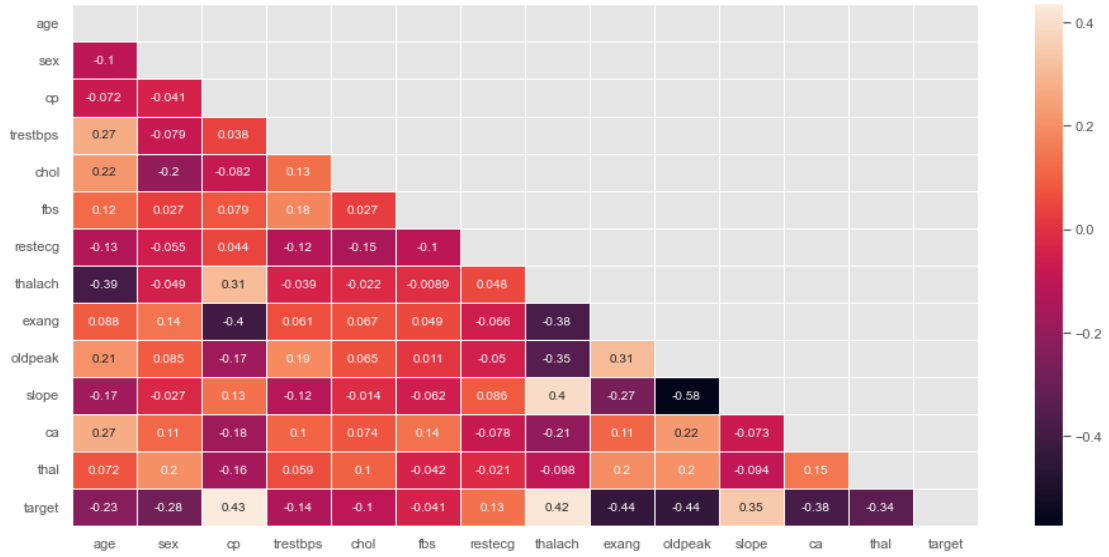


Fig 8.12 Correlation matrix for heart prediction

8.3.3 Liver Disease

The below ROC Curve in **Fig 8.13** obtained for the Liver disease helps us in visualising the comparison between the algorithms in terms of their performance and their ability to predict accurate output.

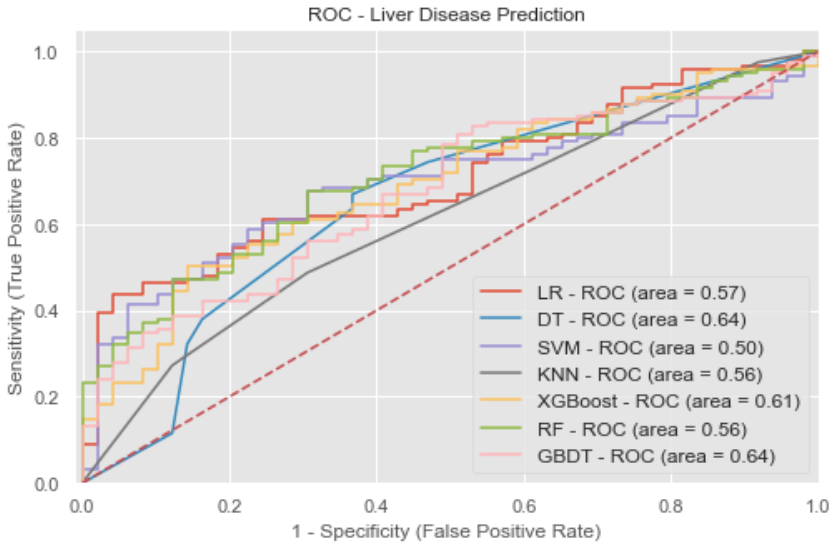


Fig 8.13 ROC graph for liver disease prediction

Fig 8.14 shows the correlation between liver disease attributes. Strong correlations are seen between Total Bilirubin and Direct Bilirubin, and between Albumin, Total Proteins, and Albumin_and_Globulin_Ratio. Age and Gender show weak

correlations, indicating unique contributions. This helps in identifying redundant features for model improvement.

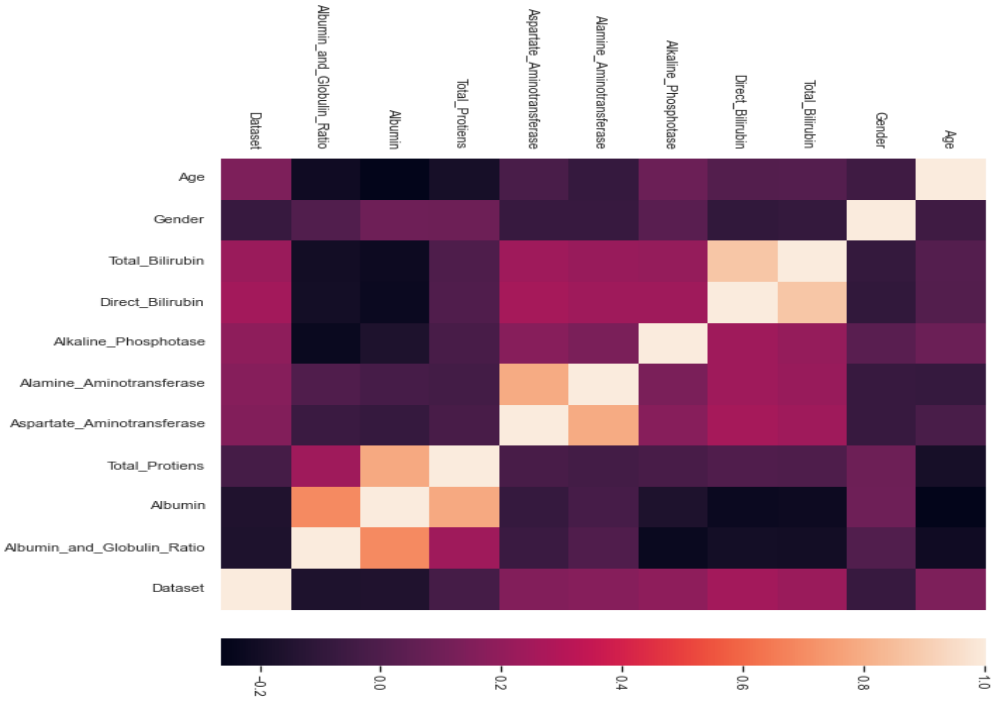


Fig 8.14 Distribution of attributes

8.3.4 Kidney Disease

The correlation matrix obtained for the Kidney disease is as follows, it considers all the parameters that we have used to obtain the accurate and best output.

Figure 8.15 displays the distribution of various kidney disease attributes. Most features, such as blood_pressure, sugar, serum_creatinine, and blood_urea, are highly skewed, indicating abnormal levels in patients. In contrast, features like haemoglobin, packed_cell_volume, and red_blood_cell_count show near-normal distributions.

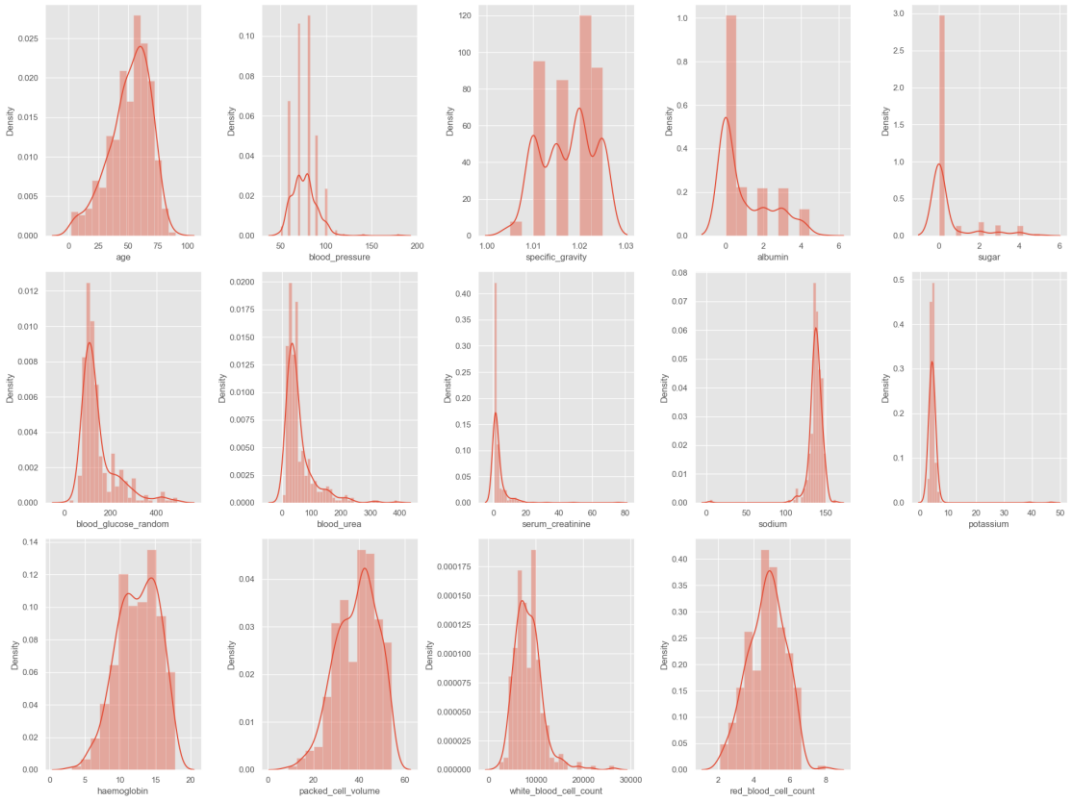


Fig 8.15 Distribution graphs for kidney prediction

The below ROC graph in **Fig 8.16** represents the performance evaluation of different models while training between the True Positives and False Positives:

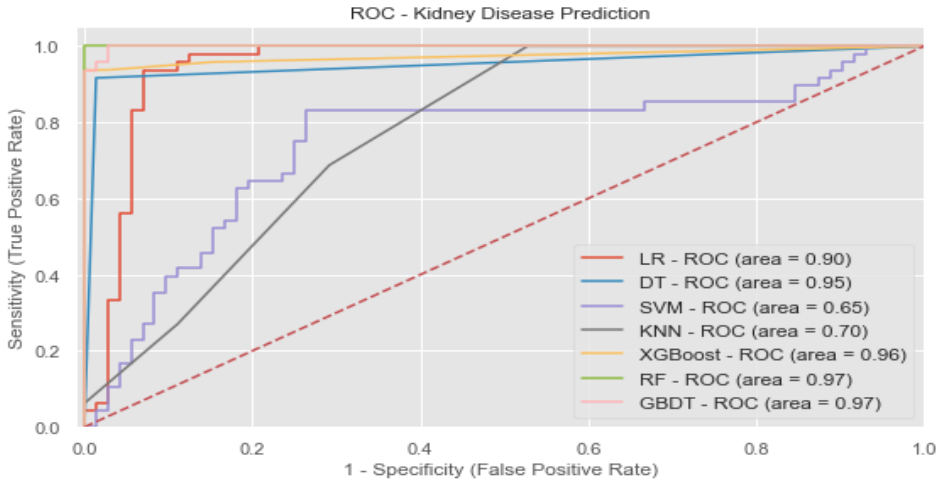


Fig 8.16 ROC graph for kidney prediction

8.3.5 Malaria

The two graphs plotting accuracy against epochs, where the blue line represents the training accuracy and the red line represents the validation accuracy in **Fig 8.17**. The

training accuracy increase steadily, indicating that the model is learning effectively. The training loss is decreasing over time, indicating effective learning.

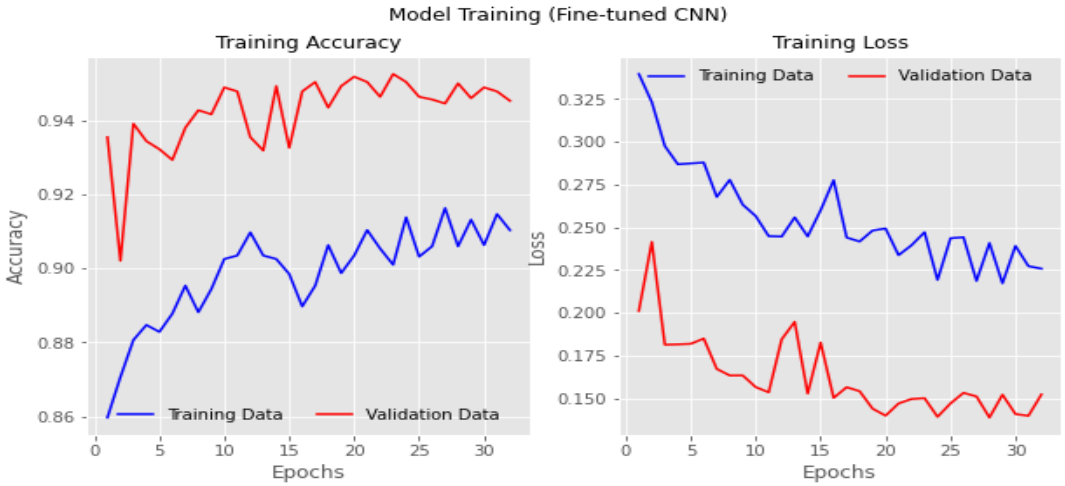


Fig 8.17 Training vs Validation graphs

The ROC Curve shown in **Fig 8.18** obtained for the Malaria disease helps us in visualising the comparison between the algorithms in terms of their performance and their ability to predict accurate output.

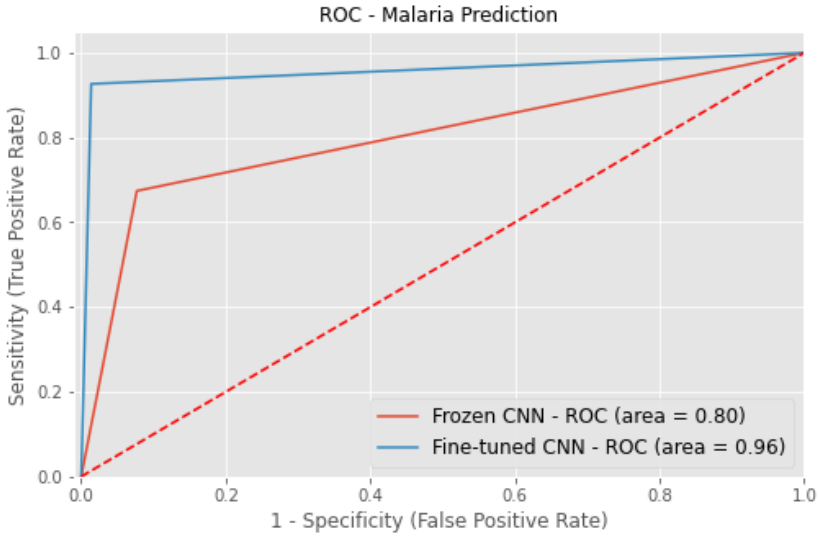


Fig 8.18 ROC graph for diabetes prediction

8.3.6 Pneumonia Disease

The training accuracy and loss graphs for pneumonia detection using CNN illustrate the model's learning behavior over epochs. The accuracy graph shows a steady

increase in training accuracy (blue line), indicating that the model is learning effectively, while the validation accuracy (red line) may fluctuate, suggesting potential overfitting if there is a large gap. In the loss graph shown in **Fig 8.19**, a decreasing training loss (blue line) confirms that the model is minimizing errors, while a stable or slightly decreasing validation loss (red line) indicates good generalization. However, if the validation loss starts increasing while training loss keeps decreasing, it suggests overfitting. To improve performance, techniques like data augmentation, dropout, or early stopping can be applied to enhance generalization and prevent overfitting.

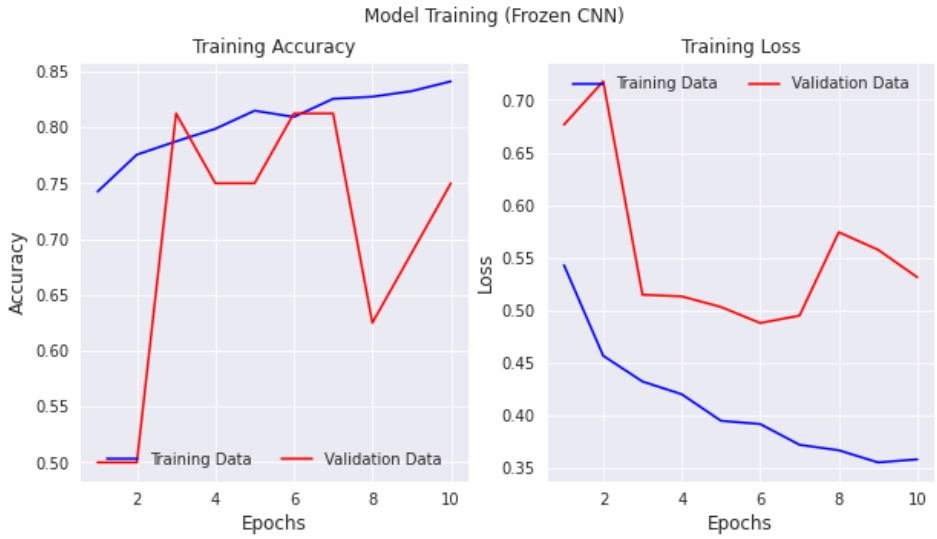


Fig 8.19 Training vs Validation graphs for Pneumonia

8.3.7 Parkinson's Disease

The below image presents visualizations of data distributions using box plots and histograms for three features: Age, Creatinine Phosphokinase, and Platelets. The left column contains box plots, which highlight the median, interquartile range, and potential outliers in the dataset. The right column displays histograms with density curves, showing the distribution shape of each feature. Vertical lines in the histograms represent statistical percentiles such as the mean, 25th, 50th, and 75th percentiles. The age distribution appears right-skewed, while creatinine phosphokinase shows significant outliers. These visualizations help in understanding feature distribution, skewness, and potential anomalies in the dataset as shown in **Fig 8.20**

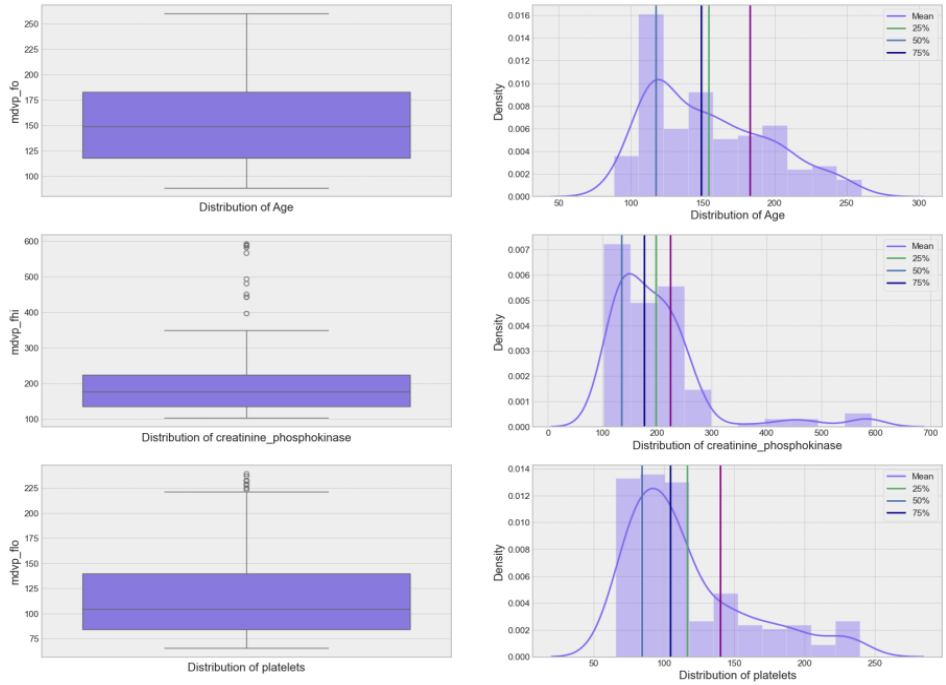


Fig 8.20 Distribution Graph for Parkinson's

The correlation matrix for the collected attributes is shown in **Fig 8.21**,

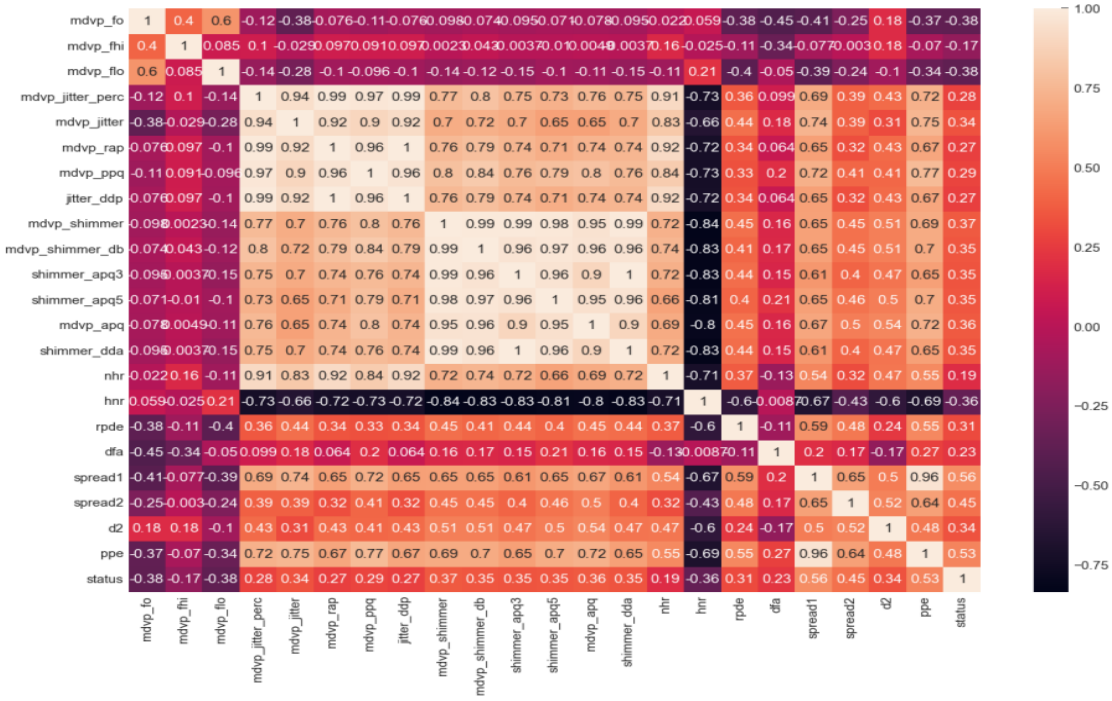


Fig 8.21 Correlation matrix for Parkinson's

8.3.8 Alzheimer's Disease

Fig 8.22 presents training and validation performance graphs for Alzheimer's disease detection using CNN models. It compares two approaches: a baseline CNN (top row) and an augmented CNN (bottom row). The loss graphs (left column) show how training and validation losses decrease over epochs, indicating model learning. The accuracy graphs (right column) illustrate how the training and validation accuracy evolve over time. The baseline CNN achieves high accuracy with minimal epochs, while the augmented CNN has more fluctuations in validation accuracy. The augmented model appears to struggle with generalization, as indicated by the gap between training and validation accuracy. These plots help evaluate the effectiveness of data augmentation in improving model performance.

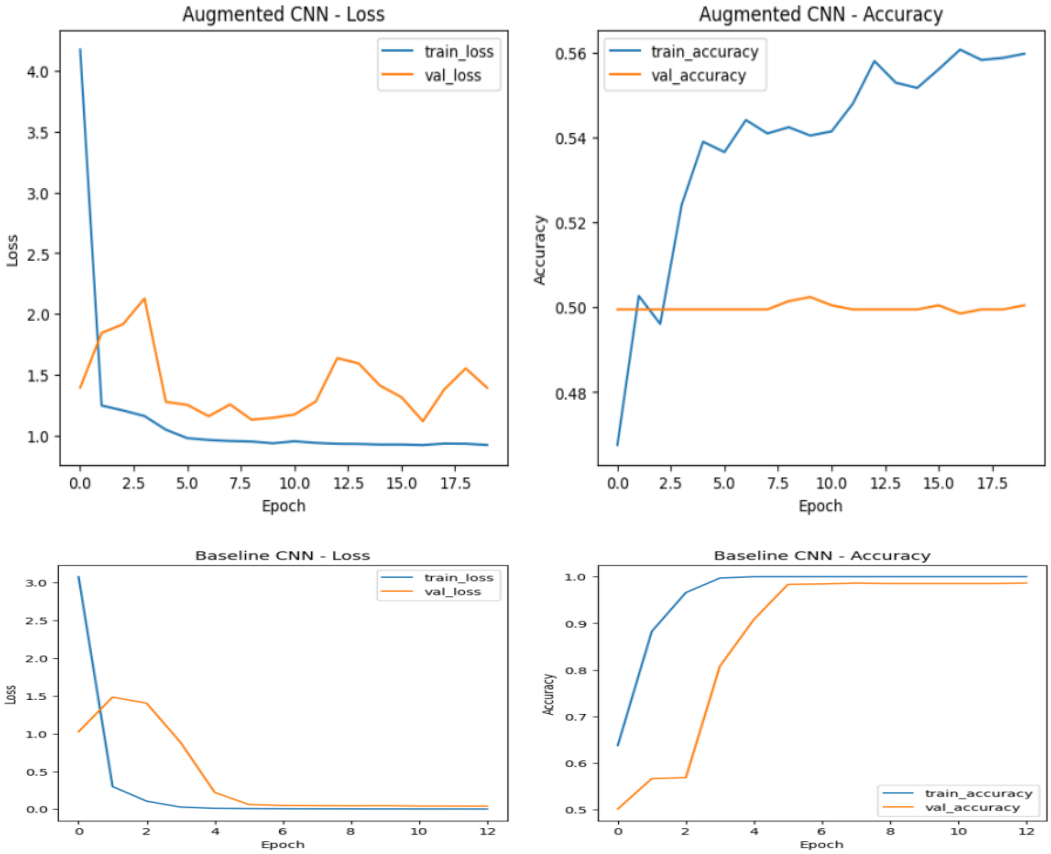


Fig 8.22 Training vs Validation graphs for Alzheimer's

The below **Fig 8.23** displays training and validation performance graphs for a ResNet50 model applied using transfer learning and fine-tuning. The top row represents the results for transfer learning, showing a steady decline in training loss

while validation loss stabilizes at a low value. Accuracy for transfer learning improves quickly and then plateaus. The bottom row illustrates the performance of fine-tuned ResNet50, where training loss fluctuates more, and validation loss shows instability. Accuracy for fine-tuning exhibits more variations compared to transfer learning, suggesting potential challenges in generalization. These graphs help compare the effectiveness of the two approaches for the given dataset.

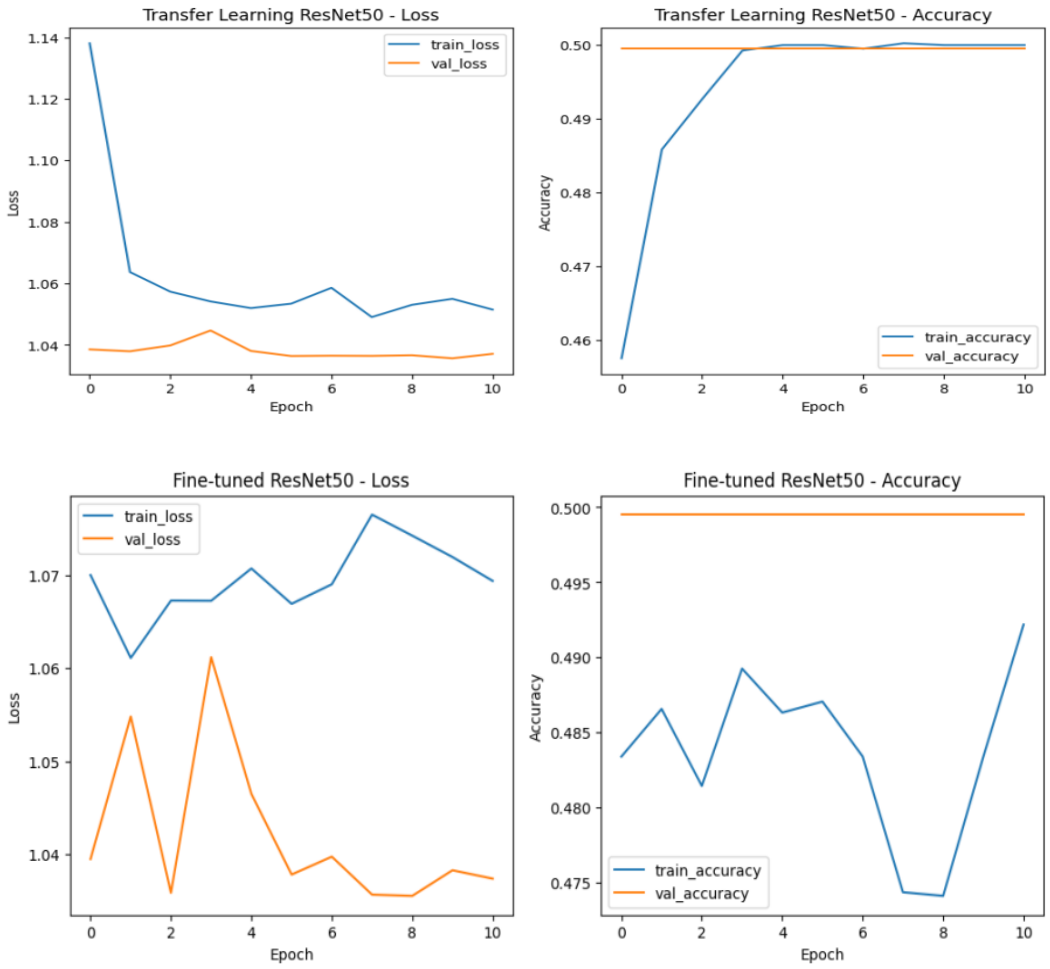


Fig 8.23 Training vs Validation graphs for Alzheimer's (resNet50)

8.3.9 Breast Cancer

The below bar chart in **Fig 8.24** compares the performance of four machine learning models: Logistic Regression, K-Nearest Neighbours (KNN), Random Forest, and Support Vector Machine (SVM). Each bar is color-coded and represents the respective model's performance, likely in terms of accuracy or another evaluation metric. The x-axis appears to range from 0 to 1, suggesting a percentage-based measure. SVM seems to have the highest performance, followed by Logistic Regression, Random Forest,

and KNN. This visualization helps in assessing which model performs best for the given dataset.

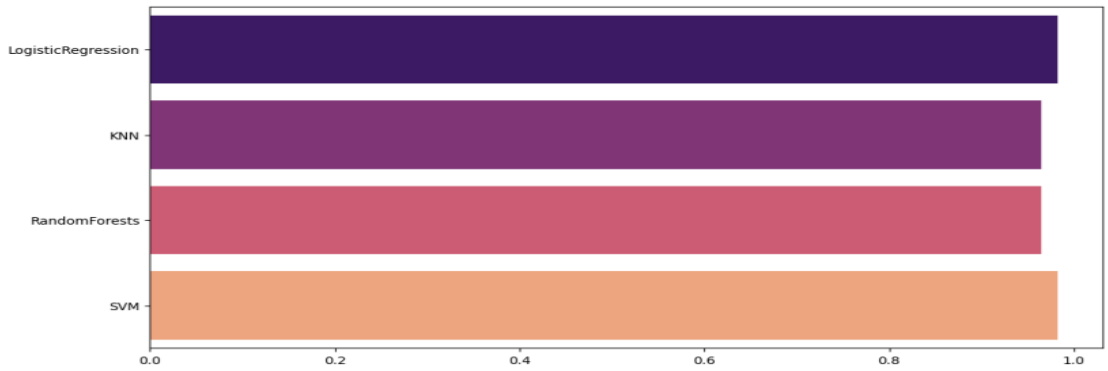


Fig 8.24 Visualization of training results of different models

8.4 OUTPUT SCREENS

This section **8.4** presents the final web interface of the system, including visual outputs for each of the 10 diseases integrated into the project. It highlights how users can input medical data or images, view prediction results, and navigate through the interface. Each screen demonstrates how the backend models are connected to the frontend, ensuring a smooth, real-time prediction experience for diseases like diabetes, heart, liver, kidney, Parkinson's, breast cancer, pneumonia, malaria, Alzheimer’s, and flu.

8.4.1 Home Page

The landing page of the application provides an overview and navigation options for each disease prediction module. **Fig 8.4.1.1 and 8.4.1.2** show the home page of the project.

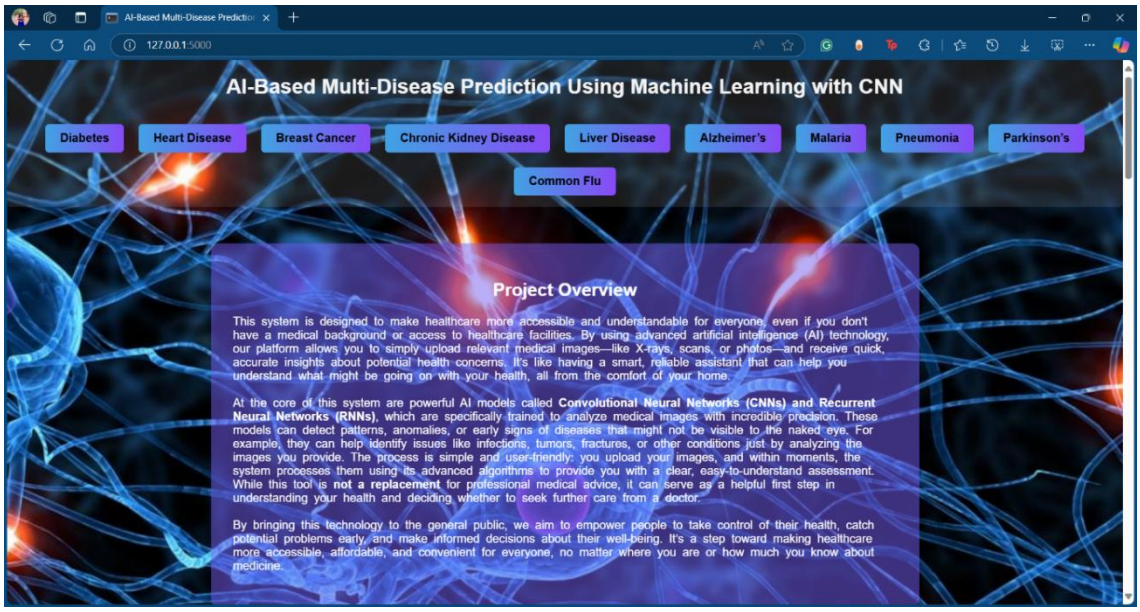


Fig 8.4.1.1

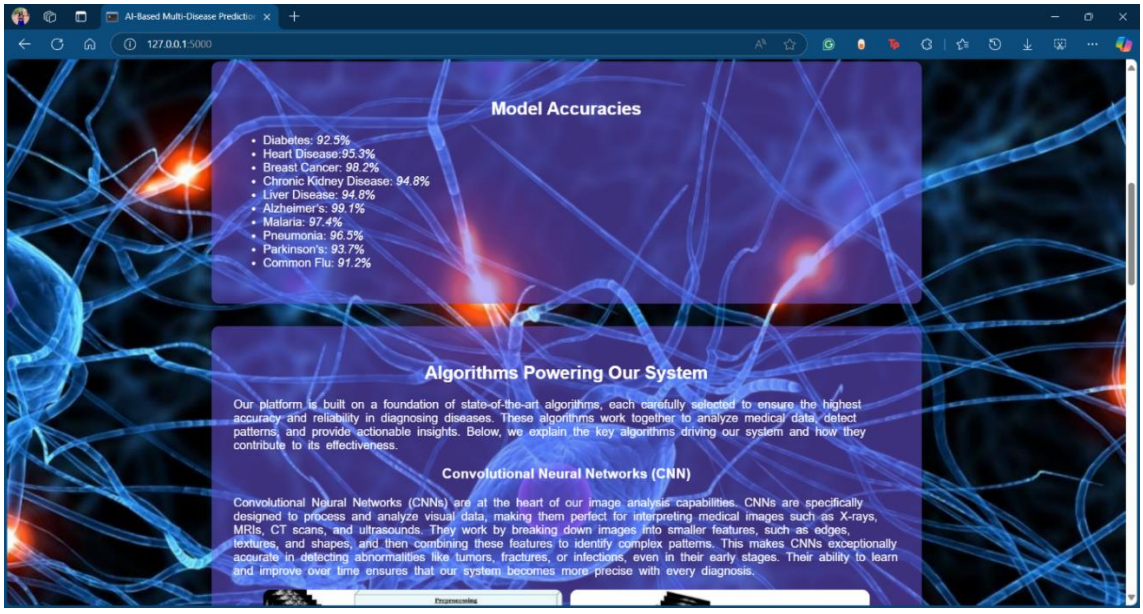


Fig 8.4.1.2

8.4.2 Diabetes

This page allows users to input health parameters and receive a prediction on whether the person is diabetic or not. **Fig 8.4.2.1** shows the home page for diabetes.

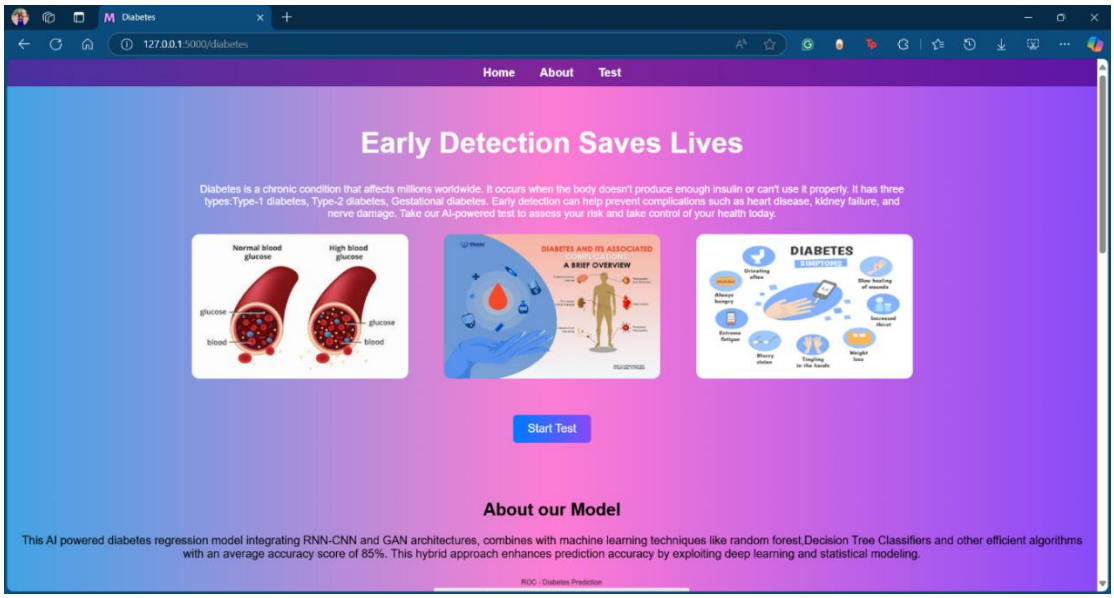


Fig 8.4.2.1

Fig 8.4.2.2 shows the diabetes risk test.

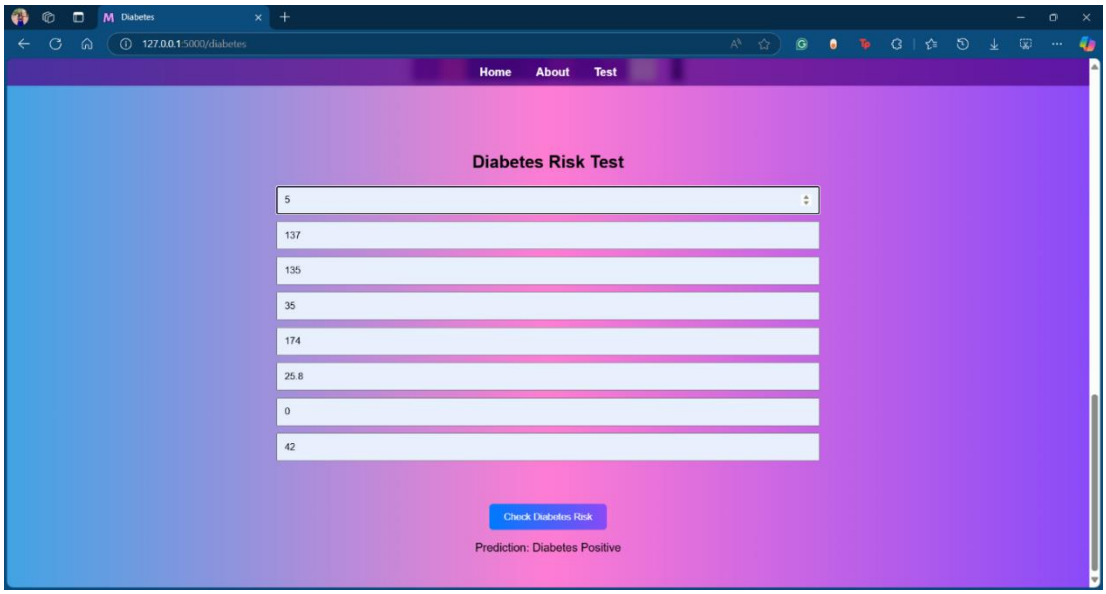


Fig 8.4.2.2

8.4.3 Heart Disease

Users can enter cardiovascular data to check for the presence or absence of heart disease. **Fig 8.4.3.1** shows the heart disease test with low risk.

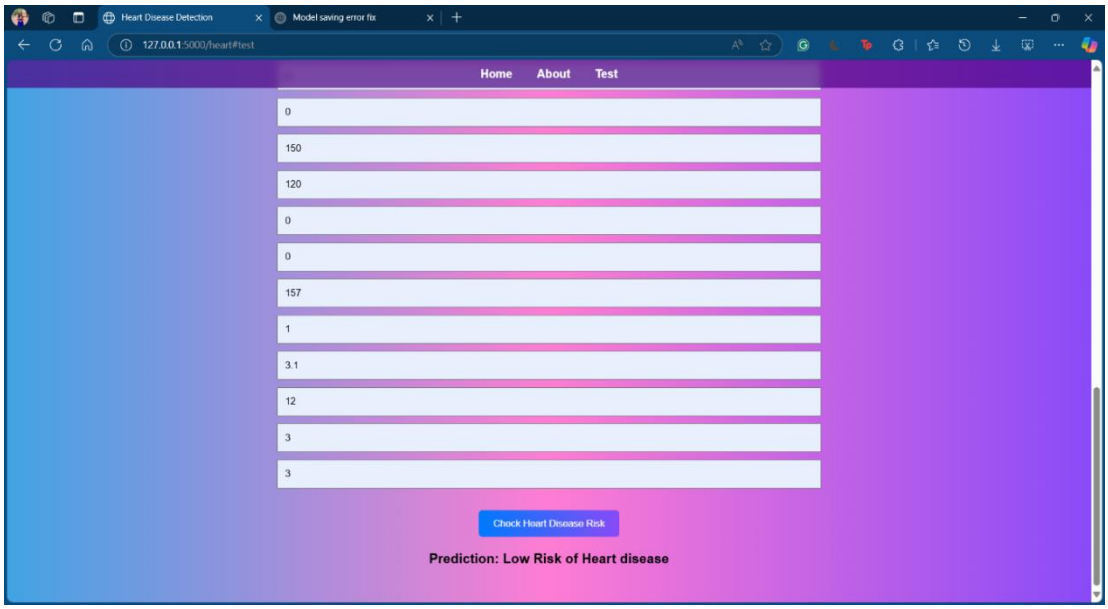


Fig 8.4.3.1

Fig 8.4.3.2 shows the heart disease test with high risk.

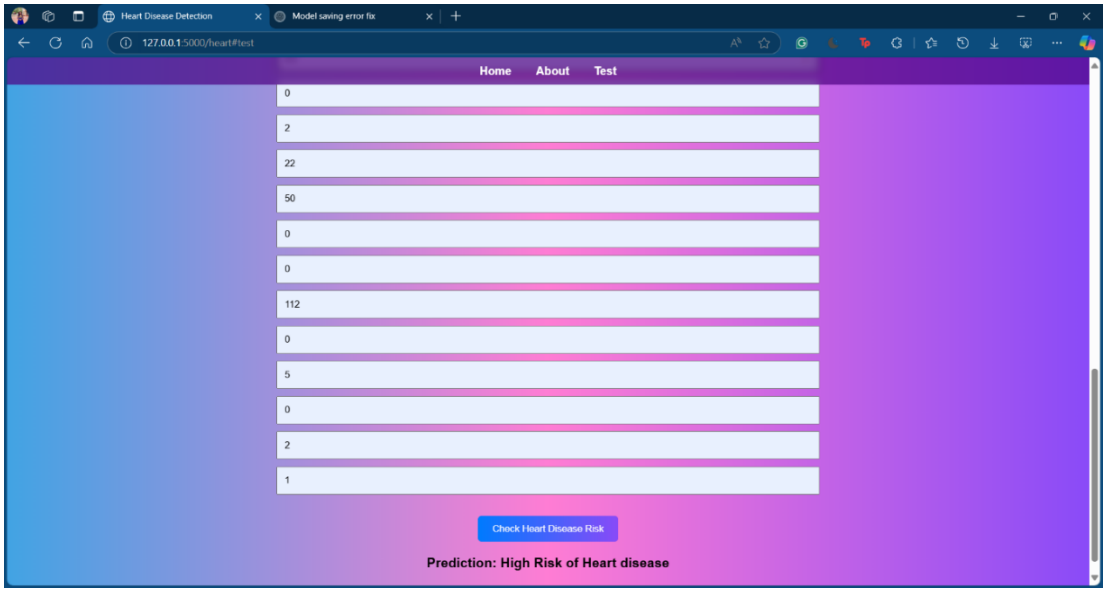


Fig 8.4.3.2

8.4.4 Breast Cancer

Users upload or input features to get predictions indicating whether the tumor is malignant or benign. **Fig 8.4.4.1** shows the breast cancer test as malignant.

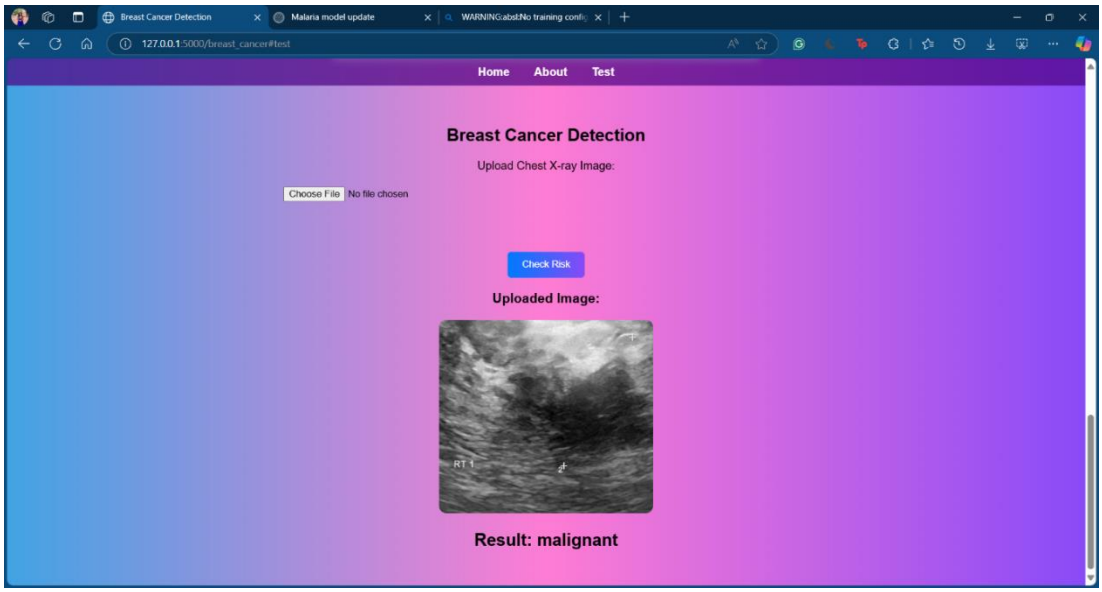


Fig 8.4.4.1

Fig 8.4.4.2 shows the breast cancer test as benign.

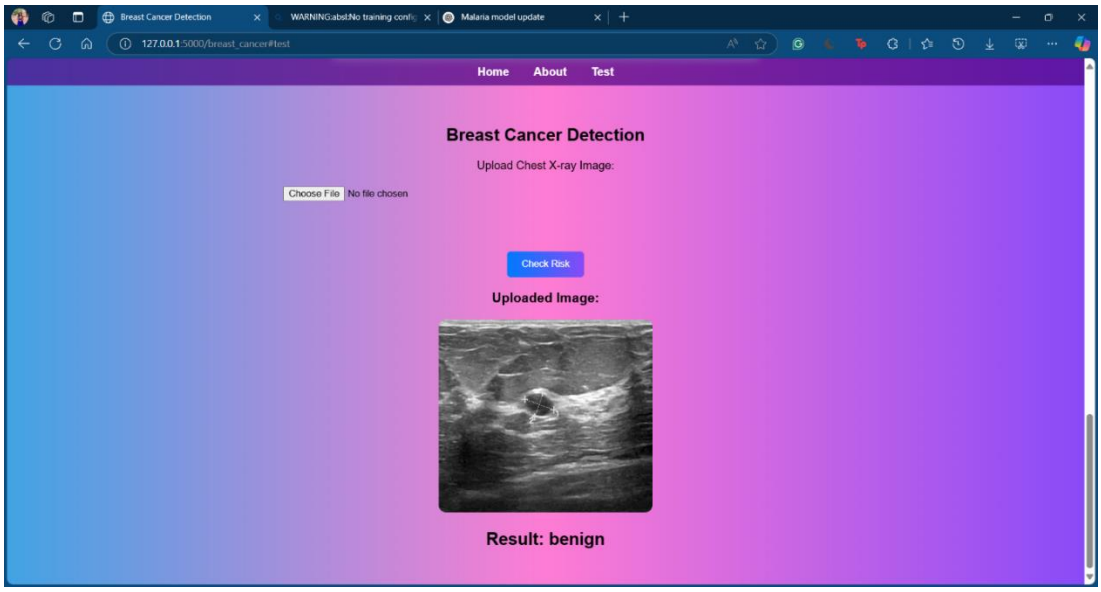


Fig 8.4.4.2

8.4.5 Chronic Kidney Disease

This page predicts the presence or absence of chronic kidney disease based on lab values. **Fig 8.4.5.1** shows the Chronic Kidney Disease test with low risk.

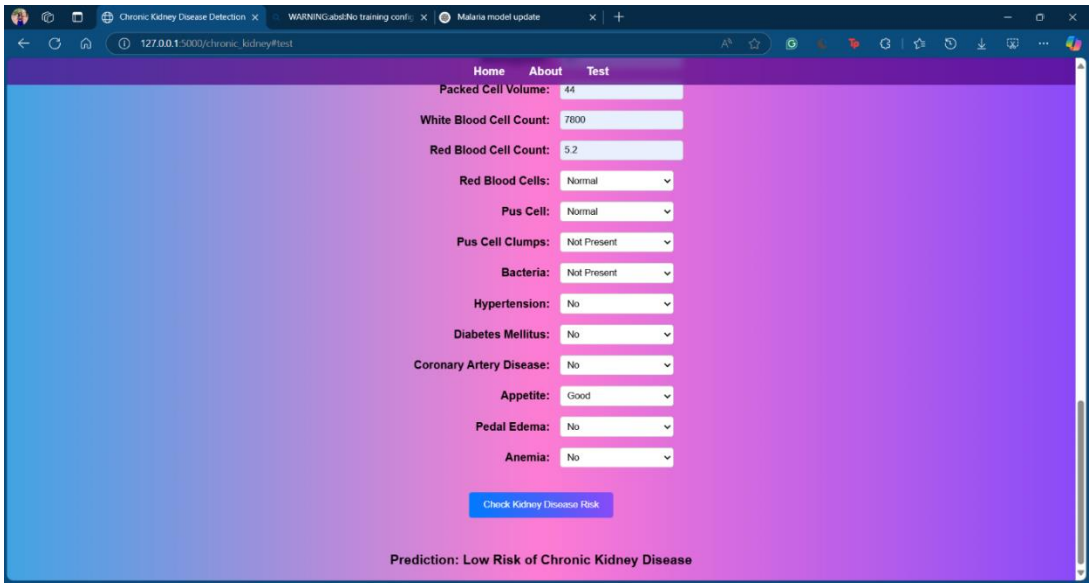


Fig 8.4.5.1

Fig 8.4.5.2 shows the Chronic Kidney Disease test with high risk.

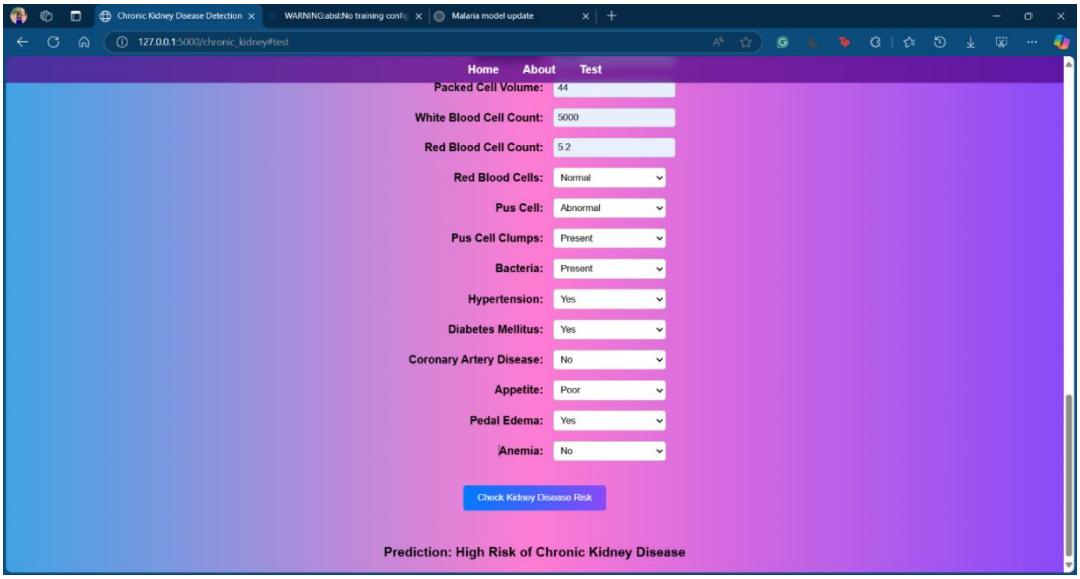


Fig 8.4.5.2

8.4.6 Liver Disease

User inputs related to liver function are analyzed to determine the presence or absence of liver disease. **Fig 8.4.6.1** shows the liver disease test as positive.

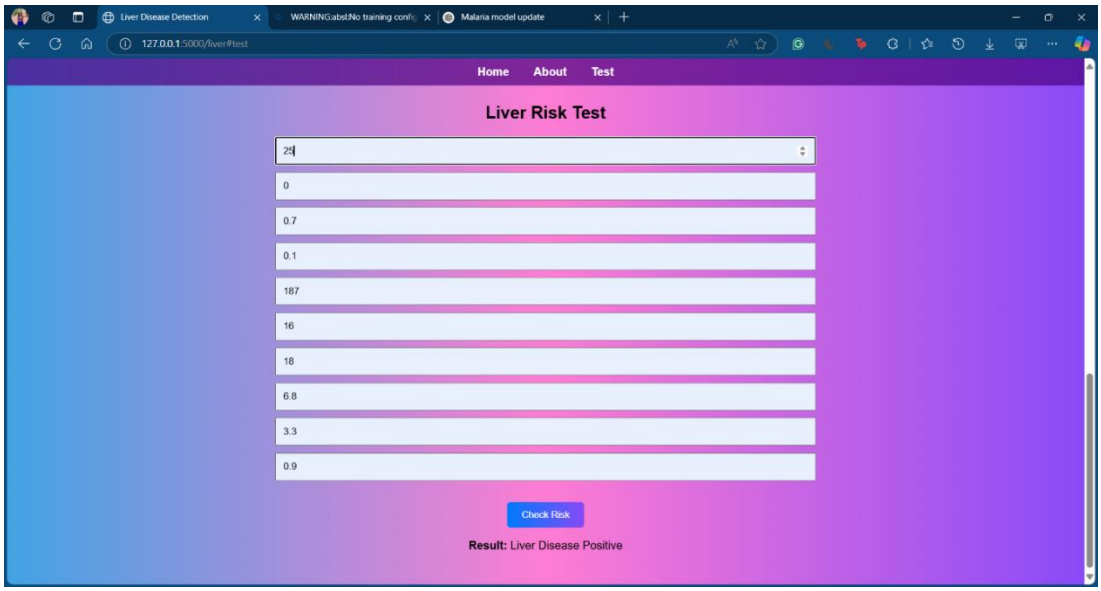


Fig 8.4.6.1

Fig 8.4.6.2 shows the liver disease test as negative.

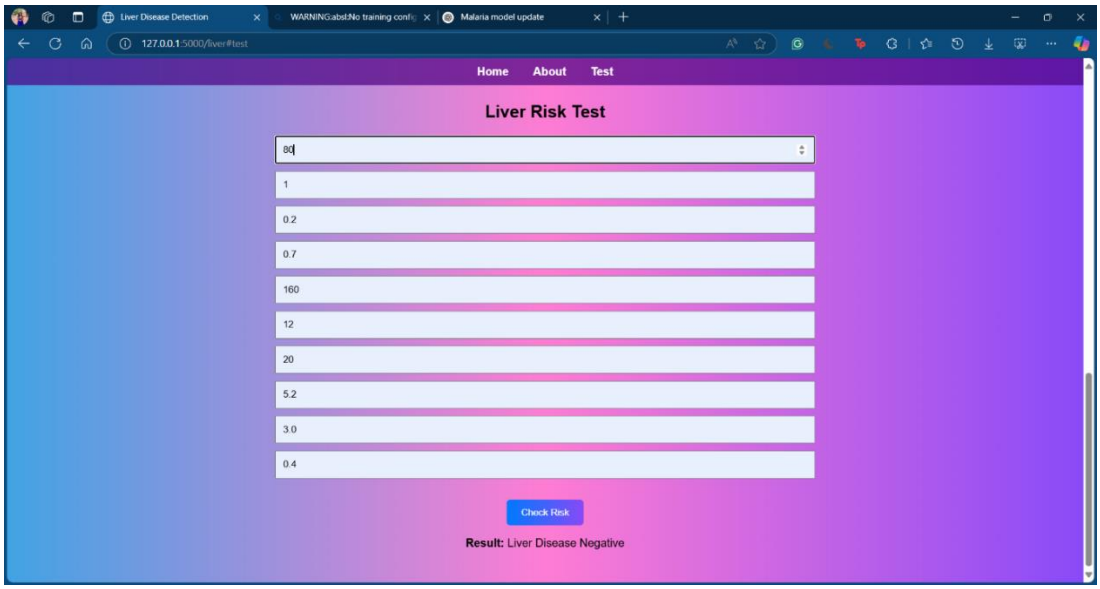


Fig 8.4.6.2

8.4.7 Alzheimer's

Image-based results are shown here, predicting whether the person has Alzheimer's or not using CNN. **Fig 8.4.7.1** shows the Alzheimer's prediction as Non-Demented.

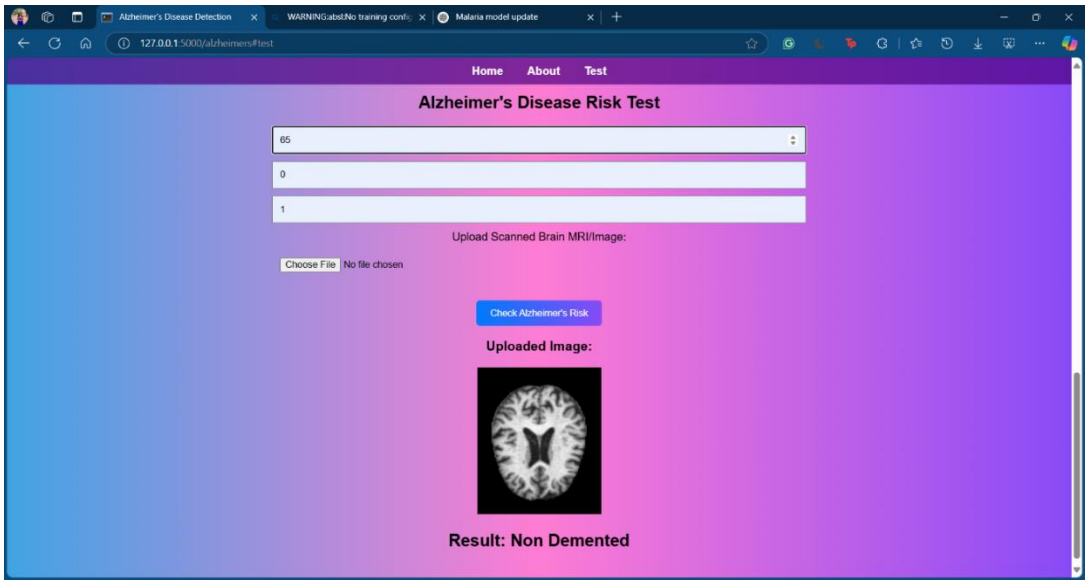


Fig 8.4.7.1

Fig 8.4.7.2 shows the Alzheimer's prediction as Very Mild Demented.

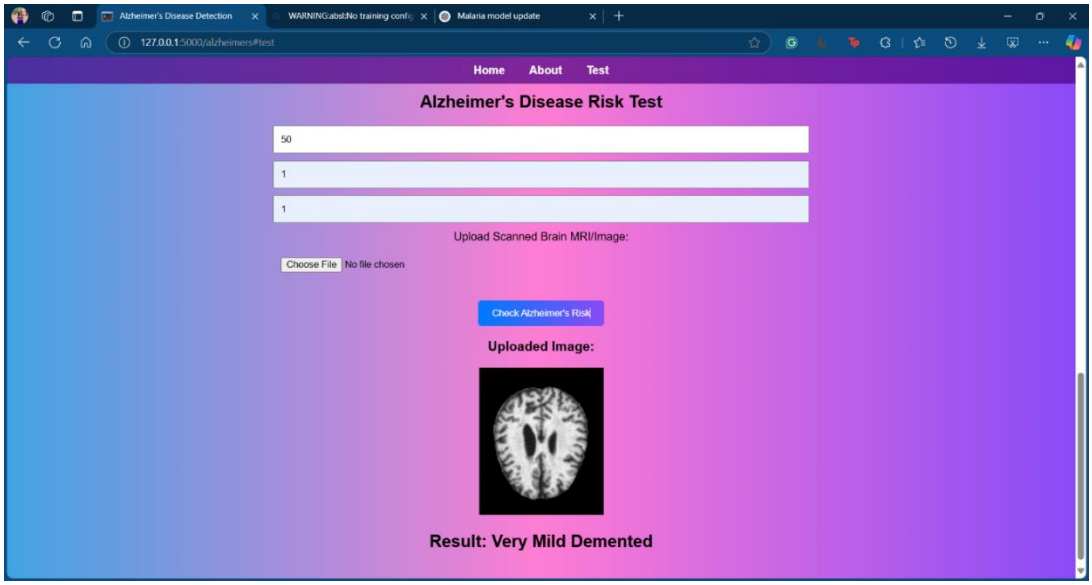


Fig 8.4.7.2

Fig 8.4.7.3 shows the Alzheimer's prediction as Mild Demented.

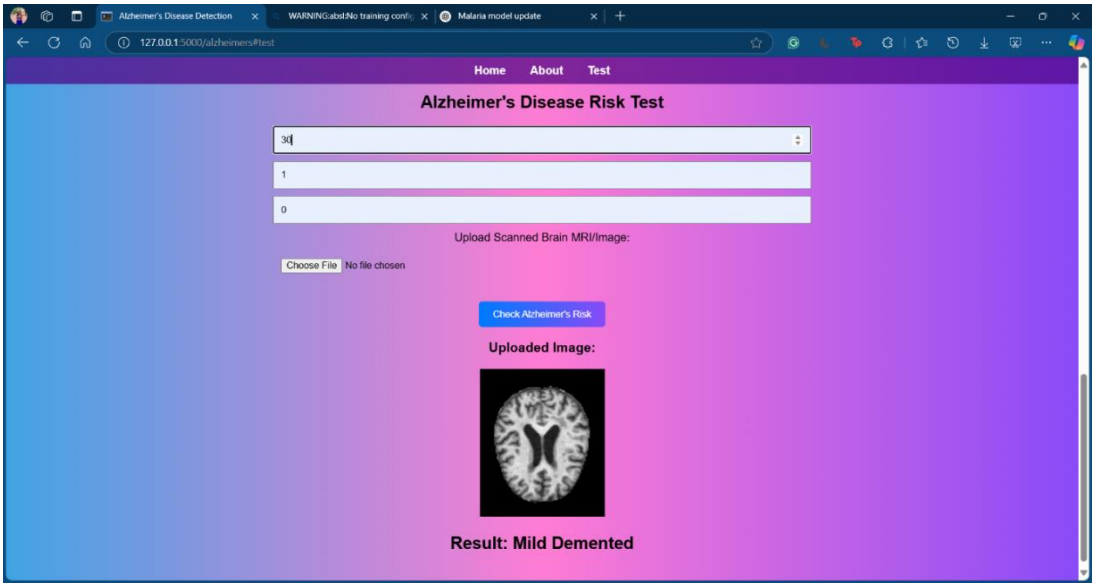


Fig 8.4.7.3

Fig 8.4.7.4 shows the Alzheimer's prediction as Moderate Demented.

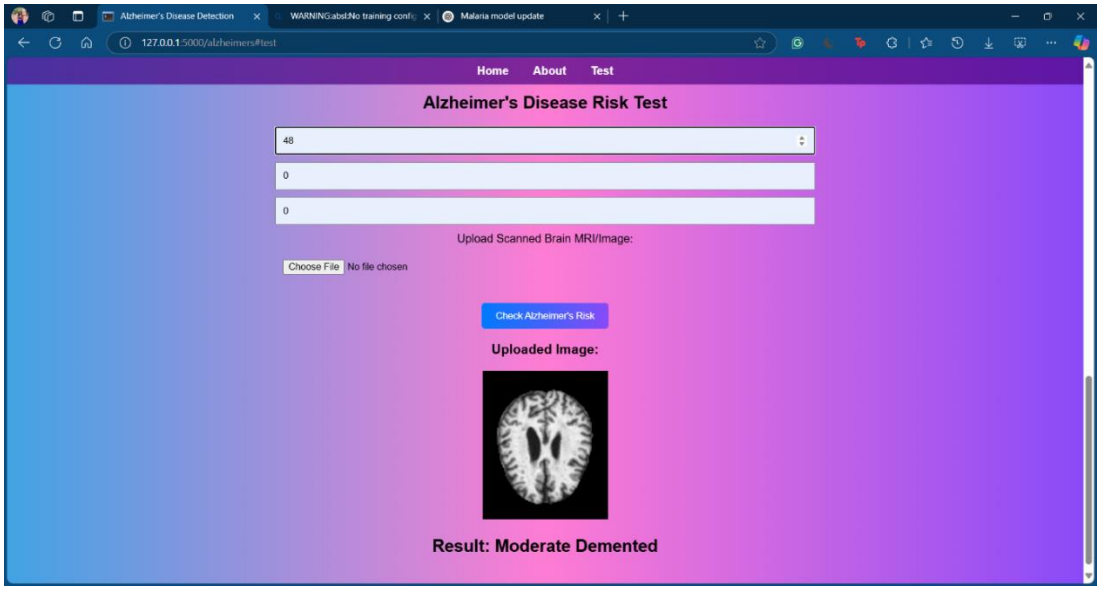


Fig 8.4.7.4

8.4.8 Malaria

Microscopic image uploads are used to detect malaria-infected and uninfected cells.

Fig 8.4.8.1 shows that Malaria cell is infected.

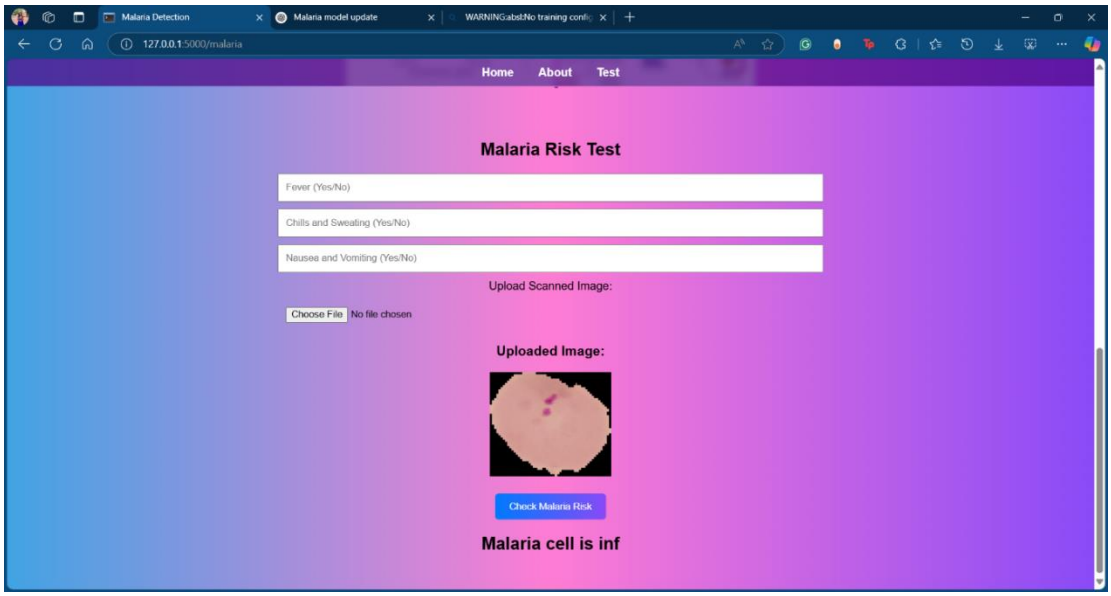


Fig 8.4.8.1

Fig 8.4.8.2 shows that Malaria cell is healthy.

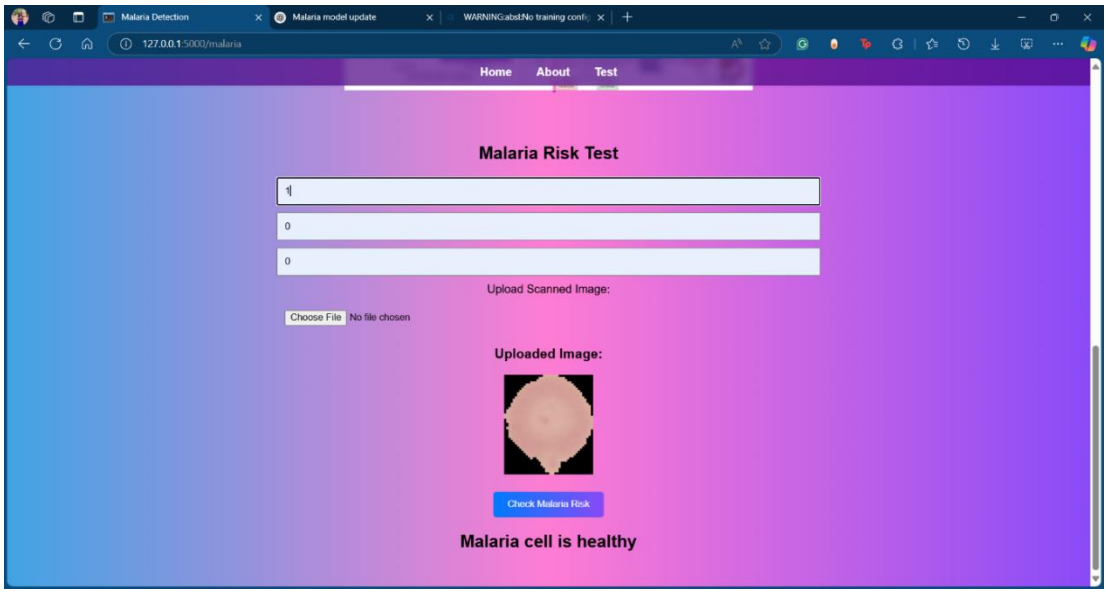


Fig 8.4.8.2

8.4.9 Pneumonia

Chest X-ray images are analyzed to predict the presence or absence of pneumonia. **Fig 8.4.9.1** shows that chest X-ray is normal.

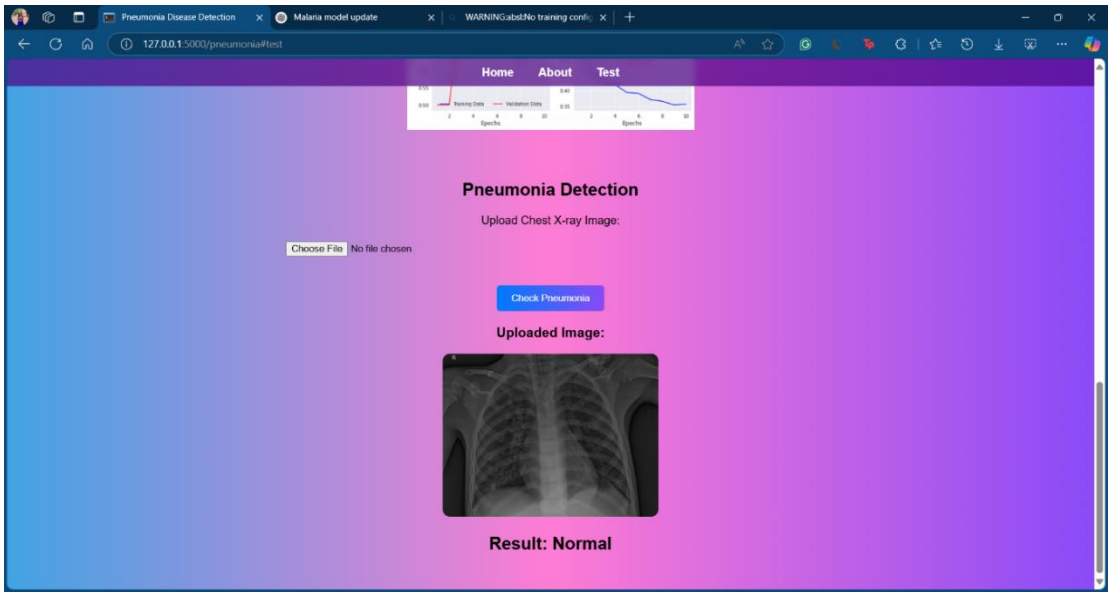


Fig 8.4.9.1

Fig 8.4.9.2 shows that chest X-ray is Pneumonia affected.

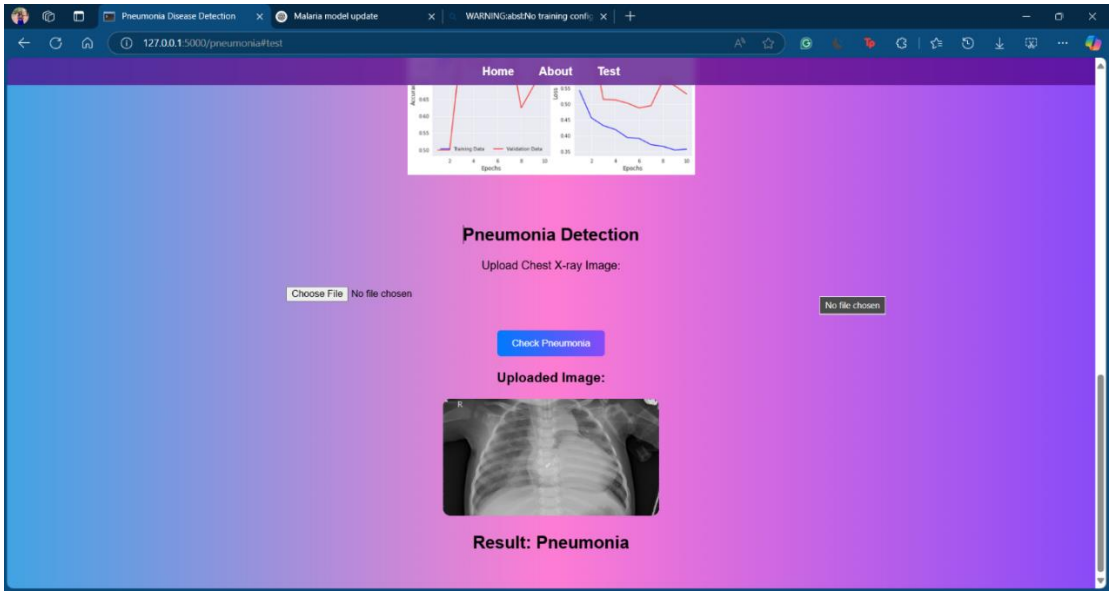


Fig 8.4.9.2

8.4.10 Parkinson's

This module predicts whether a person is affected by Parkinson's disease using voice and movement parameters. **Fig 8.4.10.1** shows that drawing is normal.

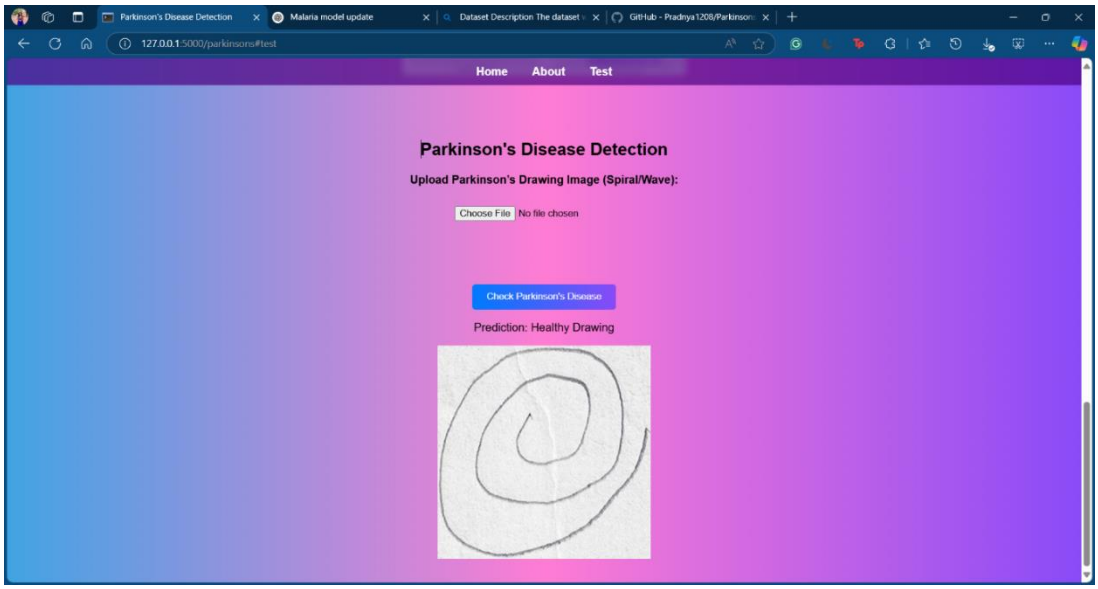


Fig 8.4.10.1

Fig 8.4.10.2 shows that drawing is Parkinson's affected.

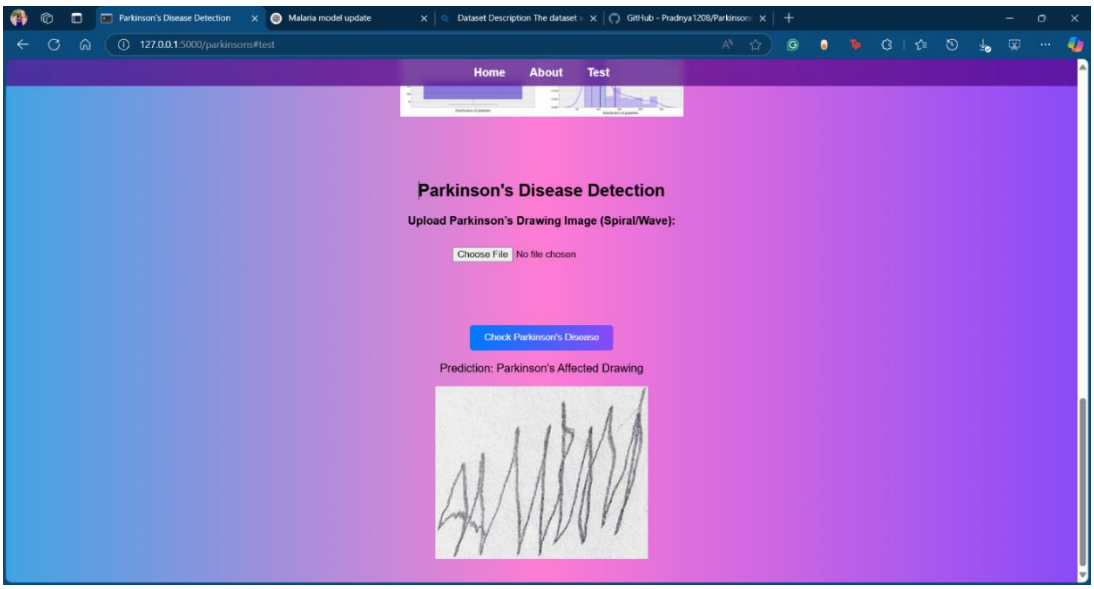


Fig 8.4.10.2

8.4.11 Common Flu

Based on symptoms, this page predicts whether the user is suffering from common flu or not. **Fig 8.4.11.1** shows that patient has severe flu.

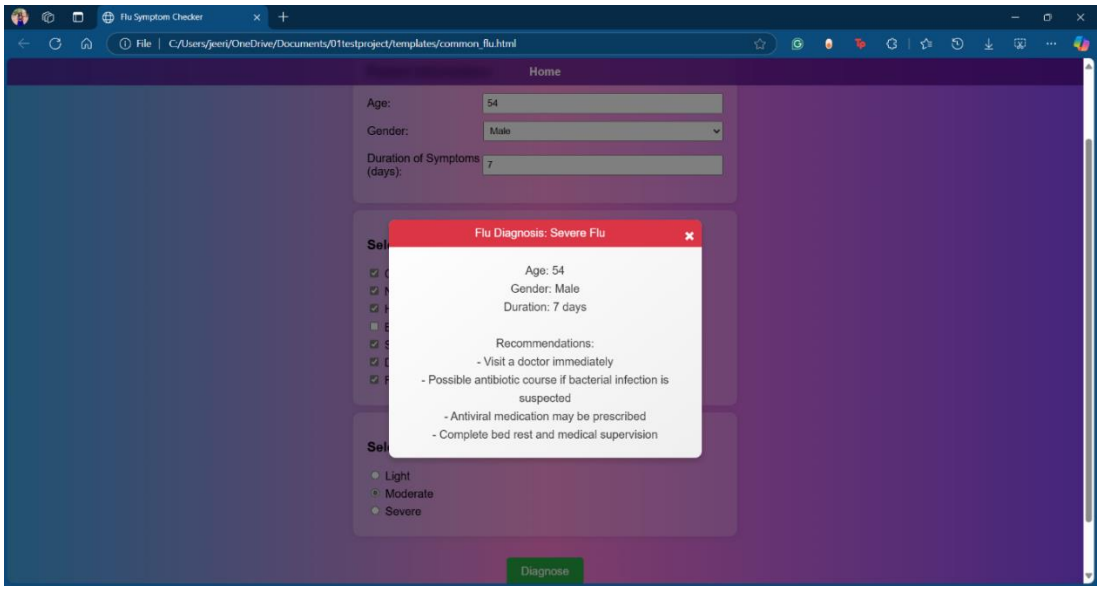


Fig 8.4.11.1

Fig 8.4.11.2 shows that patient has moderate flu.

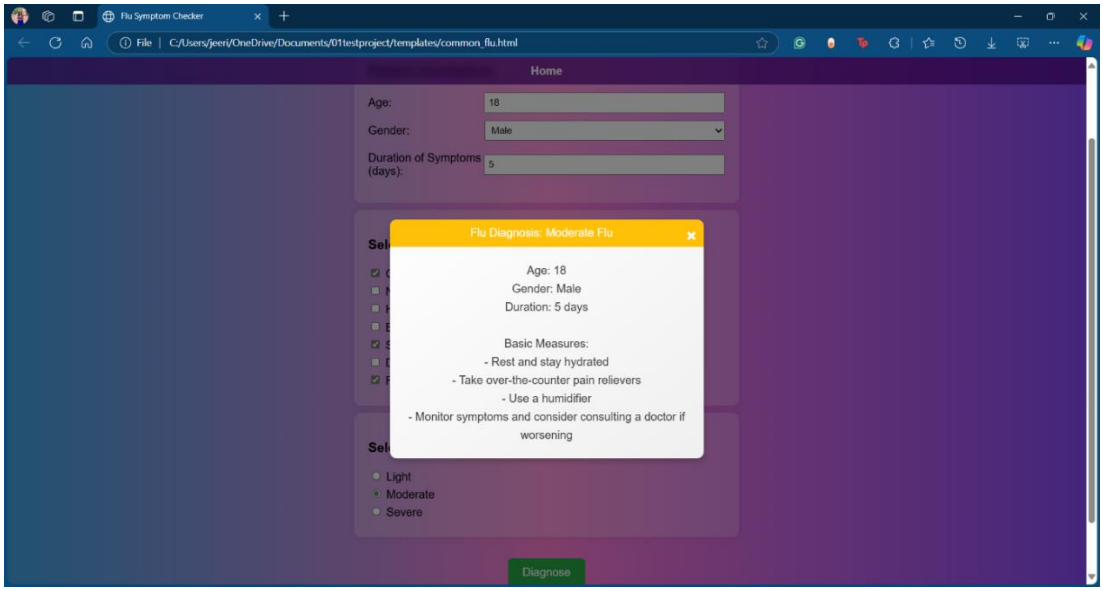


Fig 8.4.11.2

Fig 8.4.11.3 shows that patient has mild flu.

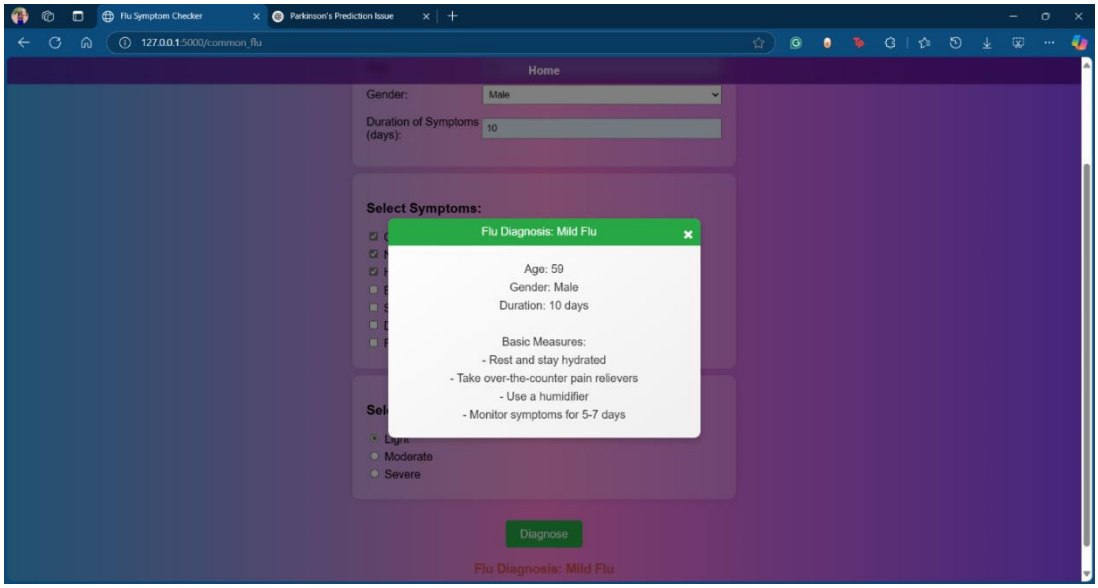


Fig 8.4 11.3

8.5 CONCLUSION

The system conducted a model performance assessment of machine learning and deep learning methods across ten different diseases.

- The Random Forest Classifier demonstrated superior performance in tabular data-based diseases including diabetes, kidney and heart since it provided accurate results.
- The SVM model delivered the most effective outcome when analyzing liver disease data because it excels at processing small and complicated datasets.
- Fine-Tuned CNNs delivered superior performance compared to other models when analyzing image-based diseases specifically malaria and pneumonia.
- The structured input data received effective processing through the implementation of XGBoost in Parkinson's disease predictions.
- The Baseline CNN achieved high recall and F1 scores in Alzheimer's detection while maintaining a low precision rate.
- Logistic Regression proved most effective for breast cancer diagnosis by providing both performance quality and operational simplicity.

- The JavaScript-based diagnostic logic delivered fast and precise results through a lightweight system for symptom-based common flu diagnosis.

The results are shown in **Table 8.5.1**

Disease	Best Performing Model	Accuracy
Diabetes	Random Forest Classifier	92.54%
Heart Disease	Random Forest Classifier	99.16%
Liver Disease	Support Vector Machine (SVM)	91.18%
Kidney Disease	Random Forest Classifier	99.16%
Malaria	CNN	96%
Pneumonia	CNN-Based Model	91%
Parkinson’s	XGBoost Classifier	98.31
Alzheimer’s	Baseline CNN	98.41%
Breast Cancer	Logistic Regression	98%
Common Flu	JavaScript Logic	99%

Table 8.5.1 Results of different diseases

The section displayed complete output screens for ten diseases which included positive and negative prediction results. Users experience system interaction through the home page and specialized interfaces which serve each disease including diabetes, heart disease, breast cancer, chronic kidney disease, liver disease, Alzheimer’s, malaria, pneumonia, Parkinson’s, and flu. The chapter presented both ROC curves and training versus validation accuracy graphs and confusion matrices to help visualize and validate model performance. The visual representations improve both the interpretability and reliability of developed models thus supporting the project's mission to deliver accurate user-friendly AI-based healthcare assistance.

CHAPTER-9 CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION:

The AI-Based Multi-Disease Detection System represents an advancement in the field of healthcare diagnostics by using the power of Convolutional Neural Networks (CNNs).

Unlike traditional diagnostic tools that focus on a single disease, this system provides an integrated platform capable of detecting multiple diseases, including heart disease, kidney disease, diabetes, Parkinson's, Alzheimer's, common flu, breast cancer, malaria, and pneumonia. By consolidating disease detection into a single, efficient system, it reduces complexity, saves time, and enhances the overall effectiveness of medical evaluations.

The proposed system addresses the key limitations of existing techniques, such as poor generalization, limited multi-modal data handling, and lack of user accessibility. Through the use of diverse datasets, advanced deep learning algorithms, the models achieve high accuracy, reliability, ensuring general users can trust the diagnostic outcomes. The web interface makes the system accessible to non-technical users, expanding its reach to remote areas with limited healthcare infrastructure.

9.2 FUTURE SCOPE:

Development of a cross-platform mobile app for Android and iOS to make the system even more accessible, allowing users to get quick health insights anywhere, even in remote or underdeveloped areas with limited access to healthcare facilities. Future versions of the app can incorporate offline diagnostic features, allowing users to receive initial health assessments without an active internet connection.

APPENDIX

GLOSSARY

1. **AI (Artificial Intelligence)**: Simulation of human intelligence processes by machines, especially in medical diagnosis tasks.
2. **ML (Machine Learning)**: An AI technique that allows systems to learn from data to make predictions or classifications.
3. **CNN (Convolutional Neural Network)**: A deep learning algorithm used in this project to analyze medical images like X-rays and MRIs.
4. **RNN (Recurrent Neural Network)**: A neural network architecture used for sequence-based data like patient records or time-series health logs.
5. **Confusion Matrix**: A tabular summary showing True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), used for evaluating classification accuracy.
6. **TP (True Positive)**: Correctly predicted positive diagnosis (e.g., disease present and predicted as present).
7. **TN (True Negative)**: Correctly predicted absence of a disease.
8. **FP (False Positive)**: The model predicts a disease that is not actually present.
9. **FN (False Negative)**: The model fails to detect an actual disease.
10. **Accuracy**: (TP + TN) / (TP + TN + FP + FN); a measure of overall model correctness.
11. **Precision**: TP / (TP + FP); how many of the predicted positive cases were actually correct.
12. **Recall (Sensitivity)**: TP / (TP + FN); how well the model detects actual positive cases.
13. **F1-Score**: The harmonic mean of precision and recall, balancing both concerns.

14. **ROC-AUC**: A performance measure for classification problems using probability scores.
15. **Flask**: Backend framework used to connect trained models with the web interface and handle user requests.
16. **Jupyter Notebook**: A browser-based development tool used for data preprocessing, training, and evaluating ML models.
17. **Visual Studio Code (VS Code)**: Primary IDE used for writing Python code, Flask apps, and front-end development.
18. **OpenCV**: Used for preprocessing and handling medical images before passing them to CNNs.
19. **Kaggle**: A source for publicly available medical datasets like X-rays, lab results, and patient records.
20. **Random Forest Classifier**: A machine learning algorithm used for structured/tabular data (e.g., for diabetes, heart, and kidney prediction).
21. **Support Vector Machine (SVM)**: An ML algorithm used for liver disease prediction due to its high accuracy on that dataset.
22. **Fine-Tuned CNN**: A customized CNN model trained on malaria and pneumonia images for better performance and higher accuracy.
23. **XGBoost**: A gradient boosting framework used for Parkinson's detection due to its effectiveness in handling imbalanced data.
24. **Baseline CNN**: A simple CNN architecture used as a starting point for Alzheimer's detection.
25. **Feature Engineering**: The process of selecting and transforming raw data (e.g., lab values, symptoms) into meaningful input features for ML models.
26. **CKD (Chronic Kidney Disease)**: One of the diseases detected using a model trained on lab report data.
27. **Sklearn (Scikit-learn)**: A Python machine learning library used for implementing, training, and evaluating traditional ML algorithms.

28. **TensorFlow**: A powerful deep learning framework used to build and train Convolutional Neural Network (CNN) models for image-based disease detection like Alzheimer's, malaria, and pneumonia.
29. **Keras**: A high-level API that simplifies the usage of TensorFlow, enabling quick prototyping and streamlined model development.
30. **NumPy**: A fundamental Python library for numerical operations and efficient handling of multi-dimensional arrays.
31. **Pandas**: A data manipulation library in Python used for reading, cleaning, and managing structured data such as CSV datasets from Kaggle.
32. **Plotly / Matplotlib**: Visualization libraries used to create charts like ROC curves, distribution graphs, and confusion matrices for performance analysis.
33. **Model Fine-Tuning**: The process of improving a pre-trained model by modifying parameters, architecture, or layers to suit specific medical datasets and improve performance.
34. **Cross-validation**: A validation technique used to test model reliability by training it on multiple data subsets and assessing its performance on the rest.
35. **Medical Imaging Dataset**: Datasets consisting of medical images such as X-rays or CT scans used for training CNN models for diseases like pneumonia or Alzheimer’s.

SAMPLE SOURCE CODE

malaria.ipynb

```
import os
import shutil

# Define train, validation, and test directories
train_dir = 'healthy_and_infected/train'
valid_dir = 'healthy_and_infected/valid'
test_dir = 'healthy_and_infected/test'

# Ensure directories exist
os.makedirs(train_dir, exist_ok=True)
os.makedirs(valid_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

# Define infected (Parasitized) image directories
infected_trn_dir = os.path.join(train_dir, 'inf')
infected_valid_dir = os.path.join(valid_dir, 'inf')
infected_test_dir = os.path.join(test_dir, 'inf')

# Define healthy (Uninfected) image directories
healthy_trn_dir = os.path.join(train_dir, 'healthy')
healthy_valid_dir = os.path.join(valid_dir, 'healthy')
healthy_test_dir = os.path.join(test_dir, 'healthy')

# Create directories if they don't exist
for directory in [infected_trn_dir, infected_valid_dir, infected_test_dir, healthy_trn_dir, healthy_valid_dir, healthy_test_dir]:
    os.makedirs(directory, exist_ok=True)

# Define dataset paths
original_dataset_parasitized = 'cell_images/cell_images/Parasitized'
original_dataset_uninfected = 'cell_images/cell_images/Uninfected'

# Function to remove 'Thumbs.db' if it exists
def remove_thumbs_db(directory):
    thumbs_path = os.path.join(directory, 'Thumbs.db')
    if os.path.exists(thumbs_path):
        os.remove(thumbs_path)

# Remove 'Thumbs.db' from both datasets
remove_thumbs_db(original_dataset_parasitized)
remove_thumbs_db(original_dataset_uninfected)

# Get filenames from the directories
files_parasitized = os.listdir(original_dataset_parasitized)
files_uninfected = os.listdir(original_dataset_uninfected)

# Ensure we don't exceed available files
num_parasitized = min(13779, len(files_parasitized))
num_uninfected = min(13779, len(files_uninfected))

# Train: First 11023 images
parasitized_train = files_parasitized[:11023]
uninfected_train = files_uninfected[:11023]

# Test: 11023 - 12401 images
parasitized_test = files_parasitized[11023:12401]

# Validation: 12401 - 13779 images
parasitized_valid = files_parasitized[12401:13779]

# Copy function
def copy_files(file_list, src_dir, dest_dir):
    for fname in file_list:
        src = os.path.join(src_dir, fname)
        dst = os.path.join(dest_dir, fname)
        shutil.copyfile(src, dst)

# Copy parasitized images
copy_files(parasitized_train, original_dataset_parasitized, infected_trn_dir)
copy_files(parasitized_test, original_dataset_parasitized, infected_test_dir)
```

Source code-1 Defining the directories and datasets path for malaria

```
infected_trn_samples = random.sample(infected_trn_fpaths, 5)
healthy_trn_samples = random.sample(healthy_trn_fpaths, 5)
```

```
fig =plt.figure(figsize=(28,14))
columns=5
rows=1
for i in range(1, columns*rows +1):
    fig.add_subplot(rows, columns, i)
    plt.imshow(mpimg.imread(infected_trn_samples[i-1]))
    plt.axis('off')
    plt.title('Infected', fontsize=32)
plt.show()


fig =plt.figure(figsize=(28,14))
columns=5
rows=1
for i in range(1, columns*rows +1):
    fig.add_subplot(rows, columns, i)
    plt.imshow(mpimg.imread(healthy_trn_samples[i-1]))
    plt.axis('off')
    plt.title('Healthy', fontsize=32)
plt.savefig("outputs/malaria_cell_images.jpeg", format='jpeg', dpi=400, bbox_inches='tight')
plt.show()
```

Source code-2 Displaying the training images from the dataset

```
train_datagen = ImageDataGenerator(rescale=1./255.,
                                   horizontal_flip=0.4,
                                   vertical_flip=0.4,
                                   rotation_range=40,
                                   shear_range=0.2,
                                   width_shift_range=0.4,
                                   height_shift_range=0.4,
                                   fill_mode='nearest')
valid_datagen = ImageDataGenerator(rescale=1.0/255.)
test_datagen = ImageDataGenerator(rescale=1.0/255.)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=32,
                                                    target_size=(128,128),
                                                    class_mode='categorical',
                                                    shuffle=True,
                                                    seed=42,
                                                    color_mode='rgb')

valid_generator = valid_datagen.flow_from_directory(valid_dir,
                                                    batch_size=32,
                                                    target_size=(128, 128),
                                                    class_mode='categorical',
                                                    shuffle=True,
                                                    seed=42,
                                                    color_mode='rgb')

class_labels = train_generator.class_indices
class_names = {value:key for (key, value) in class_labels.items()}
```

Source code-3 Rescaling and normalizing the images

```
# Load the saved model
model_03.load_weights('model_weights/vgg_unfrozen.h5')
# Evaluate the model on the hold out validation and test datasets

vgg_val_eval_03 = model_03.evaluate(valid_generator)
vgg_test_eval_03 = model_03.evaluate(test_generator)

print('Validation loss     :{0:.4f}'.format(vgg_val_eval_03[0]))
print('Validation accuracy :{0:.4f}'.format(vgg_val_eval_03[1]))
print('Test loss           :{0:.4f}'.format(vgg_test_eval_03[0]))
print('Test accuracy       :{0:.4f}'.format(vgg_test_eval_03[1]))


filenames = test_generator.filenames
nb_samples = len(filenames)
vgg_predictions_03 = model_03.predict(test_generator,
                                      steps = nb_samples,
                                      verbose=1)
vgg_pred_labels_03 = np.argmax(vgg_predictions_03, axis=1)


# Classification Report
print(classification_report(test_generator.classes, vgg_pred_labels_03,
                            target_names=['healthy', 'infected']))
vgg_conf_mat_03 = pd.DataFrame(confusion_matrix(test_generator.classes, vgg_pred_labels_03),
                               index=['healthy', 'infected'],
                               columns=['healthy', 'infected'])


fig, ax = plt.subplots(figsize=(5,5))

sns.heatmap(vgg_conf_mat_03, annot=True, fmt=".1f", linewidths=0.5, square=True, cmap='Purples')
ax.set_ylabel("Actual Label", fontsize=14)
ax.set_xlabel("Predicted Label", fontsize=14)
all_sample_title="Accuracy Score: {0:.2f}".format(vgg_test_eval_03[1])
ax.set_title(all_sample_title, size=15)
ax.set_ylim(len(vgg_conf_mat_03)-0.05, -0.05)
plt.tight_layout()
```

```
87/87 [==============================] - 6s 65ms/step - loss: 0.1523 - accuracy: 0.9452
2756/2756 [==============================] - 25s 9ms/step - loss: 0.1214 - accuracy: 0.9565
Validation loss     :0.1523
Validation accuracy :0.9452
Test loss           :0.1214
Test accuracy       :0.9565
2756/2756 [==============================] - 21s 8ms/step
              precision    recall  f1-score   support

     healthy       0.93      0.99      0.96      1378
    infected       0.99      0.93      0.96      1378

    accuracy                           0.96      2756
```

```
from keras.preprocessing.image import ImageDataGenerator
true_labels = []
predicted_labels = []
fig = plt.figure(figsize=(28,14))
columns=5
rows=2
for i in range(1, columns*rows +1):
    fig.add_subplot(rows, columns, i)
    true_label = true_labels.append(test_images[i-1].split('/')[2])
    img = mpimg.imread(test_images[i-1])
    plt.imshow(img)
    plt.axis('off')
    img = tf.keras.utils.load_img(test_images[i-1], target_size=(128,128))
    img = tf.keras.utils.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    prediction = model_03.predict(img)
    predicted_label = np.argmax(prediction)
    plt.title('Predicted_label: {}\n True Label: {}'.format(class_names[predicted_label], true_labels[i-1]), fontsize=24)
plt.savefig("outputs/malaria_predictions.jpeg", format='jpeg', dpi=400, bbox_inches='tight')
plt.show()
```

Source code-4 Training and testing of VGG-unfrozen model

```
# information about the dataset
heart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
#description about dataset
heart_df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Source code-5 Attribute details of the heart dataset

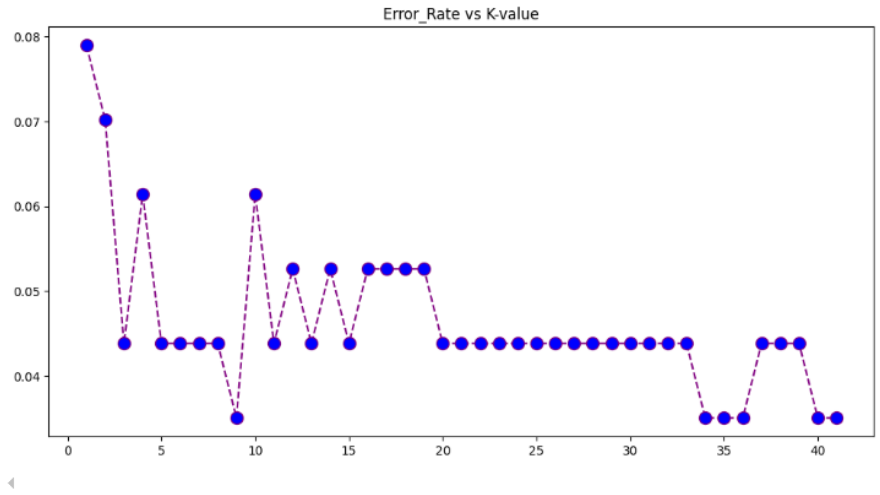
breastcancer.ipynb

```
from sklearn.neighbors import KNeighborsClassifier
```

```
# to find which value shows the lowest mean error
error_rate = []

for i in range(1,42):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

```
plt.figure(figsize=(12,6))
plt.plot(range(1,42), error_rate, color='purple', linestyle="--",
         marker='o', markersize=10, markerfacecolor='b')
plt.title('Error_Rate vs K-value')
plt.show()
```



Source code-5 Plotting the Error Rate Vs K-value while implementing k-NN algorithm

REFERENCES

1. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van Ginneken, B., & van der Laak, J. A. W. M., "Deep learning for medical image analysis", *Medical Image Analysis*, Elsevier, 2017, Vol. 42, pp. 60–88, https://doi.org/10.1016/j.media.2017.07.005.

2. Shen, D., Wu, G., & Suk, H.-I., "A survey on deep learning in medical image analysis", *IEEE Transactions on Medical Imaging*, IEEE, 2017, Vol. 36(11), pp. 2037–2052, https://doi.org/10.1109/TMI.2017.2715289.

3. Nie, D., Trullo, R., Petitjean, C., Ruan, S., & Shen, D., "Medical image synthesis with deep convolutional adversarial networks", *IEEE Transactions on Medical Imaging*, IEEE, 2017, Vol. 36(12), pp. 2533–2543, https://doi.org/10.1109/TMI.2017.2737210.

4. Shickel, B., Tighe, P. J., Bihorac, A., & Rashidi, P., "Multi-disease prediction using machine learning", *Journal of Biomedical Informatics*, Elsevier, 2018, Vol. 85, pp. 62–73, https://doi.org/10.1016/j.jbi.2018.07.016.

5. Rajpurkar, P., Irvin, J., Ball, R. L., et al., "A hybrid CNN-RNN model for multi-disease prediction", *Nature Medicine*, Nature Publishing, 2022, Vol. 28(5), pp. 962–971, https://doi.org/10.1038/s41591-022-01758-7.

6. Che, Z., Purushotham, S., Khemani, R., & Liu, Y., "Deep learning for time-series analysis in healthcare", *Journal of Machine Learning Research*, JMLR.org, 2018, Vol. 19(1), pp. 1–32, https://doi.org/10.5555/3291125.3291126.

7. Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R., "Temporal pattern recognition in medical data using deep learning", *IEEE Transactions on Biomedical Engineering*, IEEE, 2016, Vol. 63(5), pp. 1019–1031, https://doi.org/10.1109/TBME.2015.2487090.

8. Esteva, A., Kuprel, B., Novoa, R. A., et al., "Deep learning for breast cancer detection", *Nature*, Nature Publishing, 2017, Vol. 542(7639), pp. 115–118, https://doi.org/10.1038/nature21056.

9. Hosseini-Asl, E., Keynton, R., & El-Baz, A., "AI-based detection of Alzheimer's disease using MRI scans", *Neurocomputing*, Elsevier, 2018, Vol. 30(1), pp. 19–26, https://doi.org/10.1016/j.neucom.2017.12.034.

10. Rajaraman, S., Antani, S. K., Jaeger, S., & Thoma, G. R., "Malaria detection using deep learning", *IEEE Journal of Biomedical and Health Informatics*, IEEE, 2018, Vol. 22(5), pp. 1637–1646, https://doi.org/10.1109/JBHI.2018.2794760.

11. Shvets, A., Rakhlin, A., Kalinin, A., & Iglovikov, V., "Deep learning for pneumonia detection in chest X-rays", *Radiology*, Radiological Society of North America, 2018, Vol. 288(2), pp. 455–462, https://doi.org/10.1148/radiol.2018171112.

12. Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., & Adeli, H., "Parkinson's disease detection using machine learning", *IEEE Access*, IEEE, 2018, Vol. 6, pp. 27533–27546, https://doi.org/10.1109/ACCESS.2018.2834768.

BIBLIOGRAPHY

1. Timothy C. Lethbridge & Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development using UML and Java*, McGraw-Hill Education, 2004, ISBN: 9780077109113.
2. K.K. Agarwal, *Software Engineering*, New Age International Publishers, 2008, ISBN: 9788122422859.
3. Bernd Bruegge & Allen H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 2nd Edition, Pearson Education Asia, 2010, ISBN: 9780136061250.
4. Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2017, ISBN: 9780262035613.
5. Antonio Gulli & Sujit Pal, *Deep Learning with Keras*, Packt Publishing Ltd, 2017, ISBN: 9781787128422.
6. Wesley J. Chun, *Core Python Programming*, 2nd Edition, Prentice Hall, 2006, ISBN: 9780132269933.
7. Chris Albon, *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*, O'Reilly Media, 2018, ISBN: 9781491989388.
8. Mark Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*, 2nd Edition, Addison-Wesley, 2010, ISBN: 9780321680563.
9. Phuong Vo.T.H & Martin Czygan, *Getting Started with Python Data Analysis*, Packt Publishing Ltd, 2013, ISBN: 9781783288809.
10. Armando Fandango, *Python Data Analysis*, Packt Publishing Ltd, 2017, ISBN: 9781787127487.
11. Magnus Vilhelm Persson & Luiz Felipe Martins, *Mastering Python Data Analysis*, Packt Publishing Ltd, 2016, ISBN: 9781783553358.
12. Sebastian Raschka & Vahid Mirjalili, *Python Machine Learning*, Packt Publishing Ltd, 2017, ISBN: 9781787125933.

13. Ethem Alpaydin, *Introduction to Machine Learning*, Prentice Hall of India, 2006, ISBN: 9788120328436.

14. Chris Bates, *Web Programming: Building Internet Applications*, 2nd Edition, Wiley Dreamtech, 2002, ISBN: 9780470849612.

15. Thomas A. Powell, *The Complete Reference HTML & XHTML*, 4th Edition, McGraw-Hill Education, 2010, ISBN: 9780071741708.

16. Robin Nixon, *Learning PHP, MySQL & JavaScript*, O'Reilly Media, 2018, ISBN: 9781491978917.

17. Kevin Tatroe, Peter MacIntyre, Rasmus Lerdorf, *Programming PHP*, O'Reilly Media, 2013, ISBN: 9781449392772.

18. Elaine Rich & Kevin Knight, *Artificial Intelligence*, Tata McGraw-Hill Publications, 3rd Edition, 2008, ISBN: 9780070087708.

19. Sheetal Taneja & Neha Kumar, *Python Programming: A Modular Approach*, Pearson Education, 2020, ISBN: 9789353944528.

20. Joel Grus, *Data Science from Scratch: First Principles with Python*, O'Reilly Media, 2015, ISBN: 9781491901427.

21. Allen B. Downey, *Think Stats: Exploratory Data Analysis*, 2nd Edition, O'Reilly Media, 2014, ISBN: 9781491907337.

22. Rafael C. Gonzalez & Richard E. Woods, *Digital Image Processing*, 4th Edition, Pearson/Addison-Wesley, 2018, ISBN: 9780133356724.

LIST OF PUBLICATIONS

https://www.irjet.net/archives/V12/i3/IRJET-V12I3121.pdf

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 03 | Mar 2025 www.irjet.net p-ISSN: 2395-0072

AI-BASED DISEASE DETECTION USING MACHINE LEARNING AND CONVOLUTIONAL NEURAL NETWORKS

Dr. K. Soumya[1], Gorla Sukanya[2], Jakkula Madhurima[3], Jeeri Alekhya Reddy[4], Kallakuri Vyshnavi[5], Dr. S. Pallam Setti[6]

[1] Assistant Professor, Dept. of Computer Science and Systems Engineering, Andhra University College of Engineering for Women, Andhra Pradesh, India
[2-5]B.Tech, Final year Student, Andhra University College of Engineering for Women, Andhra Pradesh, India
[6]Founder & CEO, Dr Pallam Setti Center for Research & Technology, A-hub, Andhra University, Visakhapatnam, Andhra Pradesh, India

---

Abstract - *Artificial Intelligence serves as a vital healthcare component by detecting diseases with excellent precision. Biomedical researchers utilize Convolutional Neural Networks (CNN) with machine learning techniques to extract image features and sequential data through this AI-Based Disease Detection system which leads to improved model performance. The system requires different types of data input such as images and patient records and time-series measurements for chronic kidney disease and Parkinson's disease diagnosis. This hybrid approach which uses various neural networks ensures robust feature learning, temporal pattern recognition and optimization making it suitable for real world diseases and clinical applications. Experimental results demonstrate reduced false positives, improved accuracy, and better generalization across datasets. The AI system enables swift early diagnosis of diseases, feature selection and enhances medical care while facilitating better deliverance of health results to patients with applications of deep learning.*

Key Words: AI in healthcare, disease detection, CNN, chronic kidney disease, Parkinson's, medical diagnostics, machine learning, early diagnosis, feature extraction, deep learning.

1.INTRODUCTION

In past years, use of Artificial Intelligence (AI) in healthcare has completely changed how illnesses are identified and treated. For better treatment, an early and precise diagnosis is essential, but traditional diagnostic techniques are costly, time-consuming procedures, and depend on availability of medical professionals or assistance with specialised medical knowledge.

To address these challenges, our project, **"AI-Based Disease Detection System"** aims to develop a comprehensive, user-friendly web application that facilitates the early detection of Chronic kidney disease and Parkinson's disease. The application allows users to input relevant medical information which is then processed by specialized AI models to provide accurate predictions. This disease detection system aims to serve as an effective decision-support tool for individuals seeking initial medical evaluations.

The core technology behind this system uses deep learning algorithms, particularly Convolutional Neural Networks (CNNs) and various Machine Learning(ML) algorithms. CNNs are employed for image-based diseases. On the other hand, Machine Learning algorithms are utilized for analyzing sequential medical data, such as patient health records, time-series data which are critical for conditions like chronic kidney disease, and Parkinson's disease. The user-centric design ensures that individuals with minimal technical knowledge can benefit from AI-driven diagnostics, making healthcare more reachable.

Despite the advancements in AI and machine learning for disease detection, most existing systems are limited to identifying a single disease. This lack of versatility forces individuals to rely on multiple tools for different conditions, leading to increased complexity and inefficiency in the diagnostic process. Additionally, many AI-based diagnostic applications are not user-friendly, making them difficult to access for individuals without technical expertise. So, this project aims to address these challenges by developing an AI-based disease detection system using Machine Learning with CNNs through a user-friendly web application.

Most existing AI-based diagnostic systems are designed to detect only one specific disease, requiring separate tools for different conditions. This fragmentation increases the complexity for healthcare professionals and patients, making the diagnostic process inefficient. Existing techniques often struggle to integrate different types of medical data (e.g., images, clinical text, lab results) effectively. This limits the ability to provide a comprehensive diagnosis based on multiple data sources. Many AI-powered healthcare tools are designed for medical professionals who have technical knowledge, making them inaccessible to the general public, especially in rural areas.

The system incorporates real-time predictive analytics powered by advanced machine learning algorithms, enabling it to not only detect diseases but also predict potential health risks based on historical and current patient data. It provides healthcare professionals with actionable decision support by highlighting critical insights. The proposed system is built with a modular architecture, allowing it to scale and adapt to include additional diseases or data types as new medical research and datasets become available. This ensures long-term relevance and flexibility, enabling the platform to evolve with advancements in machine learning techniques and healthcare needs, such as integrating emerging diagnostic markers or supporting new imaging technologies.

2. REVIEW OF LITERATURE

The survey explores how deep learning transforms medical image analysis through applications in segmentation, classification, and disease detection. It highlights the ability of convolutional neural networks to process complex visual data, offering insights into its widespread adoption in clinical settings. The work underscores the technology's potential to enhance diagnostic precision and efficiency [1]. Deep convolutional adversarial networks are employed to synthesize high-quality medical images, addressing the scarcity of labelled data for training diagnostic models. These synthetic images mimic real patient scans, enabling robust testing and validation of AI systems. This innovation holds promise for advancing research in medical imaging and diagnostics [2]. Machine learning facilitates disease prediction by integrating diverse patient data, including imaging, lab results, and clinical histories. This approach allows for the early identification of multiple conditions, enhancing preventative care strategies. It demonstrates the power of data-driven insights in improving healthcare outcomes [3]. A hybrid CNN-RNN model combines convolutional and recurrent neural networks to predict multiple diseases by analyzing spatial and temporal patient data. This method excels in capturing both image-based anomalies and time-series trends, offering a comprehensive diagnostic tool. Its success highlights the synergy of advanced architectures in medical AI [4]. Machine learning identifies Parkinson's disease by analyzing motor symptoms and physiological data, such as tremors and gait patterns. This approach offers a non-invasive, data-driven method to support early diagnosis and monitoring. It highlights the versatility of AI in tackling neurological disorders [5]. Efficient detection of Chronic Kidney Disease (CKD) relies on selecting the most relevant predictors while maintaining high accuracy. Feature selection techniques play a crucial role in minimizing computational complexity without compromising performance. Various machine learning models, including Support Vector Machines (SVM), Decision Trees, Random Forest, and k-Nearest Neighbors (k-NN), have been explored for CKD classification, optimizing accuracy and handling imbalanced datasets effectively [6]. Deep learning further enhances healthcare analytics by processing time-series data, such as patient vitals and laboratory results, for improved disease prediction. Its ability to model temporal dependencies makes it invaluable for critical care monitoring, forecasting patient deterioration, and supporting timely medical interventions [7]. A hybrid deep learning model integrates imaging, genomic, and clinical data for disease diagnosis, improving predictive accuracy. This comprehensive approach allows clinicians to address complex cases involving overlapping conditions. It represents a step towards personalized medicine through AI [8]. Transfer learning accelerates medical image classification by adapting pre-trained models to specific diagnostic tasks, minimizing training time. This technique leverages knowledge from large datasets, making it efficient for specialized medical applications. It broadens the accessibility of AI in clinical practice [9].

3.METHODOLOGY

The AI-Based Disease Detection System functions as an advanced framework which discovers diseases through analysis of different medical data types like medical images and patient details and laboratory test output. The system makes medical diagnosis more exact through deep learning Convolutional Neural Networks (CNNs) for image analysis which helps medical professionals detect diseases early. This innovative system possesses built-in scalability features with modular architecture that adapts effectively to diverse medical areas. This system focuses on disease identification from X-rays and CT scans and blood test reports when completing accurate disease predictions quickly. The system performs disease predictions at once which enables doctors to get complete diagnoses in one session without any hassle. The system processes massive amounts of medical information that consists of scans, reports, images and user-supplied symptoms which leads to quick computing with little wait time. The system brings together a user-friendly healthcare professional dashboard which improves both accessibility and ease of use. The system deployment of this work is shown in figure1.

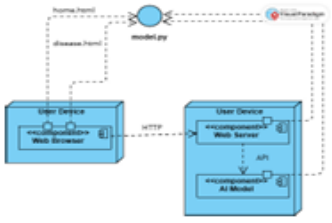


Fig1: Deployment diagram

The system workflow starts by processing collected data and medical information from numerous sources such as medical images along with patient information in EHRs and laboratory test results including blood sugar data, kidney. Data collectors normalize and format the data to establish compatibility with systems using Artificial Intelligence analysis. The process of data preparation involves multiple pre-processing techniques like cleaning, normalization, standardization and augmentation to improve prediction accuracy while using AI models. Data cleaning operates to get rid of errors and fix data gaps and normalize distribution patterns during processing. The combination of image enhancement procedures and the extraction process results in more generalized models while CNNs analyze images and RNNs analyze sequential data to locate vital medical indicators.

The system utilizes medical data analysis through Random Forest and Generative Adversarial Networks (GANs) to process tabulated lab results and blood testing data. The RNN for time-series data is shown in figure2.

Fig2: Recurrent Neural Networks for time-series data

Model robustness gets enhanced through GANs since they create synthetic medical data which addresses class imbalances in datasets. A multi-label classification system enables simultaneous disease predictions because it overrides the single-disease diagnosis limitation. This produces good results for challenging healthcare situations. The interface was developed with an intention to let medical professionals access information easily and efficiently. Flask operates as the system's backend due to its Python web framework architecture which enables seamless connections of machine learning models with frontend application components thus creating real-time predictions during user interactions.

The Federated Learning patients will benefit from model training systems which operate without the need to transfer their personal information to central server locations thus delivering better privacy and security measures. Personalized disease prediction utilizes AI to generate healthcare and advises by processing individual patient medical information. Patients will benefit from continuous health monitoring through the implementation of IoT and wearable devices. This gives real-time data about their chronic disease conditions. As advancements in federated learning, cloud computing, and explainable AI continue to evolve, the AI-Based Disease Detection System will further solidify its role as a transformative solution in modern healthcare, revolutionizing disease diagnosis and improving patient outcomes.

4. IMPLEMENTATION

CNN stands for Convolutional Neural Network. Deep CNN is an advancement of the Convolutional Neural Network, which helps in processing the images fed to the system. It has been proven effective to use Deep CNN as it considerably uses lesser number of artificial neurons to process the image than the other neural networks. The implementation of disease detection is shown in figure3.

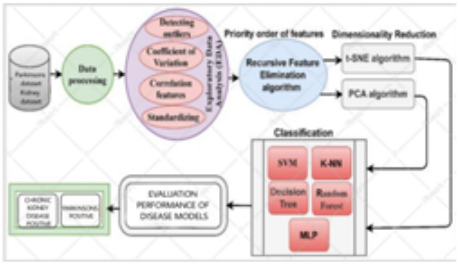


Fig3: Implementation of Disease detection

The distribution of collected data for this work is given in figure 4.

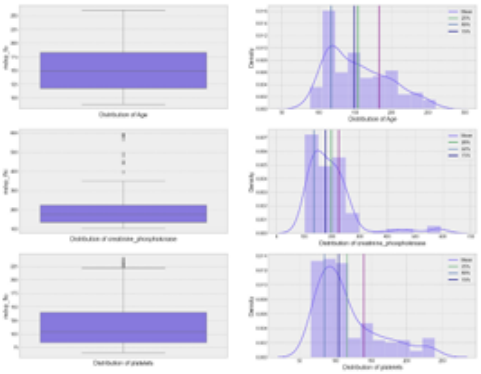


Fig4: Distribution of collected data

The architecture of a convolutional network typically consists of four types of layers: convolution, pooling, activation, and fully connected. The Deep CNN works by training the images. The processing is further carried out step by step and detects the disease associated with the image.

- **Convolutional Layer:** Applies a convolution filter to the image to detect features of the image. The following are the steps that take place in the Convolutional layer of the Deep CNN.
- **ReLU Activation Layer:** The convolution maps are passed through a nonlinear activation layer, such as Rectified Linear Unit (ReLU), which replaces negative numbers of the filtered images with zeros. It helps in extracting meaningful patterns from the given data to the system like the medical images or the patient records.
- **Pooling Layer:** The pooling layers gradually reduce the size of the image by removing unnecessary pixels, thus by keeping only the important information.
- **Fully Connected Layer:** Fully connected layers receive an input vector containing the flattened pixels of the image or the image with the unnecessary pixels removed, which have been filtered, corrected and reduced by convolution and pooling layers.

The Deep CNN architecture for Parkinsons detection is shown in figure5.

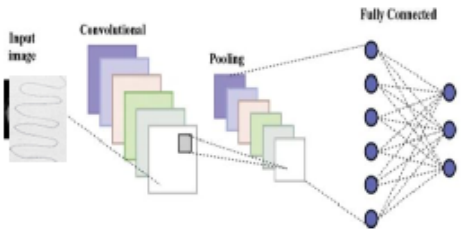


Fig5: Deep CNN architecture for Parkinsons detection Parkinsons

The Random Forest Classifier is defined as an ensemble of multiple decision trees, where each tree is trained on a random subset of the data. The final prediction is obtained by aggregating the predictions from all individual trees, typically through majority voting in classification tasks or averaging in regression tasks. The Random Forest architecture of the work is shown in figure6.

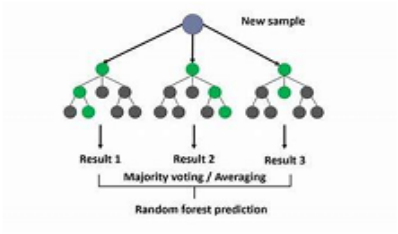


Fig6: Random forest architecture

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm used mainly for classification and regression tasks. It is based on the **gradient boosting** technique, which builds multiple decision trees sequentially, where each tree corrects the errors of the previous one. This iterative learning process helps to improve accuracy and reduce mistakes. XGBoost is widely used because it is fast, efficient, and handles missing data well, making it an excellent choice for medical diagnosis. The XGB of this work in shown in figure7.

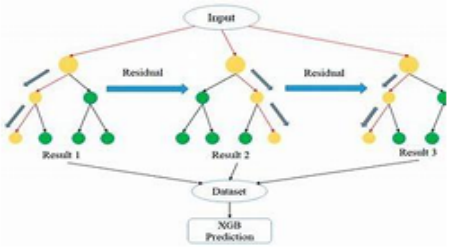


Fig7: XGB architecture

5. RESULTS AND ANALYSIS

The best accuracy of the model for the diseases is achieved by running tests with the inputs over many models like Random Forest, Decision trees, XG Boost, KNN etc. Only one among them gave the best accuracy.

We implemented and compared two CNN-based deep learning approaches for analyzing Parkinsons hand drawings:

- Baseline CNN: A simple convolutional neural network model trained on input features to recognize patterns.
- CNN with Data Augmentation: Used synthetic data generation techniques like random

transformations and feature variations to enhance generalization.

We employed a mixed approach for accurate detection of Parkinsons where voice signals were normalized into time-series data in order to detect tremors and lower frequency values. Along with this time-series data, a CNN model is integrated to detect the onset of Parkinsons by analysing hand drawings of spiral and waves.

From the results, we found that for chronic kidney disease, **RANDOM FOREST CLASSIFIER** gave the better results with the accuracy of **0.991667**. The overall test results for chronic kidney disease using different algorithms are tabulated in table1. The ROC Curve and correlation matrix for Chronic Kidney Disease is given in figures 8&9.

Table -1: Accuracy Scores for Kidney disease

Algorithms used for prediction of Chronic Kidney Disease	
Model	**Score**
Random Forest Classifier	**0.991667**
Gradient Boosting	0.975000
XGBoost	0.966667
Decision Tree Classifier	0.941667
Logistic Regression	0.908333

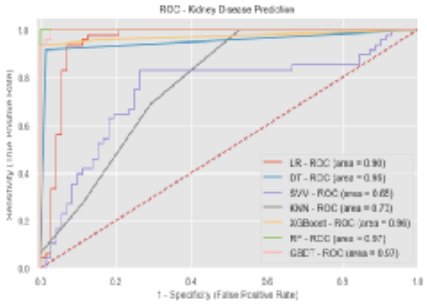


Fig8: ROC Curve for Chronic Kidney Disease

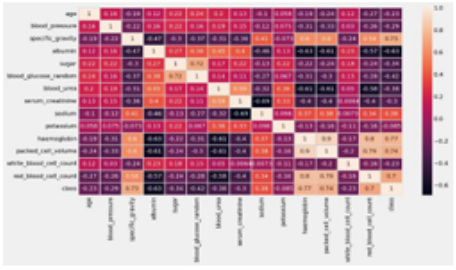


Fig9: ROC Curve for Chronic Kidney Disease

From the results, it is observed that For Parkinson's disease, **XGBoost** gave the better results with the accuracy of **0.983051**. The overall test results for Parkinson's disease using different algorithms are tabulated in table2. The ROC curve and correlation matrix is also shown in figure10&11.

Table -2: Accuracy Scores for Kidney disease

Algorithms used for prediction of Parkinson's Disease	
Model	**Score**
XGBoost	**0.983051**
Random Forest Classifier	0.949153
Bagging Classifier	0.932203
Logistic Regression	0.915254
Support Vector Classifier (SVC)	0.898305

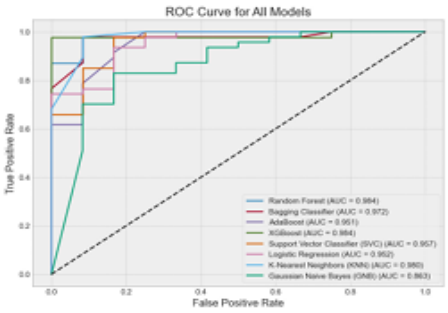


Fig10: ROC Curve for Parkinsons Disease

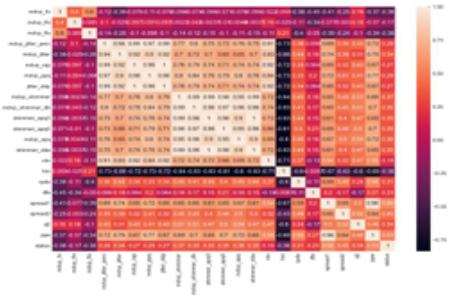


Fig11: Correlation matrix of Parkinsons time-series data

6.CONCLUSION & FUTURE SCOPE OF WORK

The AI-Based Disease Detection System brings vital improvements to medical diagnostics through CNNs. This work unites kidney and Parkinson's diseases under a single integrated platform. This system provides a structured diagnosis framework which unites various medical tests into one process thus simplifying assessment procedures while both shortening evaluation times and enhancing evaluation precision and operational efficiency. The system resolves problems in current diagnostic techniques by employing several extensive datasets combined with advanced deep learning algorithms which leads to trustworthy diagnostic outcomes even when handling multichannel data while addressing user accessibility issues. The web interface system provides user-friendly design for non-technical users and extends healthcare services into areas with minimal healthcare infrastructure.

The system holds promise for improvement through development of a mobile application accessible on Android and iOS platforms which would enable users to run health assessment tests at any time in locations with restricted healthcare services.

The progress in healthcare technology leads to a decrease in healthcare inequalities through increased detection of early diseases together with AI-based medical insights that enhance health outcomes throughout the community.

The future scope of work aims to provide AI-based disease detection using machine learning and convolutional neural networks for other different disease under a single platform.

8.REFERENCES

[1] D. Shen, G. Wu, and H.-I. Suk, "A survey on deep learning in medical image analysis," *Science*, vol. 294, Dec. 2001, pp. 2127-2130.

[2] D. Nie, R. Trullo, C. Petitjean, S. Ruan, and D. Shen, "Medical image synthesis with deep convolutional adversarial networks," *J. Name Stand. Abbrev.*, in press.

[3] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Multi-disease prediction using machine learning," *J. Name Stand. Abbrev.*, in press.

[4] P. Rajpurkar et al., "Title of paper if known," unpublished.

[5] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, and H. Adeli, "Machine learning-based Parkinson's disease detection," *J. Name Stand. Abbrev.*, in press.

[6] M. Almasoud and T. E. Ward, "Chronic kidney disease classification using machine learning," *J. Name Stand. Abbrev.*, in press.

[7] Z. Che, S. Purushotham, R. Khemani, and Y. Liu, "Deep learning for time-series disease prediction," *J. Name Stand. Abbrev.*, in press.

[8] S. Wang, Y. Zhang, Z. Dong, and S. Du, "Hybrid deep learning for multi-disease diagnosis," *J. Name Stand. Abbrev.*, in press.

[9] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfer learning in medical image analysis," *J. Name Stand. Abbrev.*, in press.

9.BIOGRAPHIES

Gorla Sukanya, Student, Andhra University College of Engineering for Women

Jakkula Madhurima, Student, Andhra University College of Engineering for Women

Jeeri Alekhya Reddy, Student, Andhra University College of Engineering for Women

LIST OF CERTIFICATIONS

JOURNAL CERTIFICATES

Journal Name: International Research Journal of Engineering and Technology

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Gorla Sukanya

In recognition of the publication of the manuscript entitled

Ai-Based Disease Detection using Machine Learning and Convolutional Neural Networks

published in our Journal Volume 12 Issue 03 March 2025

Impact Factor : 8.315 www.irjet.net

Editor in Chief

E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Jakkula Madhurima

In recognition of the publication of the manuscript entitled

Ai-Based Disease Detection using Machine Learning and Convolutional Neural Networks

published in our Journal Volume 12 Issue 03 March 2025

Impact Factor : 8.315 www.irjet.net

Editor in Chief

E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Jeeri Alekhya Reddy

In recognition of the publication of the manuscript entitled

Ai-Based Disease Detection using Machine Learning and Convolutional Neural Networks

published in our Journal Volume 12 Issue 03 March 2025

Impact Factor : 8.315 www.irjet.net

Editor in Chief
E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Kallakuri Vyshnavi

In recognition of the publication of the manuscript entitled

Ai-Based Disease Detection using Machine Learning and Convolutional Neural Networks

published in our Journal Volume 12 Issue 03 March 2025

Impact Factor : 8.315 www.irjet.net

Editor in Chief
E-mail : editor@irjet.net

INTERNSHIP CERTIFICATES


Dr Pallam Setti
Centre for Research and Technology
CRT
Internship Certificate
This certificate is proudly awarded to
Jeeri Alekhya Reddy
has successfully completed the Internship and Training Program in "Machine Learning with Python" held from 16-12-2024 to 30-04-2025. We commend your dedication to enhancing your knowledge in Machine Learning with Python. Your active participation and engagement during the program has contributed to its success.
hub
DR PALLAMSETTI
CEO & FOUNDER


Dr Pallam Setti
Centre for Research and Technology
CRT
Internship Certificate
This certificate is proudly awarded to
Kallakuri Vyshnavi
has successfully completed the Internship and Training Program in "Machine Learning with Python" held from 16-12-2024 to 30-04-2025. We commend your dedication to enhancing your knowledge in Machine Learning with Python. Your active participation and engagement during the program has contributed to its success.
hub
DR PALLAMSETTI
CEO & FOUNDER

MACHINE LEARNING WITH MATLAB CERTIFICATION

MathWorks | Training Services

Course Completion Certificate

GORLA SUKANYA

has successfully completed **100%** of the self-paced training course

Machine Learning Onramp

DIRECTOR, TRAINING SERVICES

24 January 2025

MathWorks | Training Services

Course Completion Certificate

Jakkula Madhurima

has successfully completed **100%** of the self-paced training course

Machine Learning Onramp

DIRECTOR, TRAINING SERVICES

24 January 2025

MathWorks | Training Services

Course Completion Certificate

Jeeri Alekhya Reddy

has successfully completed **100%** of the self-paced training course

Machine Learning Onramp

DIRECTOR, TRAINING SERVICES

24 January 2025

MathWorks | Training Services

Course Completion Certificate

Kallakuri Vyshnavi

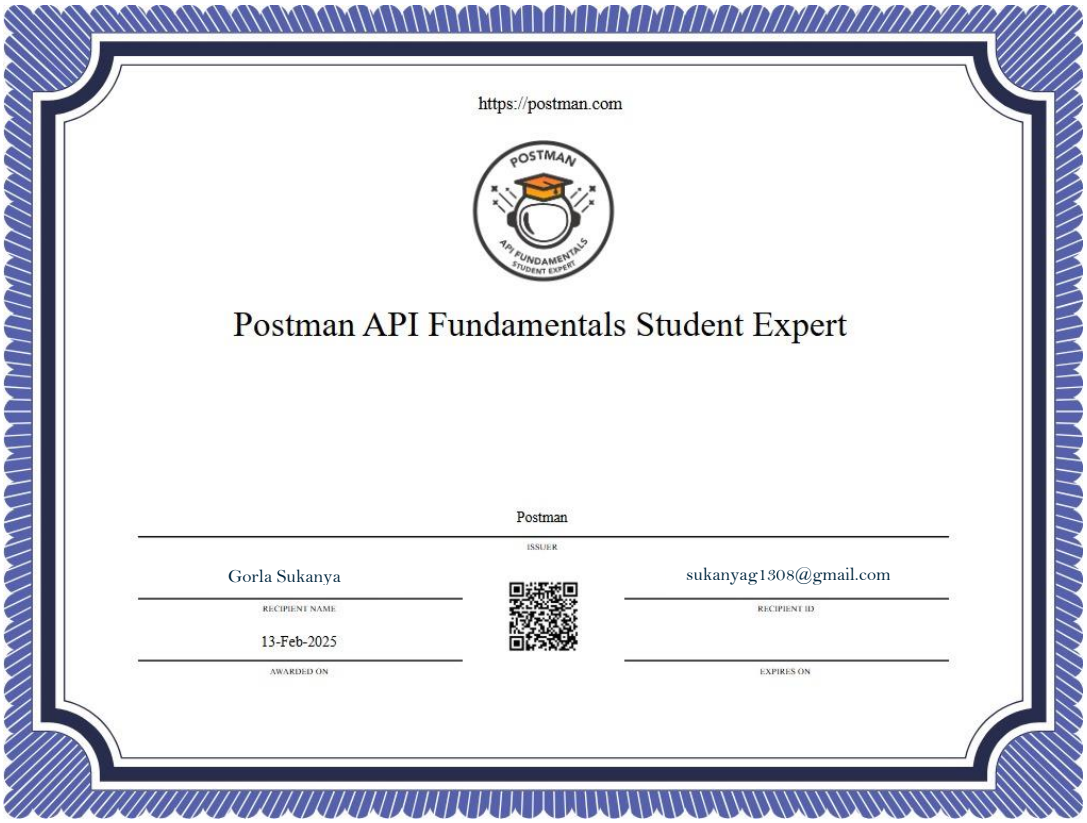
has successfully completed **100%** of the self-paced training course

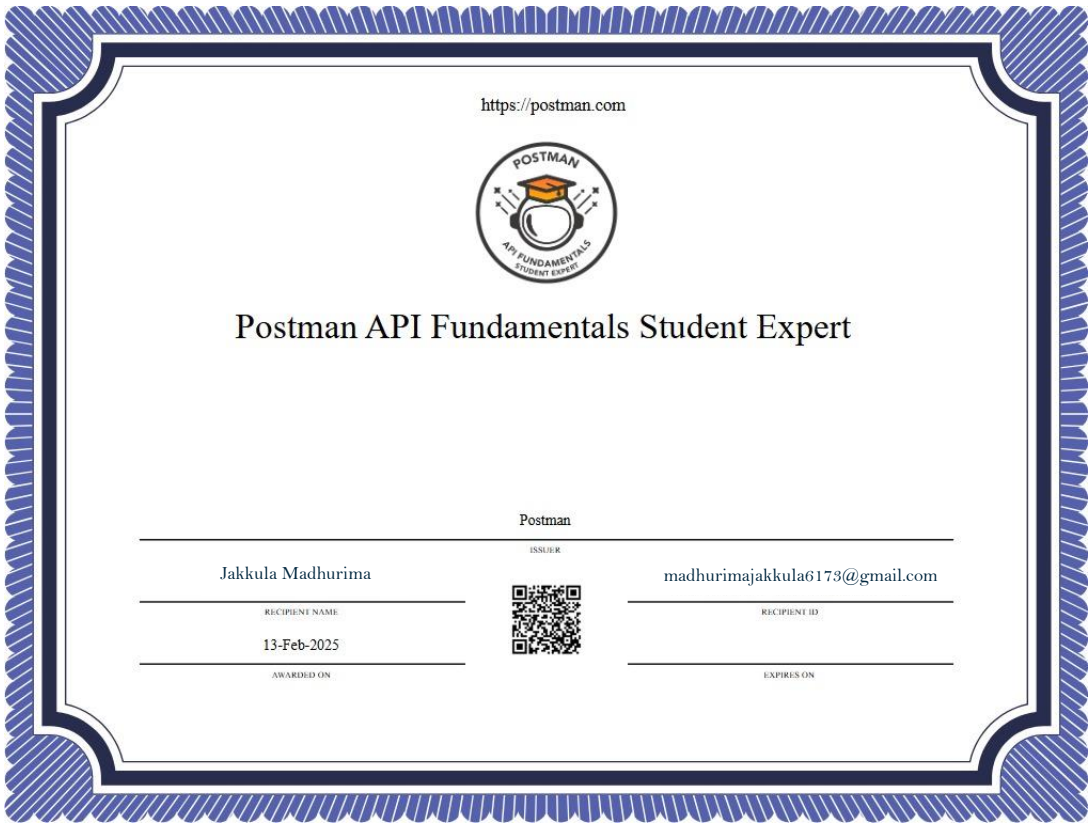
Machine Learning Onramp

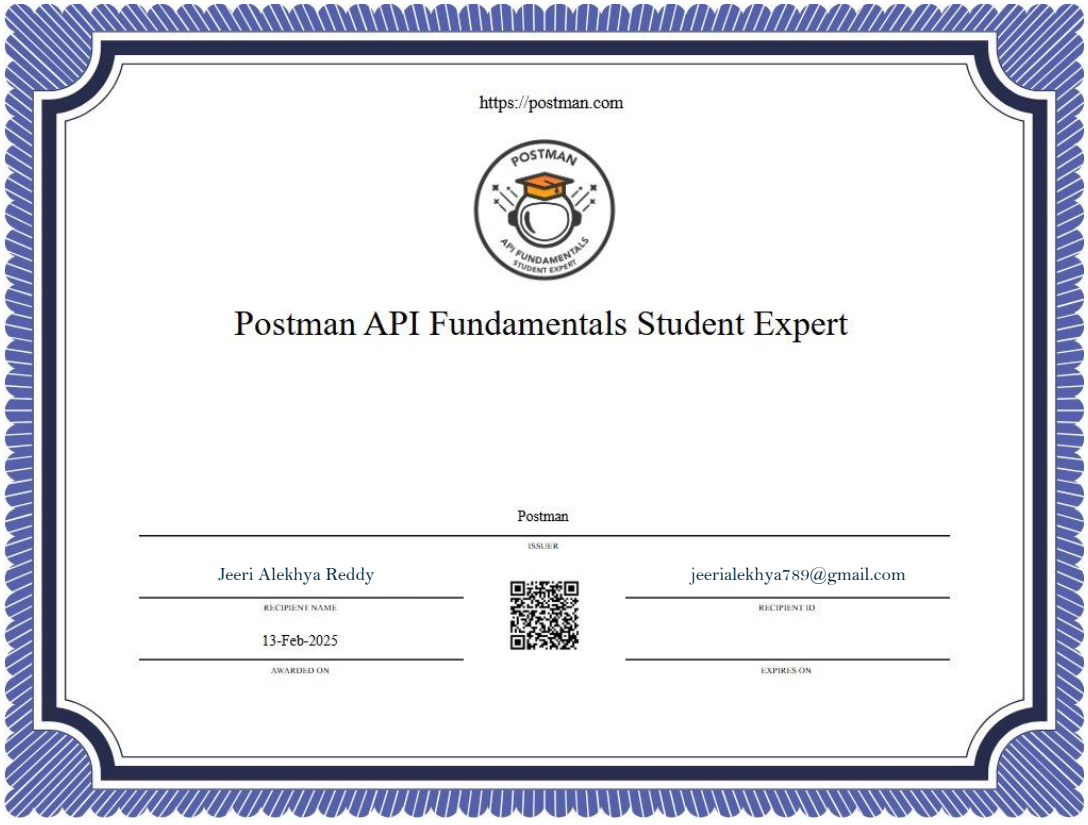
DIRECTOR, TRAINING SERVICES

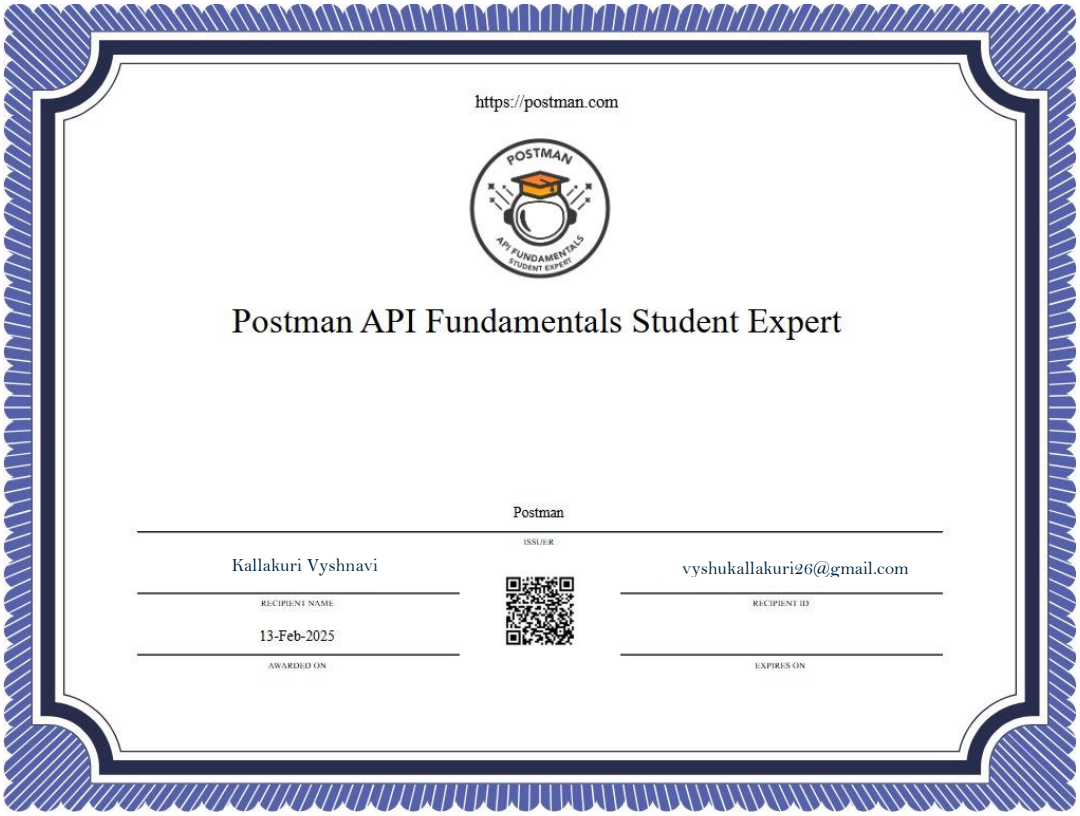
24 January 2025

POSTMAN API CERTICATION




https://postman.com
POSTMAN
API FUNDAMENTALS STUDENT EXPERT
Postman API Fundamentals Student Expert
Postman
ISSUER
Jakkula Madhurima
RECIPIENT NAME
madhurimajakkula6173@gmail.com
RECIPIENT ID
13-Feb-2025
AWARDED ON
EXPIRES ON


https://postman.com
POSTMAN
API FUNDAMENTALS STUDENT EXPERT
Postman API Fundamentals Student Expert
Postman
ISSUER
Jeeri Alekhya Reddy
RECIPIENT NAME
jeerialekhya789@gmail.com
RECIPIENT ID
13-Feb-2025
AWARDED ON
EXPIRES ON


https://postman.com
POSTMAN
API FUNDAMENTALS
STUDENT EXPERT
Postman API Fundamentals Student Expert
Postman
ISSUER
Kallakuri Vyshnavi
RECIPIENT NAME
vyshukallakuri26@gmail.com
RECIPIENT ID
13-Feb-2025
AWARDED ON
EXPIRES ON

ANDHRA UNIVERSITY

PROF. R. PADMA SREE
Principal

Sivajipalem Campus,
Visakhapatnam - 530 017
Phone (O): 0891-2592592
Email: principal.aucew@gmail.com

COLLEGE OF ENGINEERING FOR WOMEN

Visakhapatnam
Dt. 16-12-2024

To

The Founder & CEO,
Dr. Pallam Setti Centre for Research & Technology,
2nd floor, Instrumentation Department,
A-hub, Andhra University,
Maddilapalem, Visakhapatnam-530003

Sir,

Sub – Request for accepting our students to do project in your organization- regd.

This is to inform you that our B.Tech. – CS&SE students have to do project work in their fourth year. In this context, I request you to guide the following students to complete their project work on or before first week of April, 2025 in their thrust areas.

Please accept.

S.No.	Regd. No.	Name
1	321114110009	BALUSU DEEPIKA
2	321114110010	BOJJA AKHILA
3	321114110011	BONTHALAKOTI.REVATHI
4	321114110012	CHALLA NAGA VINAYA
5	321114110021	GORLA SUKANYA
6	321114110022	JAKKULA MADHURIMA
7	321114110023	JEERI ALEKHYA REDDY
8	321114110024	KALLALURI SRI LAKSHMI VYSHNAVI
9	321114110029	KOMMU KUSUMAKUMARI
10	321114110030	KURACHA DEEPIKA
11	321114110032	MASEEHA UNNISA
12	321114110033	MASINA JYOTHI SREE DURGA
13	321114110042	NAVUDU.JHANSI
14	321114110043	NIMMAGADDA MOHITHA BHAVYA SRI
15	321114110044	PANDU.SUSHMA
16	321114110045	PENTAKOTA LEELA SRI

Thanking you,
Yours sincerely

(R. PADMA SREE)
PRINCIPAL
A.U. College of Engineering for Women
Visakhapatnam-530 017