A Dynamic Selection Hybrid Model for Advancing Thyroid Care with BOOST Balancing Method

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. Md. Benazir Fathima - 21B01A05A8
2. N. Neha - 21B01A05B3
3. N. Sai Ramya Sri - 21B01A05B4
4. N. Lakshmi Priya - 21B01A05B9
5. O. Pranathi Sudha- 21B01A05C7

Under the esteemed guidance of

Mr.P.Raju

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)

(Approved by AICTE, accredited by NBA, Affiliated to JNTU Kakinada)

BHIMAVARAM–534202

2024 – 2025

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)

(Approved by AICTE, accredited by NBA, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VISHNU
UNIVERSAL LEARNING

CERTIFICATE

*This is to certify that the project entitled "**A Dynamic Selection Hybrid Model for Advancing Thyroid Care with Boosting Balancing Methods**", is being submitted by* Md. Benazir Fathima, N. Neha, N. Sai Ramya Sri, N. Lakshmi Priya, O. Pranathi Sudha *bearing the **Regd.No's. 21B01A05A8, 21B01A05B3, 21B01A05B4, 20B01A05B9, 21B01A05C7** in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by her under my guidance and supervision during the academic year **2024–2025** and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide **Head of the Department**

External Examiner

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this seminar. We take this opportunity to express our gratitude to all those who have helped us in this seminar.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju**, Chairman of SVES, for his constant support on each and every progressive work of mine.

We wish to express our sincere thanks to **Dr. G. Srinivasa Rao**, Principal of SVECW for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to **Dr. P. Srinivasa Raju**, Director and Admin of SVES, for being a source of inspirational and constant encouragement.

We wish to express our sincere thanks to **Prof. P. Venkata Rama Raju,** Vice Principal of SVECW for being a source of inspirational and constant encouragement.

We wish to place our deep sense of gratitude to **Dr. P. Kiran Sree**, Head of the Department of Computer Science & Engineering for his valuable pieces of advice in completing this seminar successfully.

We are deeply thankful to our project coordinator **Dr. P. R. Sudha Rani**, Professor for her in dispensable guidance and unwavering support throughout our project's completion. Her expertise and dedication have been invaluable to our success.

Our deep sense of gratitude and sincere thanks to **Mr. P. Raju**, Assistant Professor for his unflinching devotion and valuable suggestions throughout my project work.

Project Associates:

1. Md. Benazir Fathima - 21B01A05A8
2. N. Neha - 21B01A05B3
3. N. Sai Ramya Sri - 21B01A05B4
4. N. Lakshmi Priya - 21B01A05B9
5. O. Pranathi Sudha- 21B01A05C7

ABSTRACT

The project presents a Dynamic Selection Hybrid Model (DSHM) to improve thyroid disease diagnosis through advanced data processing and adaptive learning techniques. Traditional models struggle with inconsistent performance due to data imbalances and are often unable to adjust effectively to new data characteristics. Our DSHM approach tackles these limitations by introducing the BOO-ST balancing method, which enhances the representation of minority classes while minimizing noisy samples. This project employs robust feature selection techniques, such as Univariate and Information Gain, to retain the most critical attributes for prediction and reduce computational load. The DSHM dynamically evaluates and selects optimal classifiers based on real-time data needs, improving adaptability and accuracy in thyroid disease diagnosis. For transparency and clinical interpretability, we integrated Explainable AI tools, LIME and SHAP, allowing healthcare professionals to understand how specific factors influence the model's predictions. This enhanced explainability builds trust and improves decision-making processes in clinical settings. Through its high adaptability, performance, and clarity, DSHM shows potential as an effective tool for the early diagnosis of thyroid disorders, contributing to timely intervention and improved patient outcomes.
.

TABLE OF CONTENTS

1. INTRODUCTION

Thyroid disorders, including hypothyroidism and hyperthyroidism, affect millions of people worldwide and have a significant impact on the quality of life. These conditions can lead to a wide range of symptoms, from fatigue and weight gain to anxiety and heart issues, depending on the type and severity of the disorder. Early and accurate diagnosis is essential for effective treatment, but achieving this is often challenging due to the complexity of the conditions and the diverse ways they can present in patients.

The objective of this study is to develop a Dynamic Selection Hybrid Model aimed at enhancing the diagnostic process for thyroid disorders, specifically hypothyroidism and hyperthyroidism and the particular condition, by implementing the BOOST Balancing Method. This technique will address the issue of imbalanced data commonly found in thyroid disorder datasets.

This project focuses on the development and implementation of a hybrid machine learning model designed to:

- Improve the classification accuracy of thyroid disorders.
- Handle imbalanced data through the BOOST Balancing Method.
- Integrate multiple classifiers (Decision Trees, SVM, KNN, Random Forest, AdaBoost, Gradient Boost, catboost and xgboost) to build a robust, adaptable system.
- Evaluate performance using metrics like accuracy, precision, recall, and F1-score to ensure reliable classification.

Thyroid disorders, including hypothyroidism and hyperthyroidism, are prevalent medical conditions that significantly impact millions of individuals worldwide. Accurate diagnosis and timely treatment of thyroid disorders are essential for preventing severe health complications. However, diagnosing these conditions remains a challenge due to their complex clinical presentations and the imbalanced nature of medical data related to thyroid diseases. Traditional diagnostic methods, including some machine learning models, struggle to handle the variability and imbalance within thyroid-related datasets, resulting in less accurate predictions.

In this study, we propose a solution to these challenges by developing a Dynamic Selection Hybrid Model aimed at improving the diagnostic process for thyroid disorders. This model incorporates a variety of machine learning algorithms such as Decision Trees, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), catboost, xgboostRandom Forest, AdaBoost, and Gradient Boosting within an adaptive framework. The dynamic nature of the model allows it to select the best-performing classifier based on real-time data inputs, ensuring flexibility and responsiveness in a clinical environment.

A significant aspect of this project is addressing the problem of imbalanced data, which is common in thyroid disorder datasets. Imbalanced data can skew model predictions, making it difficult to identify minority classes, such as rarer thyroid conditions. To resolve this, we implement the BOOST Balancing

Method, which combines Boosting with Sample Weighting (BS), SMOTE (Synthetic Minority Over-sampling Technique), and Tomek Links. This method ensures a more equitable representation of all thyroid disorder classes, enabling the model to classify thyroid conditions more accurately across eight specific categories: four hyperthyroid conditions (subclinical, T3 toxic, toxic goitre, secondary toxic) and four hypothyroid conditions (subclinical, primary hypothyroid, compensated hypothyroid, secondary hypothyroid).

By integrating multiple machine learning algorithms with real-time adaptability and advanced balancing techniques, the Dynamic Selection Hybrid Model aims to significantly enhance the precision of thyroid disorder diagnoses. The model's dynamic selection mechanism allows for higher adaptability in clinical settings, improving the accuracy of diagnoses and advancing thyroid care quality. This approach is expected to outperform traditional static models, offering a more robust, flexible, and reliable tool for clinicians.

2. LITERATURE REVIEW

I. A. Fatima, M. Gul, and M. Abid, "Ensemble Learning for Thyroid Disease Prediction Using Hybrid Models," in *IEEE Access*, vol. 11, pp. 47832-47845, 2023.

Explanation: This paper presents a hybrid ensemble learning approach combining multiple machine learning algorithms such as Support Vector Machine (SVM), Decision Trees, and Random Forest for thyroid disease classification. The authors implemented a meta-classifier that integrates predictions from various models to improve overall diagnostic accuracy. The hybrid model was tested on the Thyroid Disease dataset from the UCI repository, achieving higher accuracy compared to individual models. The study also addressed the issue of data imbalance using Synthetic Minority Over-sampling Technique (SMOTE), highlighting its impact on improving classification for minority classes like hyperthyroidism.

II. S. Chen, X. Zhao, and Z. Huang, "An Explainable AI Approach for Thyroid Disease Detection Using Deep Learning," in *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 4, pp. 1345-1357, 2023.

Explanation: This paper focuses on the application of deep learning models, particularly Convolutional Neural Networks (CNNs), for the detection of thyroid disorders. The study emphasizes the need for interpretability in medical diagnostics and presents an explainable AI (XAI) framework that highlights important features contributing to thyroid disease predictions. The model was trained and validated on a large-scale thyroid disease dataset, with attention given to improving the classification of complex conditions like subclinical hypothyroidism. The results show that the explainable model can offer accurate predictions while providing insights into the decision-making process, helping clinicians trust the AI system.

III. H. Wang, Q. Lin, and Y. Liu, "Handling Data Imbalance in Thyroid Disease Classification Using a Hybrid SMOTE and Cost-Sensitive Learning Approach," in *IEEE Access*, vol. 10, pp. 89467-89476, 2022.

Explanation: In this paper, the authors propose a hybrid approach to handle class imbalance in thyroid disease classification using a combination of SMOTE and cost-sensitive learning techniques. The hybrid approach was applied to a thyroid disease dataset, where imbalanced classes, particularly underrepresented conditions like secondary hypothyroidism, were a challenge. The study utilized machine learning classifiers such as Decision Trees, Random Forest, and Gradient Boosting. The results showed a significant improvement in the classification of minority classes, enhancing the overall performance of the model. The paper provides valuable insights into how imbalance handling methods can improve healthcare applications where data distribution is uneven.

IV. N. Khurana and R. Sharma, "Thyroid Disease Prediction Using Random Forest and Boosting Techniques with Imbalanced Data," in *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 20, no. 2, pp. 226-235, 2023.

Explanation: This paper presents a novel approach to thyroid disease diagnosis using Random Forest and Boosting algorithms, specifically targeting the challenges of imbalanced datasets. The authors incorporated the Adaboost technique alongside Random Forest to improve the classification of minority thyroid conditions, such as T3 toxic and secondary hypothyroidism. The study evaluated the model on a real-world thyroid disease dataset and demonstrated that combining these techniques with oversampling methods like SMOTE significantly enhanced prediction accuracy, especially for rare thyroid conditions. The hybrid model achieved a higher recall rate for these minority classes compared to traditional models.

V. R. Jadhav and A. Ghosh, "A Comparative Study of Machine Learning Algorithms for Thyroid Disease Classification Using SMOTE and Tomek Links," in *IEEE Access*, vol. 10, pp. 78767-78780, 2022.

Explanation: The paper compares various machine learning algorithms, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Gradient Boosting, for thyroid disease classification. The authors applied the Synthetic Minority Over-sampling Technique (SMOTE) in conjunction with Tomek Links to handle imbalanced data in the dataset. The study emphasizes the role of combining oversampling with undersampling to achieve a more balanced dataset and better classification performance, particularly for conditions like subclinical hypothyroidism and hyperthyroidism. The comparative results show that Gradient Boosting, paired with SMOTE and Tomek Links, achieved the highest classification accuracy and reduced false positive rates.

3. SYSTEM ANALYSIS

3.1 Existing System

Current methods for thyroid disorder classification primarily rely on traditional machine learning techniques and ensemble methods. Popular approaches include Random Forest, AdaBoost, and Gradient Boosting, which aggregate multiple classifiers to improve prediction accuracy. These models use decision trees and boosting algorithms to enhance performance by combining the predictions of several weak learners to create a more accurate and robust classifier.

3.2.Disadvantages of existing systems

- Limited Handling of Imbalanced Data: Traditional models often struggle with imbalanced datasets, leading to biased results, particularly in predicting minority classes.
- Complexity and Computation: Ensemble methods can be computationally intensive and complex to implement, requiring significant resources for training and deployment.
- Overfitting Risk: While ensemble methods generally reduce overfitting, they are still susceptible to it if not properly tuned, especially with noisy or high-dimensional data.

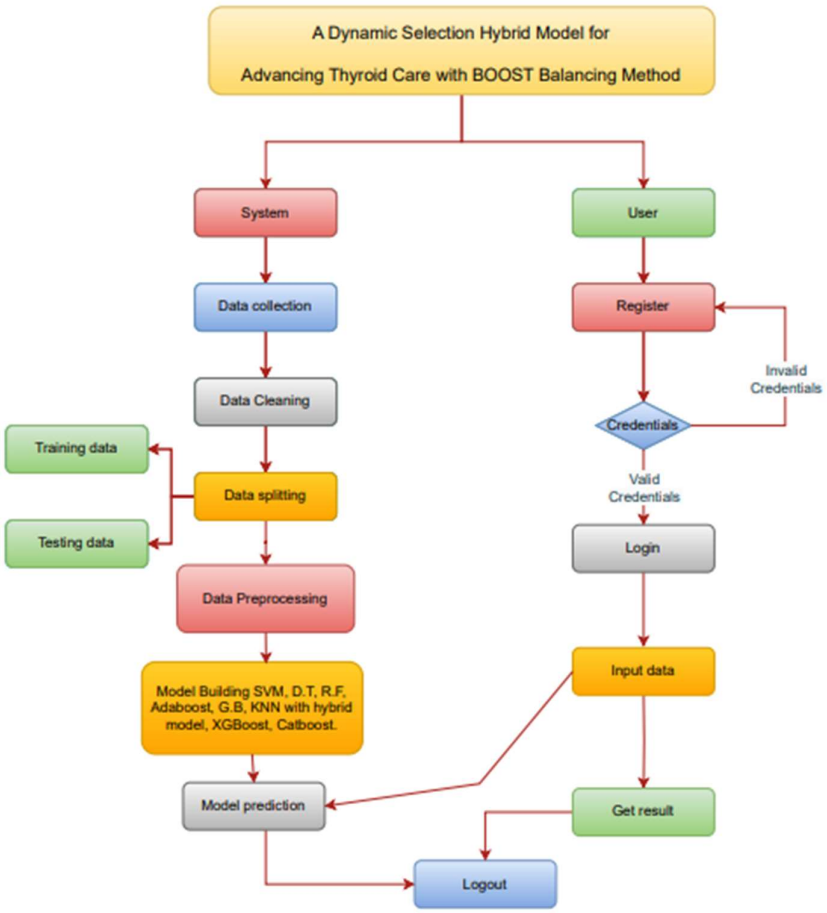
3.3. Proposed System

The proposed system is a Dynamic Selection Hybrid Model designed to enhance thyroid disorder classification by integrating multiple machine learning algorithms like Decision Trees, SVM, KNN, Random Forest, AdaBoost, Gradient Boosting, Catboost, and Xgboost within an Adaptive Dynamic Ensemble Framework. Additionally, the BOOST Balancing Method is incorporated to handle imbalanced data effectively.

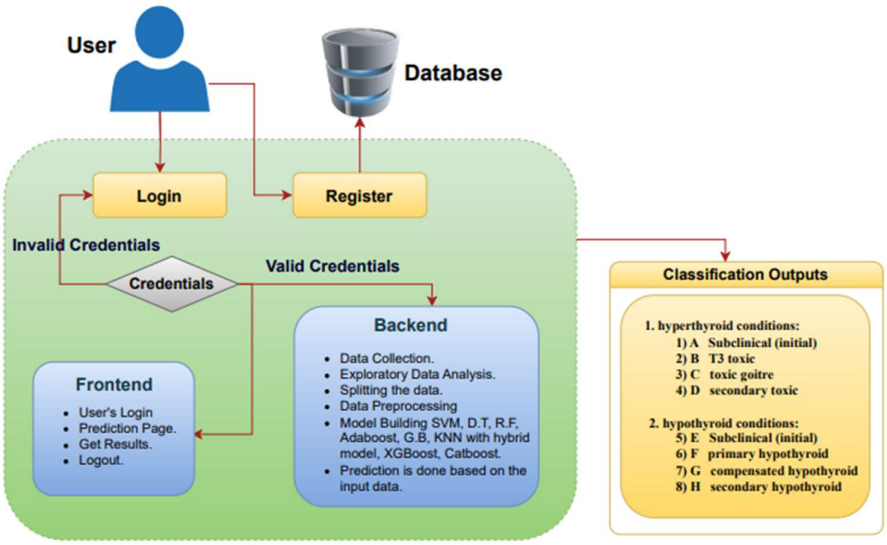
3.4. Advantages of Proposed System

- Enhanced Accuracy: By integrating multiple machine learning algorithms and dynamically selecting the best-performing classifier, the system improves overall classification accuracy for thyroid disorders, reducing errors in diagnosis.
- Effective Handling of Imbalanced Data: The incorporation of the BOOST Balancing Method ensures better classification performance on imbalanced datasets, improving prediction accuracy for minority classes and reducing bias.
- Reduced Overfitting: The dynamic selection and adaptive learning mechanism help minimize overfitting, especially when dealing with noisy or high-dimensional data, resulting in more consistent and reliable model performance.

3.5 Project Flow



3.6 Architecture Diagram



4. METHODOLOGY

4.1 Overview

1. Data Collection: Obtain a dataset containing historical thyroid data.

    - Dataset Source:Thyroid disorder Dataset (thyroidDF.csv) from kaggle website.
    - Content: The dataset includes features like hypopituitary gland, age, sex, goiter, etc., Totally 31 columns.

2. Data Exploration: Understand the dataset's structure and content.

    - Actions:
        - Load the dataset into a DataFrame.
        - Display basic information about the dataset (e.g., data types, missing values).
        - Generate summary statistics to understand the distribution of numeric features.
        - Visualize the distribution of different thyroid disorder conditions.

3. Data preprocessing

    - Drop unwanted columns and duplicates
    - Handle Null values
        - Drop if it contains more than 50% null values
        - Fill or replace null values with mean, median or mode.
    - Implement BOOST balancing method to balance the data.

        BOO-ST:

        - BS(Boosting with Sample Weighting),
        - SMOTE (Synthetic Minority Over-sampling Technique),
        - Tomek Links (TL)

        Implementation:

        - SMOTE: Balance the dataset by oversampling the minority class.
        - Tomek Links: Remove noisy points to improve class separation.
        - Boosting: Train our model on the refined dataset.
    - Implement Label encoding to convert object type columns into numeric columns.

4. Data Splitting

5. Build a Dynamic Selection Hybrid Model

   Implementation:

   - Define the classifiers
   - Step 1: Train all classifiers and compute Permutation Feature Importance (PFI)

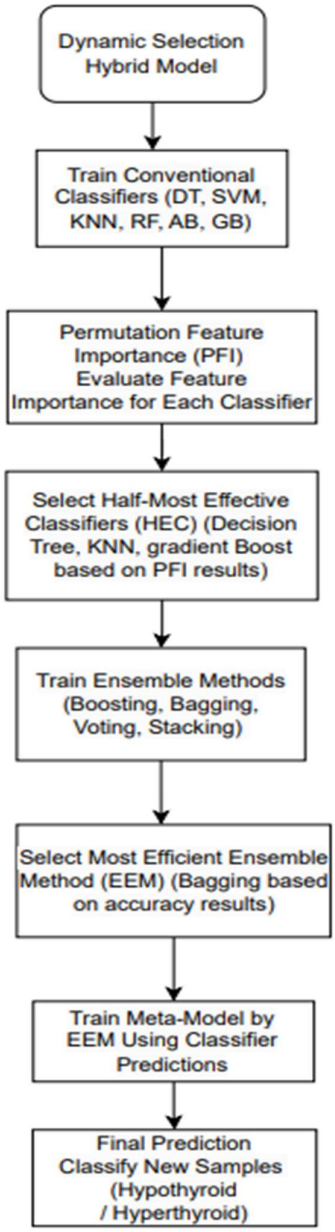
   - Step 2: Select Half-Most Effective Classifiers (HEC) based on PFI
   - Step 3: Define the ensemble methods using the selected classifiers
   - Step 4: Train each ensemble method and evaluate accuracy
   - Step 5: Select Most Efficient Ensemble Method (EEM)
   - Build final model.

6. Save the model for deploy into web application frontend.

8. Deployment and Testing: Test the prediction function with sample inputs.

   - Use various sample inputs to validate its accuracy and reliability.

4.2 Dynamic Selection Mechanism methodology:



1. Split Data:

   - Split the preprocessed data into training and testing sets (e.g., 80% training, 20% testing) to prepare for model evaluation.

2. Define the Classifiers:

   - Define a set of classification algorithms such as Decision Trees, Support Vector Machines, K-Nearest Neighbors, Random Forest, AdaBoost, and Gradient Boosting.

3. Train All Classifiers and Compute Permutation Feature Importance (PFI):

    - Train each classifier on the training dataset.
    - Compute Permutation Feature Importance (PFI) for each trained model to evaluate the relative importance of features. The PFI is a method to measure how each feature contributes to the model's predictive performance.

4. Rank Trained Models Using PFI:

    - Rank the trained classifiers based on their PFI scores. The classifiers with higher PFI scores indicate better importance evaluation.

5. Select Half-Most Effective Classifiers (HEC):

    - Select the top 50% most effective classifiers (HEC) based on their PFI ranking. This selection ensures only the best-performing classifiers are used for further model integration.
    - If the number of models is odd, divide the number by 2 and round up to the nearest integer. For example, if there are 5 classifiers, 50% of 5 is 2.5. In this case, we'll select the top 3 classifiers instead of just 2. This ensures that more of the effective classifiers are included in the ensemble, avoiding dropping a potentially valuable model.

6. Define the Ensemble Methods:

    - Create ensemble methods using the selected HEC classifiers. Common ensemble techniques include Boosting, Bagging, Voting, and Stacking.
    - Define each ensemble method using the selected classifiers to enhance performance through collaboration between different models.

7. Train Each Ensemble Method and Evaluate Accuracy:

    - Train each defined ensemble method on the training dataset.
    - Evaluate the accuracy of each ensemble method using the testing dataset to measure its predictive performance.

8. Select Most Efficient Ensemble Method (EEM):

    - Compare the accuracy of all ensemble methods and select the Most Efficient Ensemble Method (EEM), which has the highest accuracy.

This process ensures that the final model is not only accurate but also dynamically selected based on the importance of features and classifier performance, providing a robust solution for classifying thyroid disorders.

4.3 Algorithm implementation (Dynamic Selection Hybrid Model)

- **Define the Classifiers**: Define classification algorithms.

Classifiers: Decision Tree, SVM, KNN, Random Forest, AdaBoost, Gradient Boost

```
# Define the classifiers
classifiers = {
    'Decision Tree': DecisionTreeClassifier(),
    'SVM': SVC(probability=True),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier()
}
```

- Step 1: Train all classifiers and compute Permutation Feature Importance (PFI). Permutation Feature Importance (PFI) is a method used to evaluate the importance of features in a machine learning model.

Implementation:

1. **Training**:
   - Fit the model to the training data using the .fit(X_train, y_train) method.
2. **Prediction**:
   - Predict the thyroid conditions on both training and test datasets using the .predict(X_train) and .predict(X_test) methods.
3. **Evaluation**:
   - Calculate accuracy scores for the training and test datasets using accuracy_score().
   - Generate and visualize the confusion matrix using confusion_matrix() and seaborn.heatmap().
   - Use classification_report() to obtain precision, recall, and F1 scores.
4. **Result Interpretation**:
   - Examine the confusion matrix and classification report to assess the model's performance in predicting thyroid disorders.

Result for Decision Tree:

	Precision	Recall	F1-Score	Support
0	1	0.99	0.99	87
1	1	1	1	86
2	1	1	1	87
3	0.97	0.98	0.97	87
4	1	1	1	86
5	0.93	0.99	0.96	86
6	1	0.94	0.97	86
7	1	1	1	88
Accuracy	0.99	0.99	0.99	693
Macro Avg	0.99	0.99	0.99	693

Table: Classification report for Decision Tree

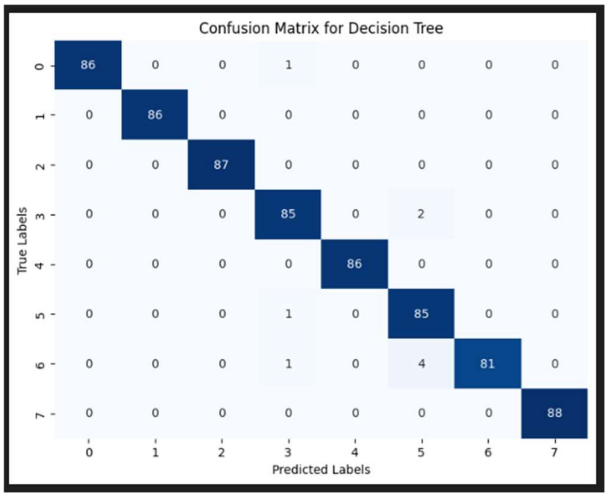


Figure: Confusion matrix for Decision Tree

Result for SVM :

	precision	recall	f1-score	support
0	0.9	0.92	0.91	87
1	0.72	0.85	0.78	86
2	0.83	0.78	0.8	87
3	0.93	0.91	0.92	87
4	0.89	0.81	0.85	86
5	0.75	0.6	0.67	86
6	0.74	0.66	0.7	86
7	0.8	1	0.89	88
accuracy	-	-	-	568.26
macro avg	0.82	0.82	0.81	693
weighted avg	0.82	0.82	0.82	693

Table: Classification report for SVM

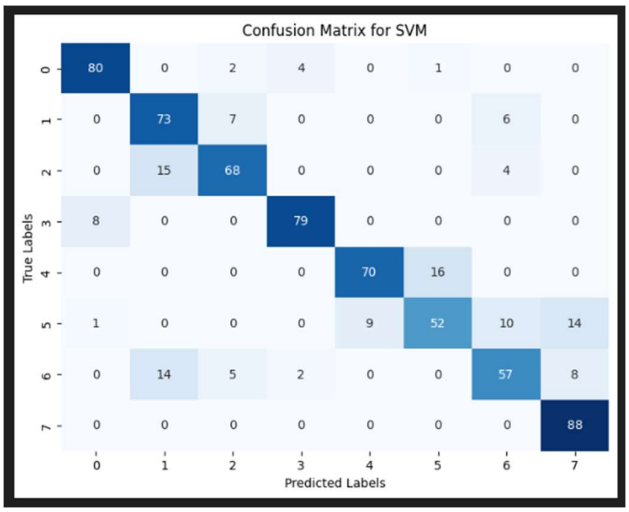


Figure: Confusion matrix for SVM

Result for KNN :

	Class	Precision	Recall	F1-Score	Support
0	0	0.99	0.89	0.93	87
1	1	0.82	0.87	0.85	86
2	2	0.89	0.91	0.9	87
3	3	0.93	1	0.96	87
4	4	0.89	0.92	0.9	86
5	5	0.89	0.78	0.83	86
6	6	0.9	0.85	0.87	86
7	7	0.92	1	0.96	88
8	Accuracy				693
9	Macro Avg	0.9	0.9	0.9	693
10	Weighted Avg	0.9	0.9	0.9	693

Table: Classification report for KNN

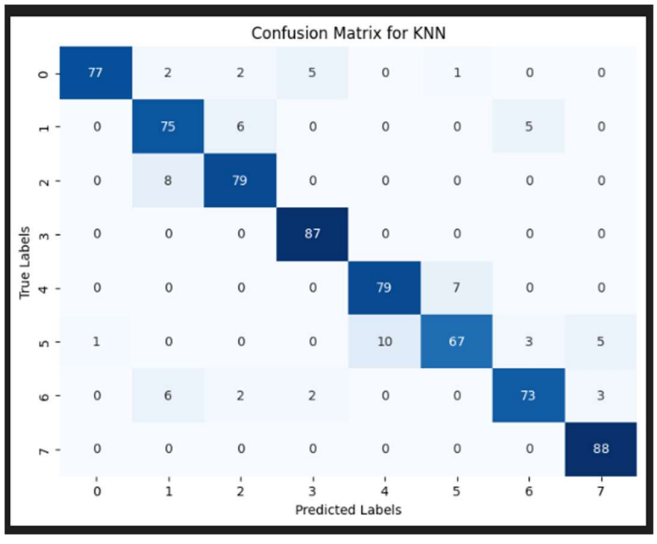


Figure: Confusion matrix for KNN

Result for Random Forest:

Class	Precision	Recall	F1-Score	Support
0	1	0.99	0.99	87
1	1	1	1	86
2	1	1	1	87
3	0.96	1	0.98	87
4	1	1	1	86
5	1	0.98	0.99	86
6	0.99	0.98	0.98	86
7	1	1	1	88
accuracy		0.99		693
macro avg	0.99	0.99	0.99	693
weighted avg	0.99	0.99	0.99	693

Table: Classification report for Random Forest

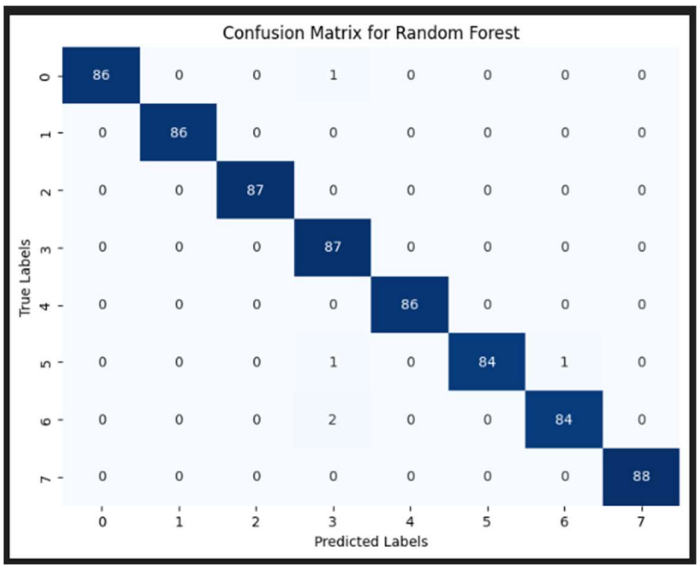


Figure: Confusion matrix for Random Forest

Result for Adaboost:

	precision	recall	f1-score	support
0	0.32	0.99	0.48	87
1	0	0	0	86
2	0.99	1	0.99	87
3	0.92	0.63	0.75	87
4	0	0	0	86
5	0.91	0.9	0.9	86
6	0.77	0.88	0.82	86
7	1	1	1	88
macro avg	0.61	0.67	0.62	693
weighted avg	0.61	0.68	0.62	693

Table: Classification report for Adaboost

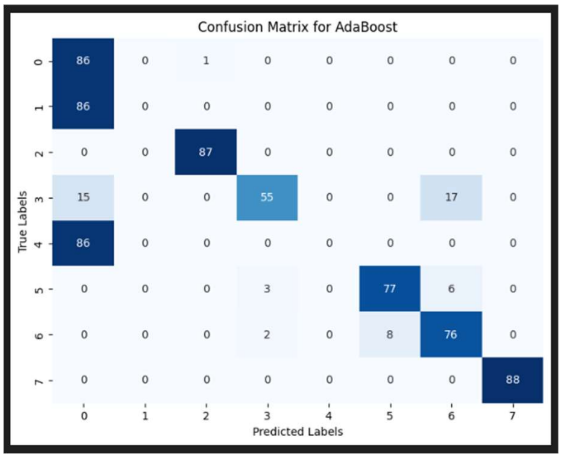


Figure: Confusion matrix for Adaboost

Result for Gradient Boosting:

	Precision	Recall	F1-Score	Support
0	1	0.99	0.99	87
1	1	1	1	86
2	1	1	1	87
3	0.99	1	0.99	87
4	1	1	1	86
5	0.99	0.97	0.98	86
6	0.97	0.99	0.98	86
7	1	1	1	88
accuracy	0.99			693
macro avg	0.99	0.99	0.99	693
weighted avg	0.99	0.99	0.99	693

Table: Classification report for Gradient Boosting

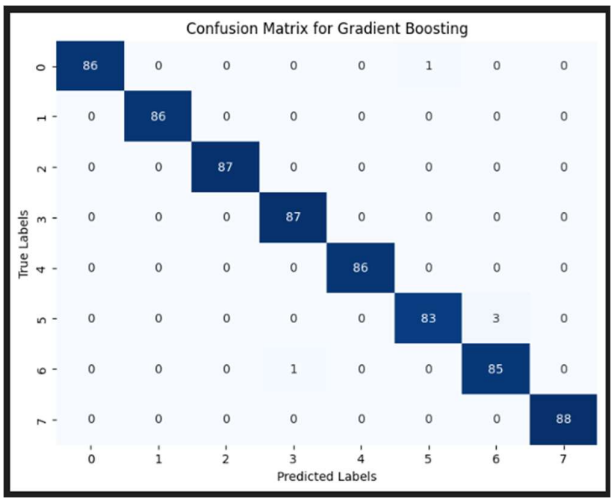


Figure: Confusion matrix for Gradient Boosting

Results for Xgboost:

	Precision	Recall	F1-Score	Support
0	1	0.99	0.99	87
1	0.99	1	0.99	87
2	1	1	1	88
3	0.98	0.97	0.97	87
4	1	1	1	87
5	0.97	1	0.98	86
6	0.98	0.94	0.96	85
7	0.99	1	0.99	87
accuracy	0.99			694
macro avg	0.99	0.99	0.99	694
weighted avg	0.99	0.99	0.99	694

Table: Classification report for XGBoost

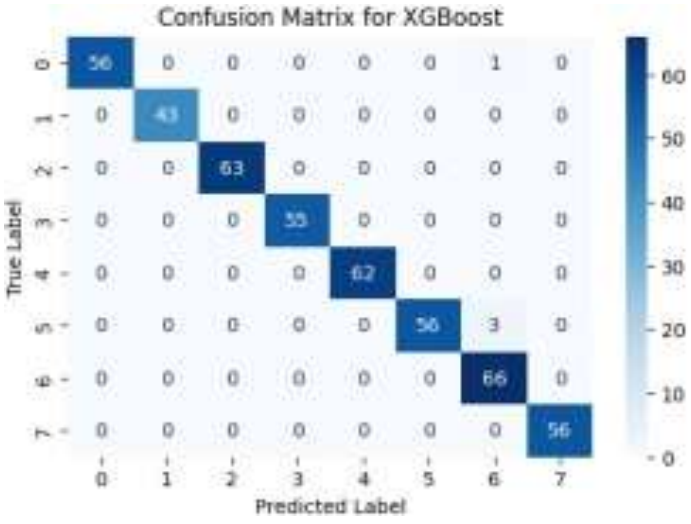


Figure: Confusion matrix for XGboost

Results for Catboost:

	Precision	Recall	F1-Score	Support
0	1	0.99	0.99	87
1	1	1	0.99	87
2	1	1	1	88
3	0.97	0.97	0.97	87
4	1	1	1	87
5	0.99	1	0.99	86
6	0.96	0.95	0.96	85
7	0.99	1	0.99	87
accuracy	0.99			694
macro avg	0.99	0.99	0.99	694
weighted avg	0.99	0.99	0.99	694

Table: Classification report for CatBoost

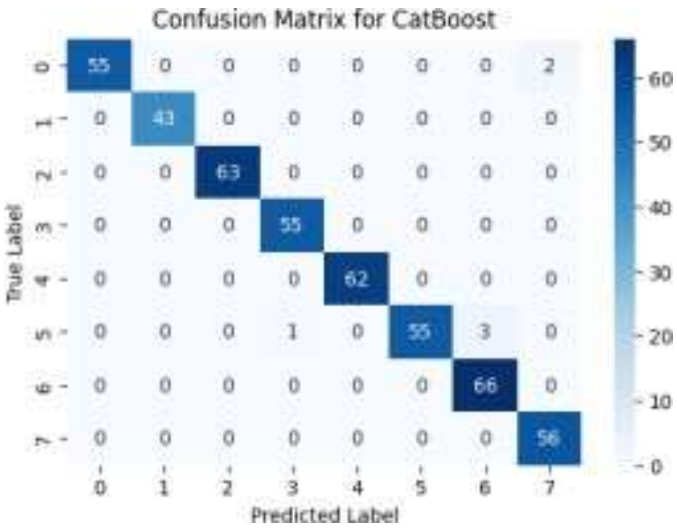


Figure: Confusion matrix for Catboost

- Step 2: Select Half-Most Effective Classifiers (HEC) based on PFI

Model	Accuracy (%)	PFI Score
Decision Tree	98.7	0.158043344
SVM	81.82	0.105919505
KNN	90.19	0.125517028
Random Forest	99.28	0.123417957
Adaboost	67.68	0.092873065
Gradient Boosting	99.28	0.123585139

    - In here totally 6 algorithms were implemented. So we need to take most effective 3 (half) algorithms based upon the PFI score.
    - HEC: Decision Tree, KNN, Gradient Boosting.
    - So we can take these three algorithms to define ensemble methods.
- Step 3: Define the ensemble methods using the selected classifiers.
    - **Boosting: AdaBoost Classifier**

      AdaBoost (Adaptive Boosting) enhances the performance of a weak classifier by focusing on the mistakes made by previous classifiers. It iteratively adjusts the weights of incorrectly classified instances, making them more significant for subsequent classifiers. The ensemble combines multiple weak models to form a strong predictor.

        - **Base Estimator**: Decision Tree
        - **Purpose**: To improve model accuracy by sequentially adjusting to errors made by the model.
        - **Characteristics**: Focuses on difficult-to-classify examples, reduces bias, and can handle noisy data well.

    - **Bagging: Bagging Classifier**

      Bagging (Bootstrap Aggregating) improves the stability and accuracy of machine learning algorithms by training multiple models on different subsets of the data and averaging their predictions. It helps reduce variance and prevent overfitting by using the same model type but with different data samples.

        - **Base Estimator**: Decision Tree
        - **Purpose**: To improve prediction accuracy and robustness by reducing variance.
        - **Characteristics**: Uses random sampling with replacement to create multiple datasets, trains multiple models in parallel, and combines their predictions.

- **Voting: Voting Classifier**

  Voting combines the predictions of multiple classifiers to make a final decision. It can be either hard voting, where the majority class is chosen, or soft voting, where class probabilities are averaged to make the final decision. This method leverages the strengths of different classifiers to improve overall performance.

  - **Estimators**: Decision Tree, K-Nearest Neighbors (KNN), Gradient Boosting
  - **Purpose**: To aggregate predictions from diverse classifiers to improve accuracy and generalizability.
  - **Characteristics**: Combines multiple model types, can handle various types of classifiers, and is robust to overfitting.

- **Stacking: Stacking Classifier**

  Stacking involves training multiple base models and then combining their predictions using a meta-model. The base models generate predictions, which are then used as inputs for a meta-classifier that makes the final prediction. This method leverages the strengths of different models to achieve better performance.

  - **Base Estimators**: Decision Tree, K-Nearest Neighbors (KNN), Gradient Boosting
  - **Meta-Classifier**: Typically, a simple model like Logistic Regression is used to combine the outputs of the base models.
  - **Purpose**: To improve model performance by leveraging the diverse strengths of various models and combining them effectively.
  - **Characteristics**: Sequentially trains models, uses predictions of base models as features for the meta-model, and often provides better performance by learning from the strengths and weaknesses of individual base models.

- Step 4: Train each ensemble method and evaluate accuracy

Ensemble Method	Accuracy
Boosting	0.98701
Bagging	0.98999
Voting	0.99427
Stacking	0.99422
Xgboost	0.9870
Catboost	0.9885

- Step 5: Select Most Efficient Ensemble Method (EEM)
  - Most Efficient Ensemble Method is Voting classifier (accuracy = 99.427 %)
  - So we are taking Voting classifier to define our final model.
- Step 6: Build the final model
  - Final ensemble model: Voting classifier.
    - Estimators: Decision Tree, K-Nearest Neighbors (KNN), Gradient Boosting
- Result for Final Model

Class	Precision	Recall	F1-Score	Support
0	1.00	0.99	0.99	87
1	1.00	1.00	1.00	86
2	1.00	1.00	1.00	87
3	0.98	1.00	0.99	87
4	1.00	1.00	1.00	86
5	0.98	0.99	0.98	86
6	1.00	0.98	0.99	86
7	1.00	1.00	1.00	88
Accuracy			**0.99**	**693**
Macro Avg	0.99	0.99	0.99	693
Weighted Avg	0.99	0.99	0.99	693

Figure: Classification report for final model (Voting)

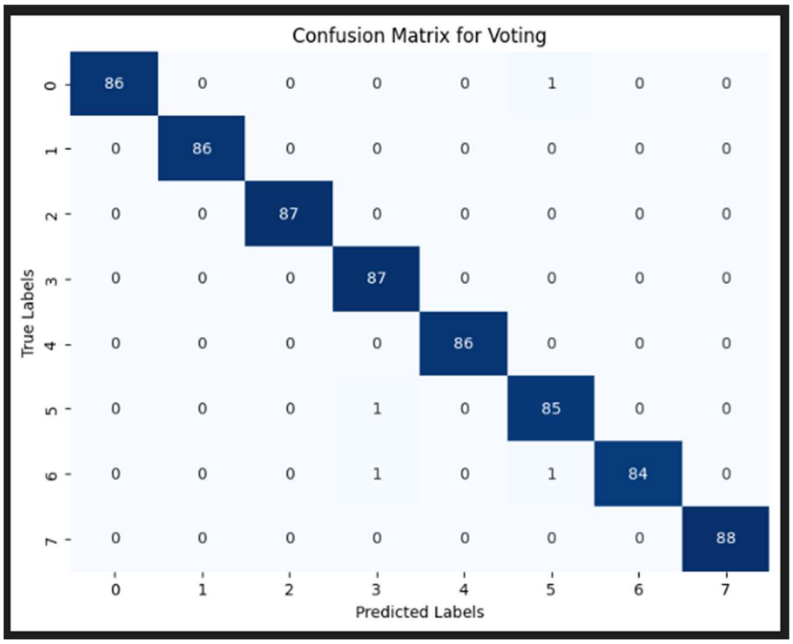


Figure: Confusion matrix

5. REQUIREMENT ANALYSIS

5.1 Function and non-functional requirements

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1) Authentication of user whenever he/she logs into the system
2) System shutdown in Solar prediction.
3) A verification email is sent to user whenever he/she register for the first time on some software system.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

5.2 Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

5.3 Software Requirements

- Operating System : Windows 7/8/10
- Programming Language : Python
- Libraries : Pandas, Numpy, scikit-learn.
- IDE/Workbench : Visual Studio Code.

6. SYSTEM DESIGN

6.1 Introduction of Input Design

In a thyroid disorder diagnosis system, input design is a critical phase because the quality of the input data directly affects the accuracy of the diagnostic model and the decision-making process. The input consists of various patient health data, including clinical test results (e.g., TSH, TT4, T3, FTI), demographic details (e.g., age, gender), and medical history (e.g., presence of thyroid treatments or surgeries). These inputs are gathered from multiple sources, such as healthcare records, manual data entry, or medical databases.

During the input design phase, it is essential to ensure that the data is accurate, complete, and user-friendly for healthcare professionals and patients alike. A well-structured input system guarantees that the diagnostic outputs are reliable, ensuring timely and correct identification of thyroid conditions.

Objectives for Input Design:

- To design an efficient system for entering and collecting patient health data related to thyroid conditions.
- To minimize data entry errors and ensure consistency in the inputs.
- To develop user-friendly data entry forms or interfaces that healthcare providers can easily interact with.
- To implement data validation techniques (e.g., range checks for clinical test results) to maintain data integrity and accuracy.
- To streamline the input process, allowing seamless data entry through electronic health records (EHR) systems, APIs, or manual input.

Key Features of Input Design:

- The system will focus on **collecting medical data** relevant to thyroid conditions, such as TSH, TT4, T3, FTI levels, and patient history (e.g., on thyroxine, pregnant, previous thyroid surgeries).
- **Accuracy and consistency** will be ensured through proper validation rules for clinical values, preventing incorrect data entry.
- **User-friendly interfaces** will allow healthcare providers and users to input or review patient data efficiently, reducing cognitive load and chances of mistakes.
- The system will focus on **simplicity and consistency** to ensure that the input process aligns with the end users' preferences and minimizes the risk of data entry errors.

6.2 Output Design

The output design for the thyroid disorder diagnosis system focuses on providing meaningful diagnostic results and insights based on the machine learning model's predictions. The primary output of the system will be the classification of the thyroid condition into one of eight categories (e.g., hyperthyroid or hypothyroid conditions). The design will ensure that the diagnostic results are presented in a clear, user-friendly manner to healthcare providers and patients, making it easy to understand and act upon.

Objectives of Output Design:

- To design output formats that deliver accurate and easy-to-understand predictions for thyroid conditions (e.g., Subclinical, Primary Hypothyroid, T3 Toxic).
- To ensure that the diagnostic results meet the user's requirements, such as detailed reports for specific patients, including clinical test results and medical recommendations.
- To ensure that the output is **clear, concise**, and delivered in an appropriate format, such as **tables, graphs, or textual reports.**
- To provide **real-time results**, allowing healthcare providers to make timely decisions for patient care.
- To implement **notifications or alerts** for critical thyroid conditions, ensuring that healthcare providers are informed immediately in case of severe or unusual conditions.

Key Features of Output Design:

- The system will display **real-time diagnostic results** for thyroid conditions, based on patient data inputs, and present them in a user-friendly interface.
- Output data will be presented through **interactive dashboards** with features such as graphs or tables, showing patient-specific insights like trends in TSH or TT4 levels over time.
- Diagnostic results will be available in various formats, including **summary reports, detailed diagnostic feedback**, and **interactive visuals** to help healthcare providers make informed decisions.
- **Alerts and notifications** will be sent when the system detects critical thyroid conditions (e.g., severe hypothyroidism), ensuring timely medical intervention.

The output design emphasizes ease of interpretation and actionable insights, providing healthcare providers with the tools needed to deliver high-quality thyroid care. The system will also ensure that outputs are presented promptly and clearly, especially in cases requiring urgent medical attention.

6.3 UML diagrams

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

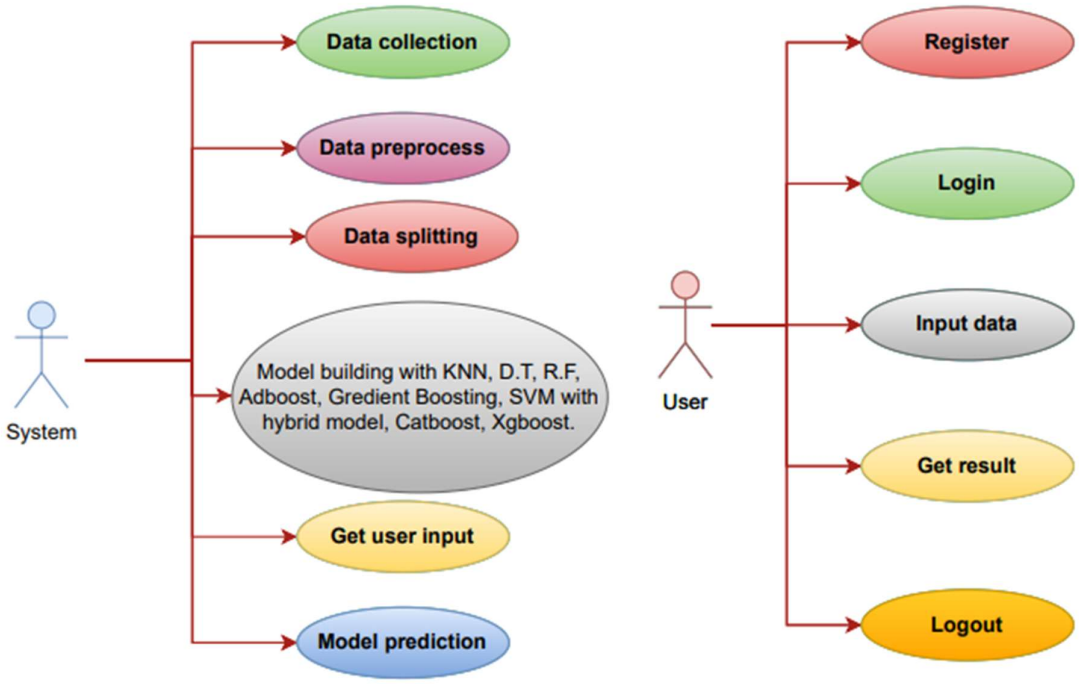
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

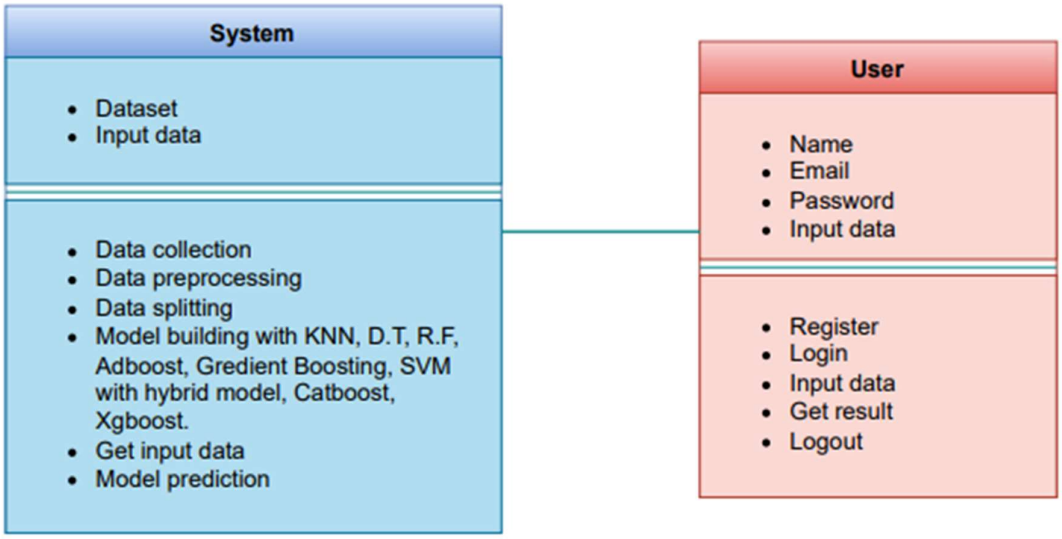
USE CASE DIAGRAM

- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
- Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



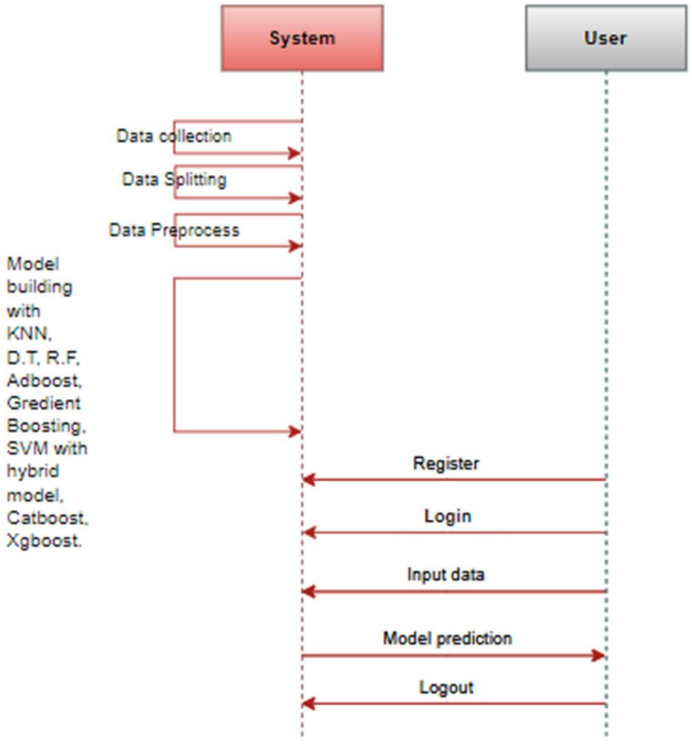
CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



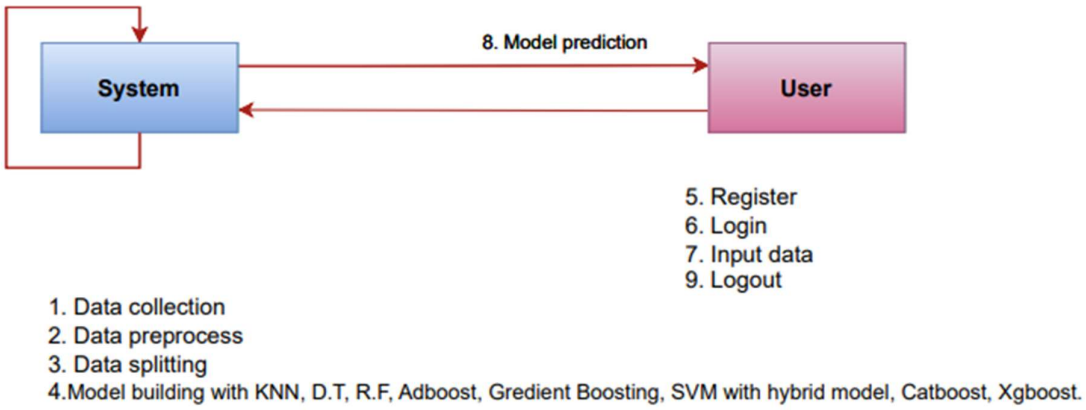
SEQUENCE DIAGRAM

- A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
- It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams



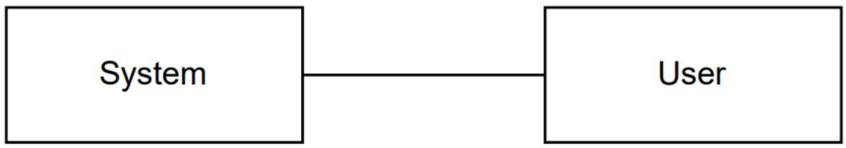
COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



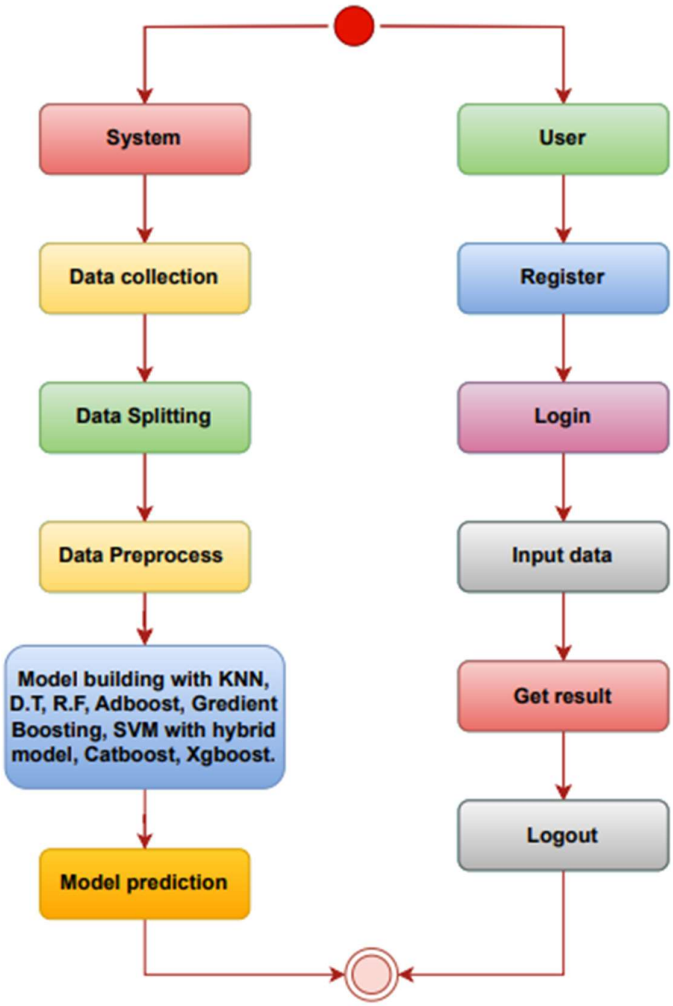
DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.



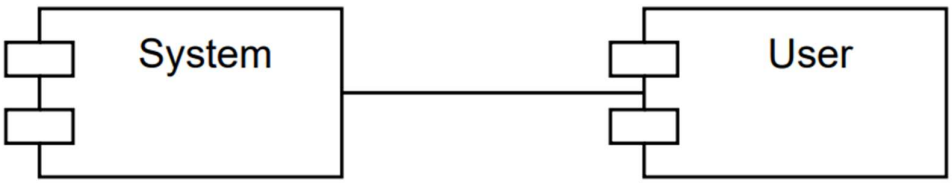
ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



COMPONENT DIAGRAM:

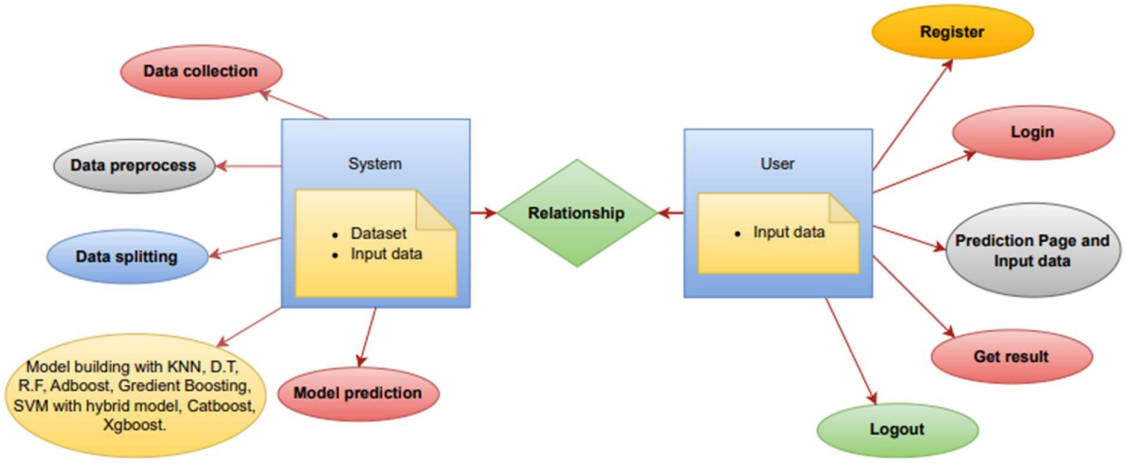
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



ER DIAGRAM:

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

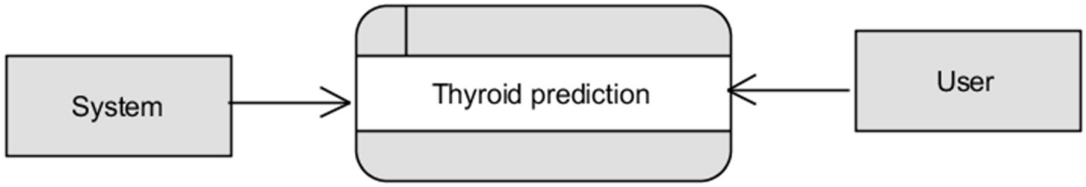
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.



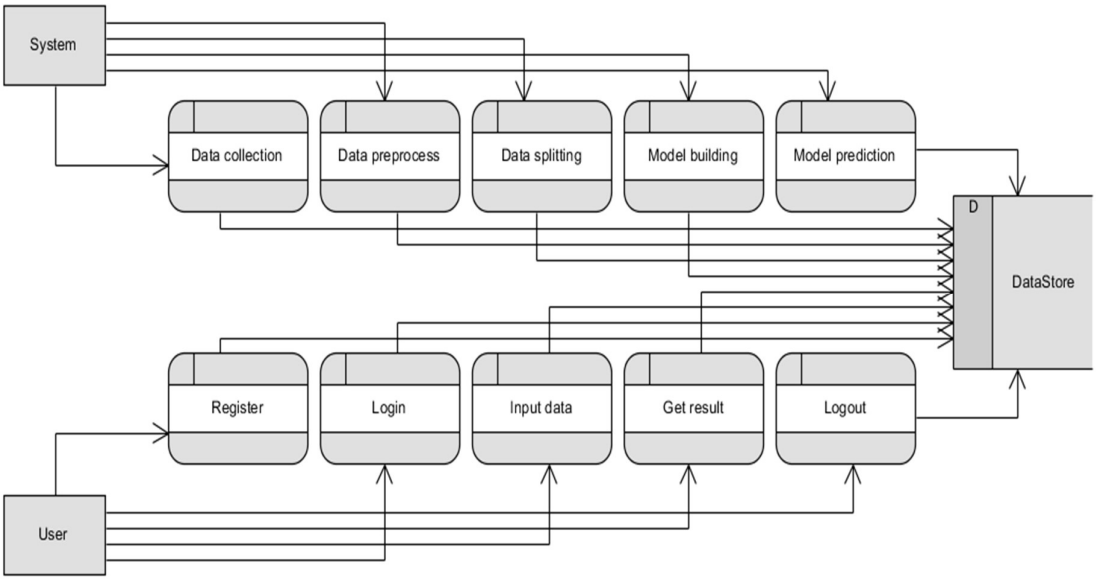
6.3 DATA FLOW DIAGRAM :

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

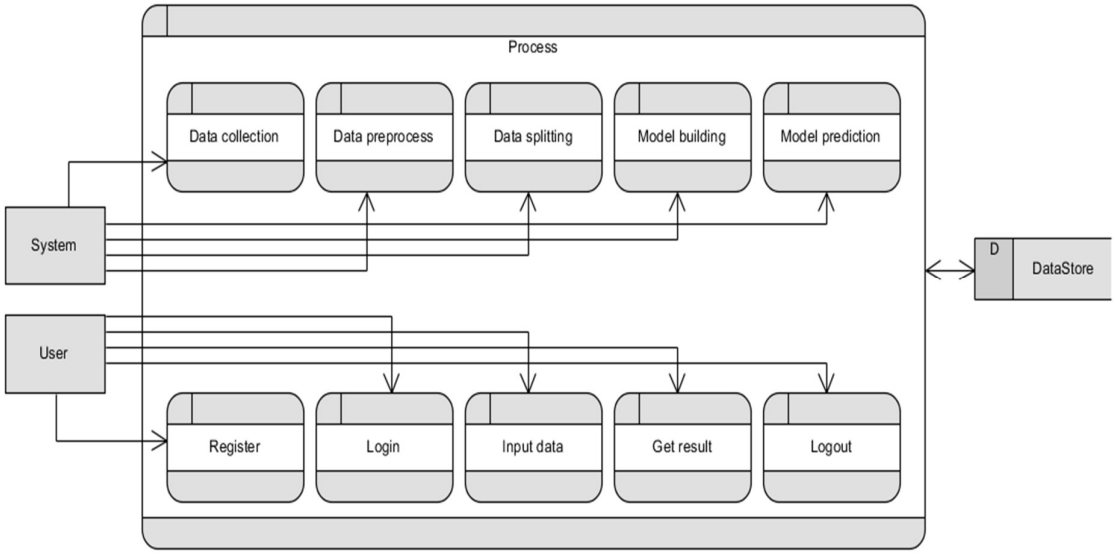
Contrast Level:



Level 1 Diagram:



Level 2 Diagram:



7. IMPLEMENTATION AND RESULTS

7.1 Modules

1. System:

1.1Data Collection:

- **Source:** Collect thyroid disorder data from trusted sources such as Kaggle, which offers publicly available datasets on thyroid conditions.

1.2Data Preprocessing:

- **Missing Values:** Handle missing values by filling them with the mean, median, or mode. If necessary, drop records with missing data.
- **Drop Unwanted Columns:** Remove columns that are not relevant for model building or do not correlate with the target column.
- **Drop Duplicates:** Eliminate duplicate rows to ensure data quality and consistency.

1.3Feature Engineering:

- **Label Encoding:** Use 'LabelEncoder' to convert categorical variables into numerical format for machine learning algorithms.
- **Balancing the Data:** Implement the 'BOOST' Balancing Method to manage imbalanced datasets and ensure equal representation of all thyroid disorder classes.
- **Select K Best Features:** Apply the 'SelectKBest' method to identify and select the most relevant features from the dataset, improving model performance by reducing dimensionality.

1.4Data Splitting:

- **Model Training:** Use 80% of the preprocessed dataset for training the machine learning models. The training process will involve optimizing model parameters to improve classification accuracy for thyroid disorders.
- **Model Testing:** Reserve the remaining 20% of the dataset for testing the model's performance using metrics like accuracy, precision, recall, and F1-score.

1.5 Model Integration:

- **Classification Algorithms:** Decision Trees, SVM, KNN, Random Forest, AdaBoost, Gradient Boosting, Catboost and Xgboost.
- **Dynamic Selection Mechanism:** Implement a dynamic selection mechanism to choose the best-performing model based on real-time data inputs**.**

1.6Evaluation and Validation:

- Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
- Validate the model using test datasets to assess its generalization capability and ensure it performs well on unseen data.

1.7Model Prediction:

- The trained model will predict thyroid disorder conditions for the input data, providing specific classifications such as hyperthyroidism, hypothyroidism, and specific sub-conditions.
- Predictions:

    1) hyperthyroid conditions:

        i. Subclinical (initial)
        ii. T3 toxic
        iii. toxic goiter
        iv. secondary toxic

    2) hypothyroid conditions:
        i. Subclinical (initial)
        ii. primary hypothyroid
        iii. compensated hypothyroid
        iv. secondary hypothyroid

2. User:

2.1Register:

- Users must first register by entering their credentials such as username, email, password to create an account in the system.

2.2Login:

- Users log in with their registered credentials to access the system.

2.3Input Data:

- Users input their thyroid-related data into the system for prediction.

2.4Viewing Results:

- The system processes the input data using the trained model, and users can view prediction.

2.5Logout:

- Users can log out of the system to secure their session and personal data.

7.2 Techniques Used

7.2.1 Label Encoding

Objective: Convert categorical labels into numerical values to make them suitable for machine learning algorithms.

In this project, Label Encoding is utilized to transform categorical variables related to thyroid conditions into numerical values. This step is essential for preparing the data for machine learning models, as most algorithms require numerical input to process and learn from the data effectively.

Specifically, Label Encoding is applied to various categorical features within the dataset, such as sex, on_thyroxine, query_on_thyroxine, sick, pregnant, thyroid_surgery, query_hypothyroid, query_hyperthyroid, and more. Each of these columns contains categorical labels representing different conditions or states, and they need to be converted into integer values before model training.

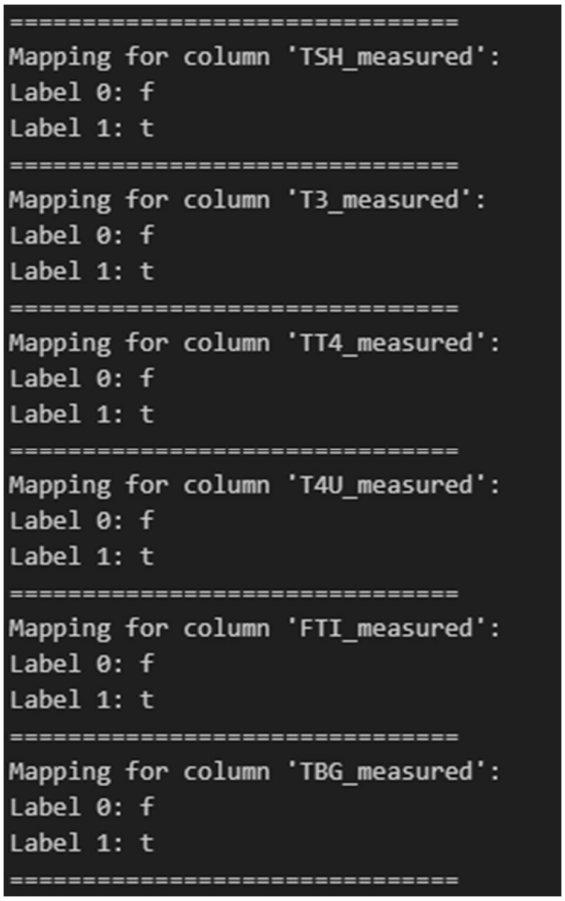
The LabelEncoder from the sklearn.preprocessing module is employed to carry out this transformation. For each unique category in the identified columns, LabelEncoder assigns a unique integer. This is done by first initializing the LabelEncoder and then fitting it to the specific column of the dataset. The categories are then mapped to corresponding integer values, which replace the original text labels in the dataset.

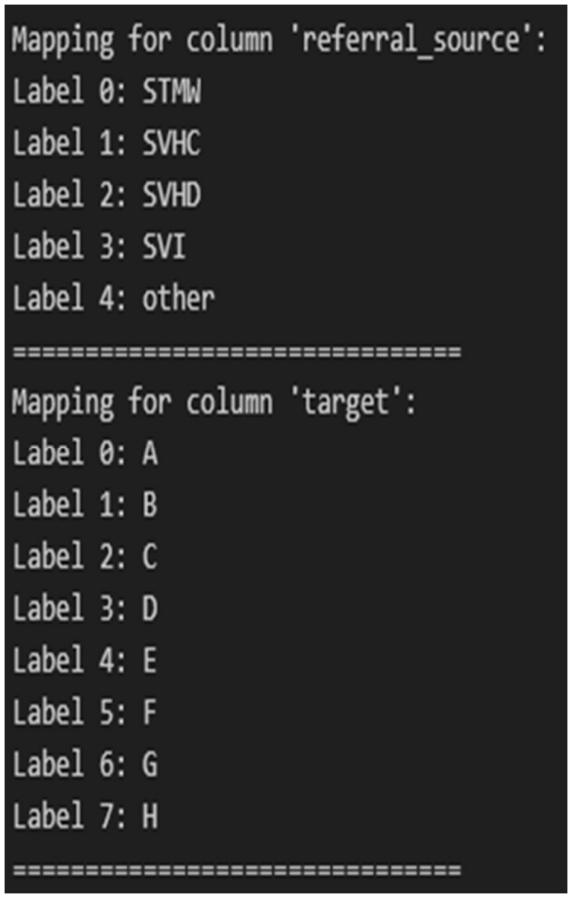
Implementation Steps:

1. **Identify categorical columns**: The first step is identifying all categorical columns that need to be encoded, such as the columns representing different medical conditions and treatments.
2. **Initialize and apply LabelEncoder**: The LabelEncoder() is initialized, and for each categorical column, it is fitted and transformed to convert the categorical values into numerical ones.
3. **Mapping of categorical values**: Each unique value in the categorical columns is assigned a corresponding integer, and the mapping between the original categories and their numeric equivalents is documented. This helps maintain clarity and allows for reverse transformation if needed.

```
Mapping for column 'sex':
Label 0: F
Label 1: M
==============================
Mapping for column 'on_thyroxine':
Label 0: f
Label 1: t
==============================
Mapping for column 'query_on_thyroxine
Label 0: f
Label 1: t
==============================
Mapping for column 'on_antithyroid_meds
Label 0: f
Label 1: t
==============================
Mapping for column 'sick':
Label 0: f
Label 1: t
==============================
Mapping for column 'pregnant':
Label 0: f
Label 1: t
==============================
Mapping for column 'thyroid_surgery':
Label 0: f
Label 1: t
==============================
```

```
Mapping for column 'I131_treatment':
Label 0: f
Label 1: t
==============================
Mapping for column 'query_hypothyroid':
Label 0: f
Label 1: t
==============================
Mapping for column 'query_hyperthyroid':
Label 0: f
Label 1: t
==============================
Mapping for column 'lithium':
Label 0: f
Label 1: t
==============================
Mapping for column 'goitre':
Label 0: f
Label 1: t
==============================
Mapping for column 'tumor':
Label 0: f
Label 1: t
==============================
Mapping for column 'hypopituitary':
Label 0: f
Label 1: t
==============================
```


==========
Mapping for column 'TSH_measured':
Label 0: f
Label 1: t
==========
Mapping for column 'T3_measured':
Label 0: f
Label 1: t
==========
Mapping for column 'TT4_measured':
Label 0: f
Label 1: t
==========
Mapping for column 'T4U_measured':
Label 0: f
Label 1: t
==========
Mapping for column 'FTI_measured':
Label 0: f
Label 1: t
==========
Mapping for column 'TBG_measured':
Label 0: f
Label 1: t
==========


Mapping for column 'referral_source':
Label 0: STMW
Label 1: SVHC
Label 2: SVHD
Label 3: SVI
Label 4: other
==========
Mapping for column 'target':
Label 0: A
Label 1: B
Label 2: C
Label 3: D
Label 4: E
Label 5: F
Label 6: G
Label 7: H
==========

7.2.2 Select KBest Technique

Objective: Select the most relevant features from the dataset to improve model performance by reducing dimensionality and eliminating irrelevant or redundant features.

In this project, SelectKBest is employed to select the top features that have the strongest relationship with the target variable. By identifying and retaining only the most informative features, the model can focus on the key variables that contribute most to the prediction of thyroid conditions. This helps improve the performance of the machine learning models, reduces overfitting, and speeds up the training process.

SelectKBest works by applying a statistical test to each feature and assigning a score. The features with the highest scores are selected for use in the model, while those with lower scores (less significant for the target variable) are discarded. This method is particularly useful for large datasets with many features, where some features may have little or no predictive value.

In this project, SelectKBest is implemented using the f_classif function from the sklearn.feature_selection module, which is based on the ANOVA F-test. This test measures the correlation between each feature and the target variable, ranking them by their significance.

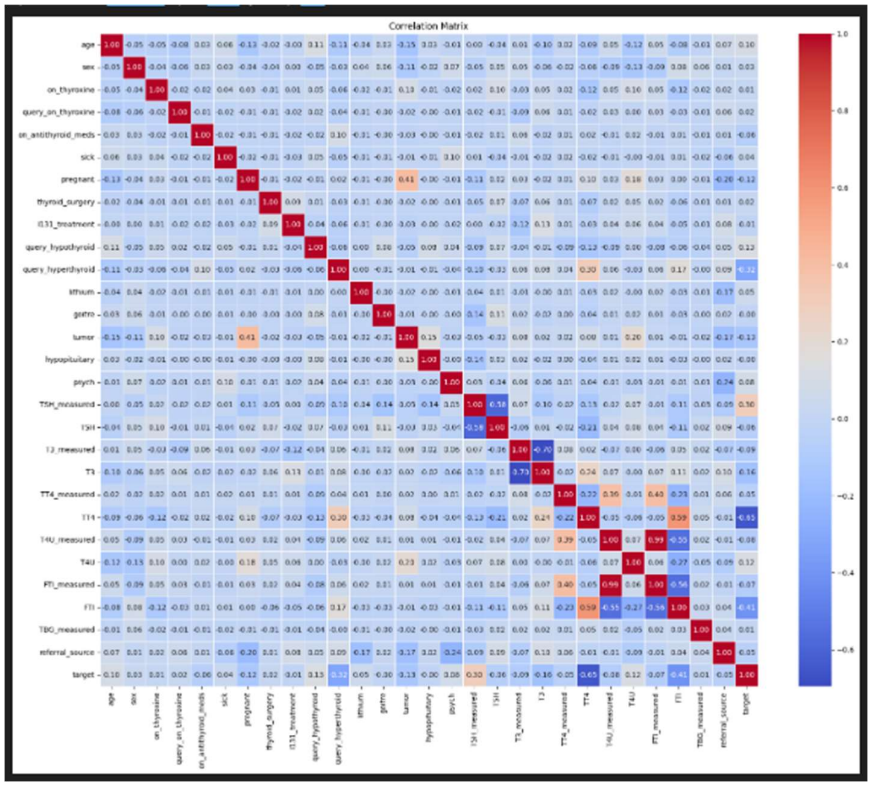


Figure: Correlation matrix

Implementation Steps:

1. **Initialize SelectKBest**: We initialize SelectKBest by specifying the statistical test (f_classif) and the number of top features (k) to be selected. In this case, k is set based on experimentation, keeping the most relevant features for classifying thyroid conditions.
2. **Fit and transform the data:** After initializing SelectKBest, we fit it to the training dataset to calculate the scores for each feature. The features with the highest scores are retained, and the remaining features are discarded.
3. **Review selected features:** Once the transformation is complete, the model retains only the top k features. These selected features represent the most important variables related to thyroid disorder diagnosis. The feature importance scores are also reviewed to ensure the model is focusing on the most predictive features.
4. **Feature ranking and mapping:** The mapping between the original feature names and their scores is documented to maintain clarity. This allows us to track which features were selected and why they were considered important.

```
# view thedataframe
df_top_10
```

	TT4	tumor	FTI	TSH_measured	pregnant	TSH	query_hyperthyroid	T4U	on_antithyroid_meds	query_hypothyroid
0	48.0	0	47.0	1	0	68.000000	0	1.02	0	0
1	157.0	0	176.0	1	0	0.050000	0	0.89	0	0
2	33.0	0	31.0	1	0	140.000000	0	1.07	0	0
3	114.0	0	136.0	1	0	9.799999	0	0.84	0	0
4	7.5	0	7.5	1	0	90.000000	0	0.94	0	0
...	...	...	...	...	...	...	...	...	...	...
914	16.0	0	15.0	0	0	142.554587	0	1.10	0	1
916	16.0	0	15.0	0	0	482.851302	1	1.10	0	1
917	16.0	0	15.0	0	0	328.322221	0	1.10	0	1
918	16.0	0	15.0	0	0	362.792842	0	1.10	0	1
919	16.0	1	15.0	0	0	127.110510	0	1.10	0	1

Figure: Top 10 selected features

7.2.3BOO-ST balancing method

The BOO-ST balancing method is a three-stage process designed to address the issue of class imbalance in datasets, particularly in the context of machine learning. The acronym BOO-ST stands for **BS (Boosting with Sample Weighting)**, **SMOTE (Synthetic Minority Over-sampling Technique)**, and **Tomek Links (TL)**. The goal of this approach is to improve the representation of

minority classes in imbalanced datasets while minimizing the introduction of noisy or irrelevant data, which is a common issue with other oversampling techniques.

Here's a breakdown of the BOO-ST method:

- **SMOTE: Balance the dataset by oversampling the minority class.**
- **Tomek Links: Remove noisy points to improve class separation.**
- **Boosting: Train our model on the refined dataset.**

  Outcome:

- The BOO-ST method results in a more balanced dataset, with synthetic instances that better represent the minority class and reduced noise.

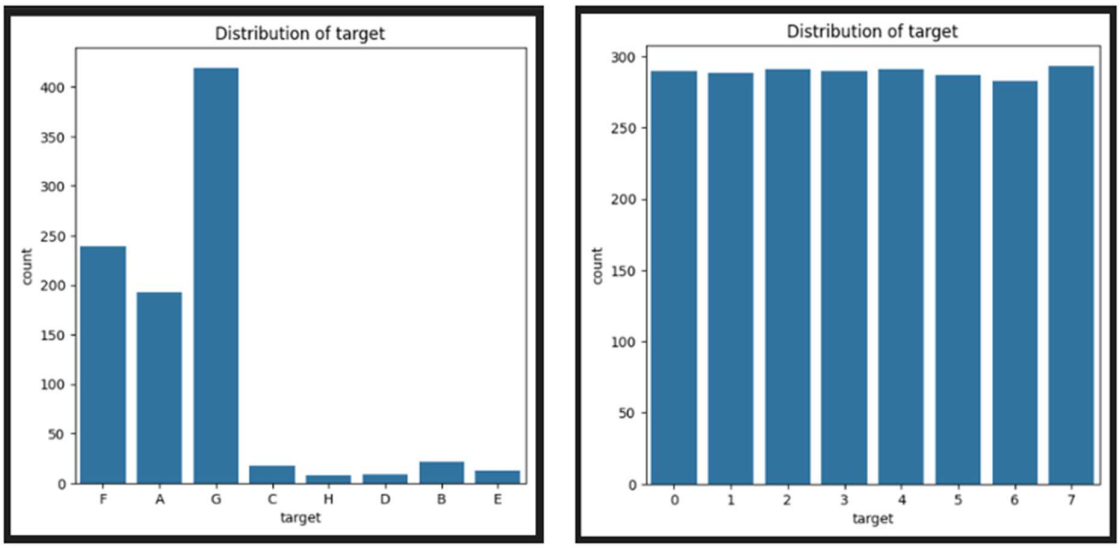
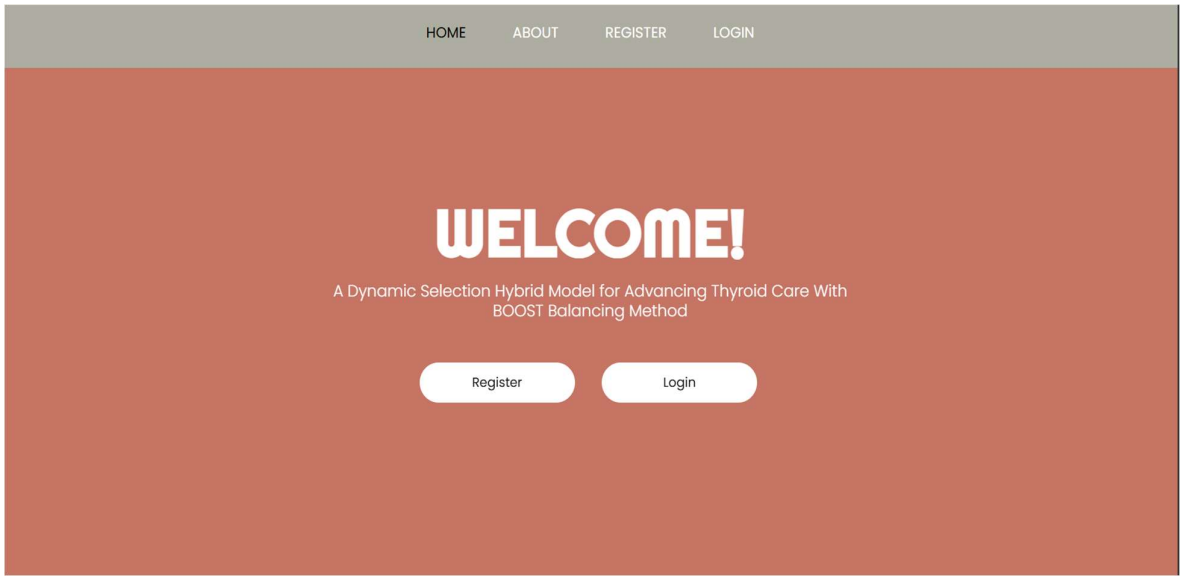


Figure: Distribution graph for target column before and after implementing BOO-ST technique.

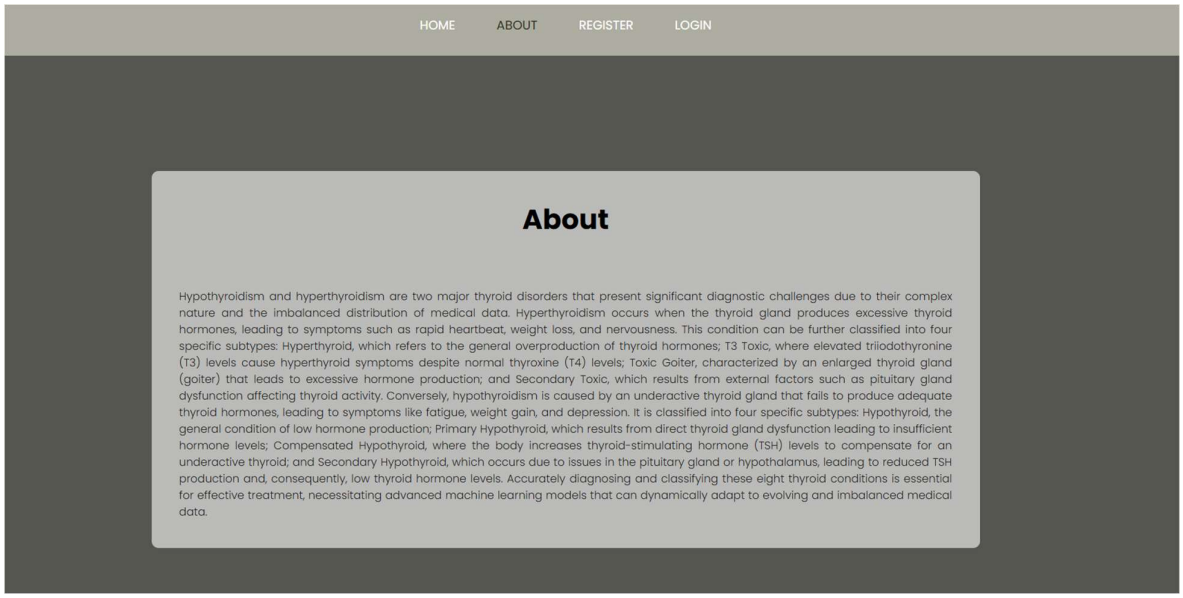
In summary, the BOO-ST method is a systematic approach to handle class imbalance by first weighting minority samples, then generating synthetic minority instances with SMOTE, and finally refining the dataset using Tomek Links to eliminate noisy majority samples. This combination allows for more robust handling of imbalanced data and improves model performance on the minority class.

7.3Output Screens

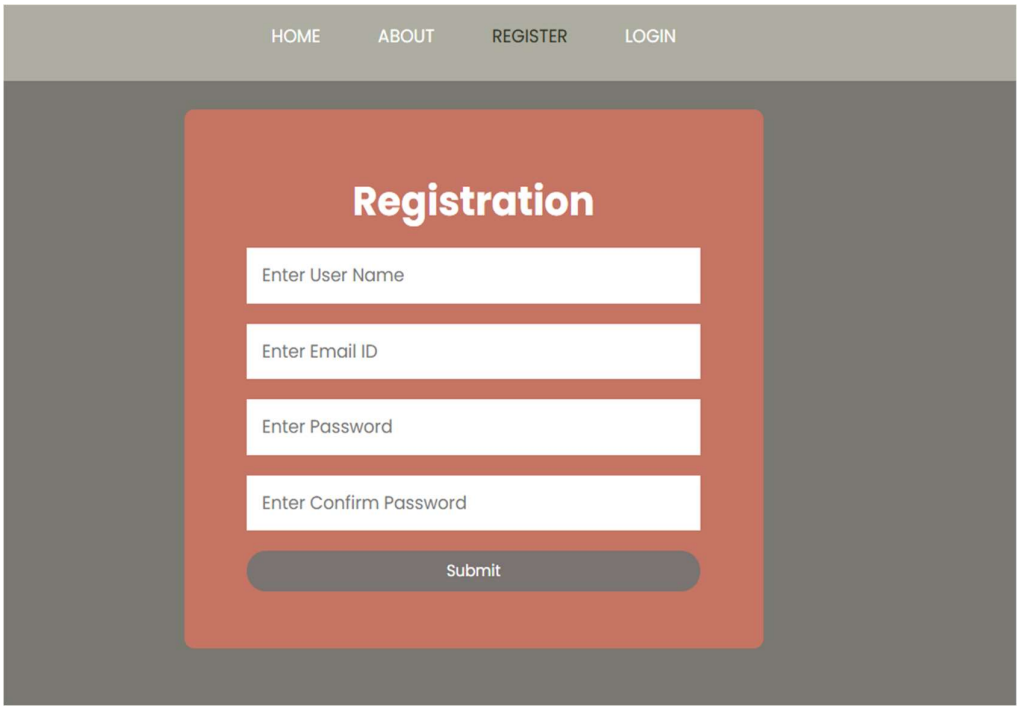
HomePage: The HomePage serves as the landing page of your application.



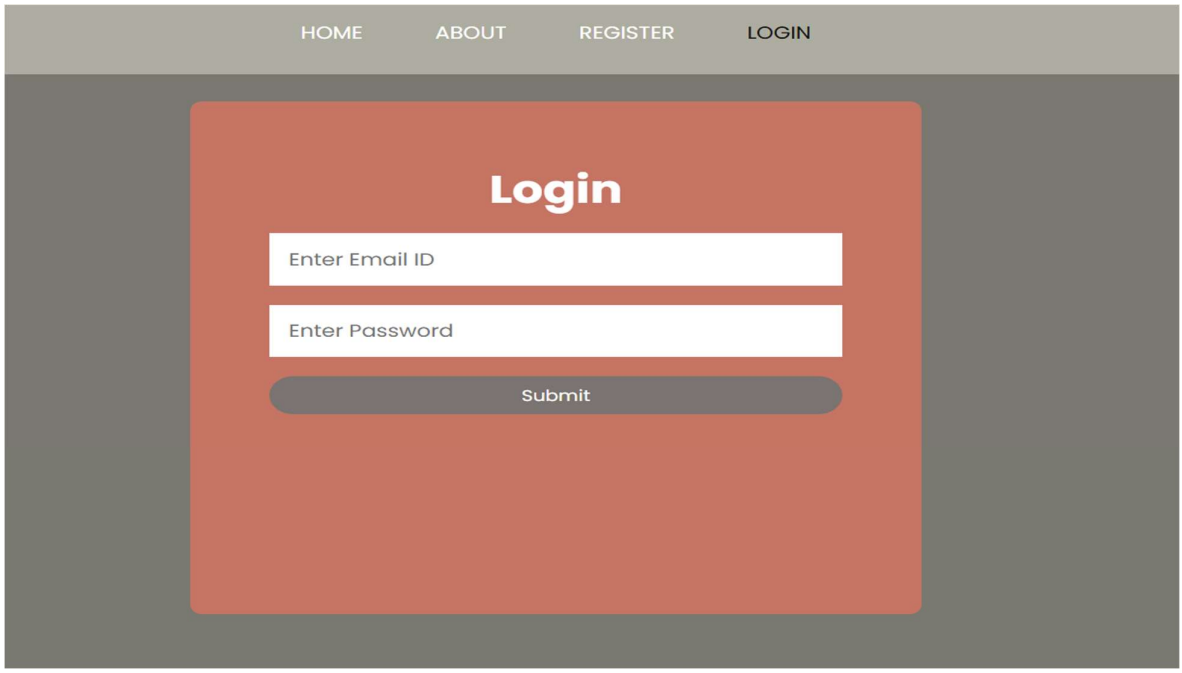
About Page: The About Page offers detailed information about the project, including its purpose, goals, and the technology used.



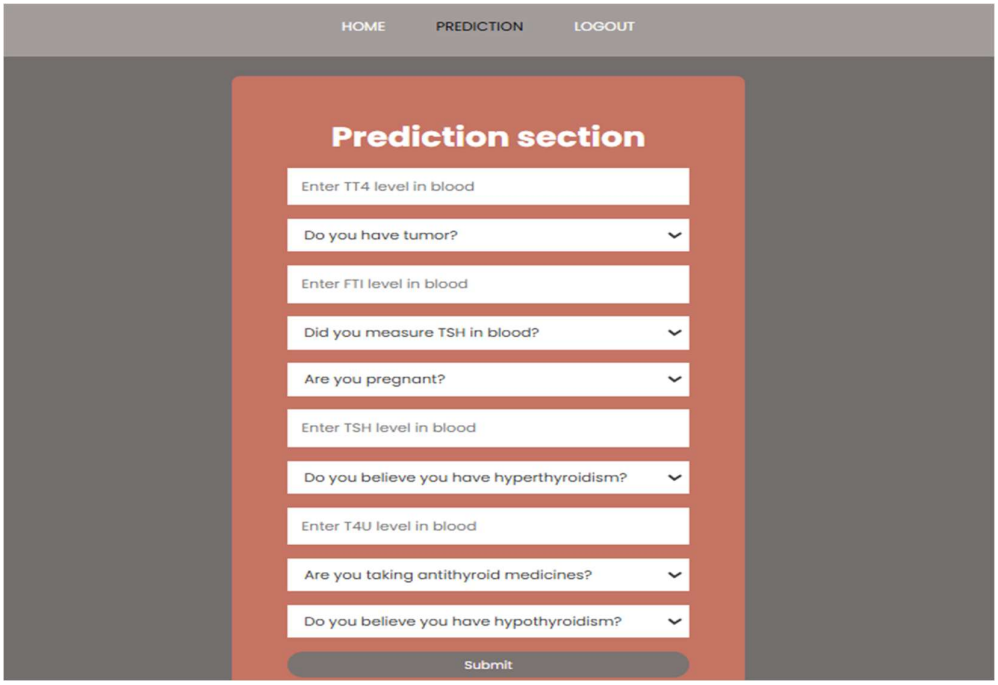
Registration Page: The Registration Page allows new users to create an account with the application.



Login Page: The Login Page enables users to access their existing accounts by entering their credentials. It usually includes fields for entering a username/email and password.

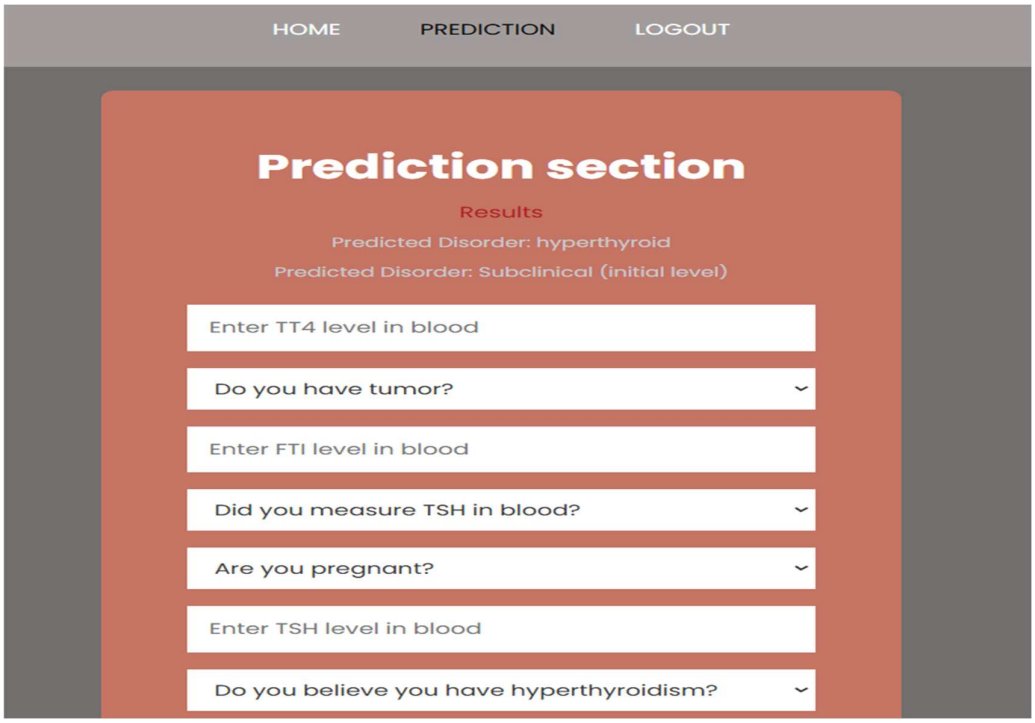


Prediction Page: The Prediction Page allows users to input data and receive predictions based on the trained machine learning models. This page typically includes a form or interface for uploading or entering data.

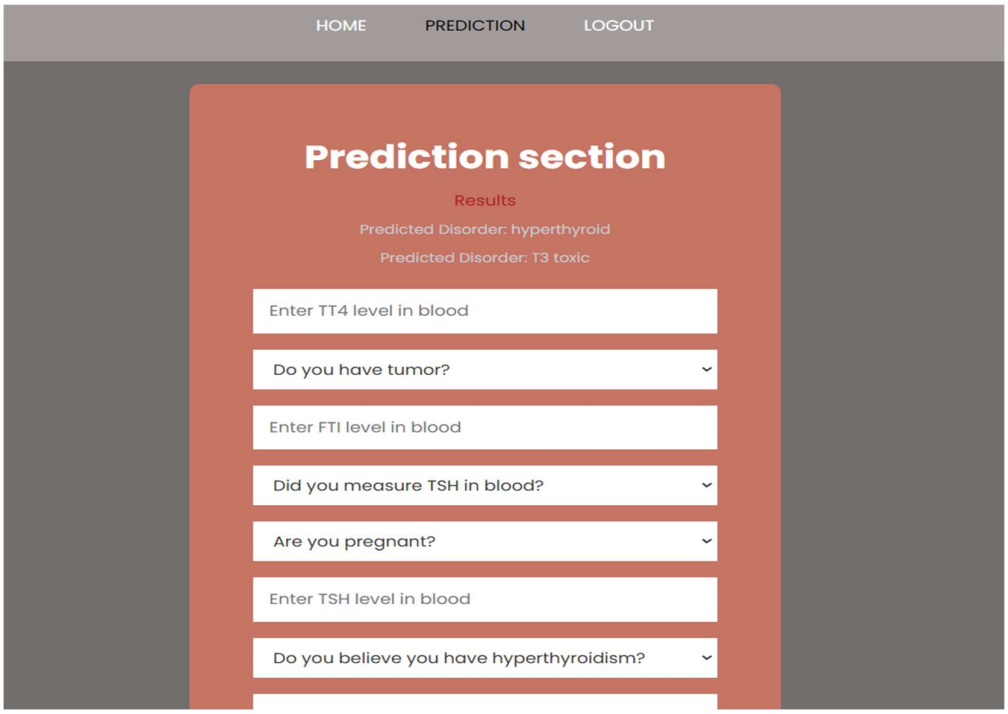


Result Page: In here result will be display which is predicted by our trained model.

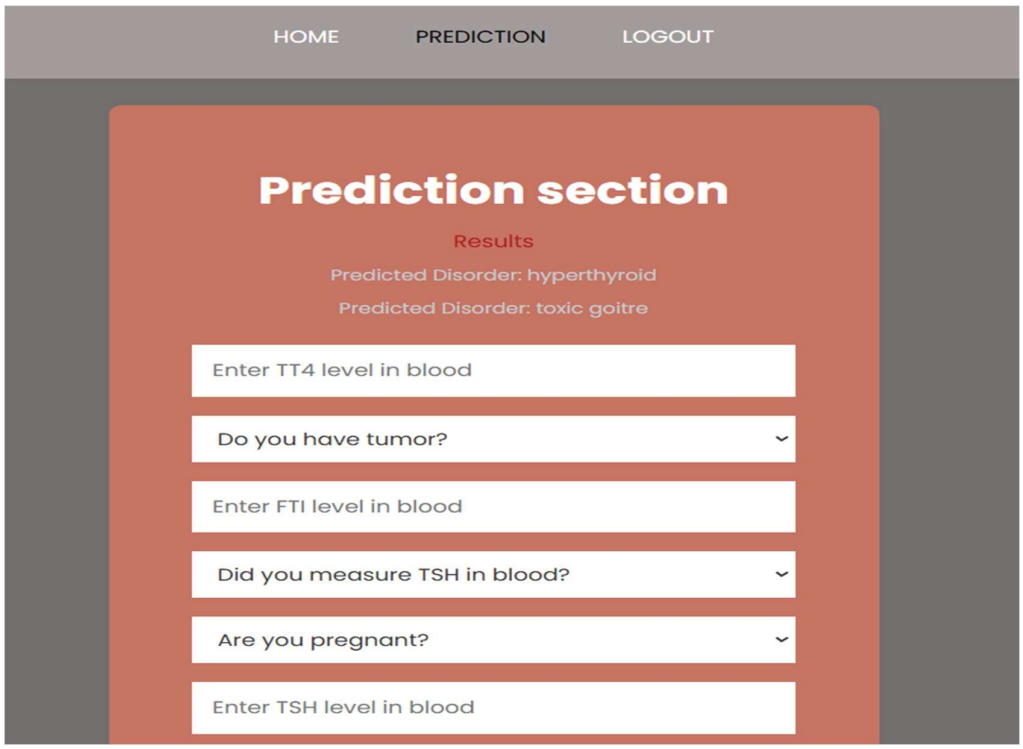
Predictions:disoreder–hyperthyroid, condition - Subclinical (initial).



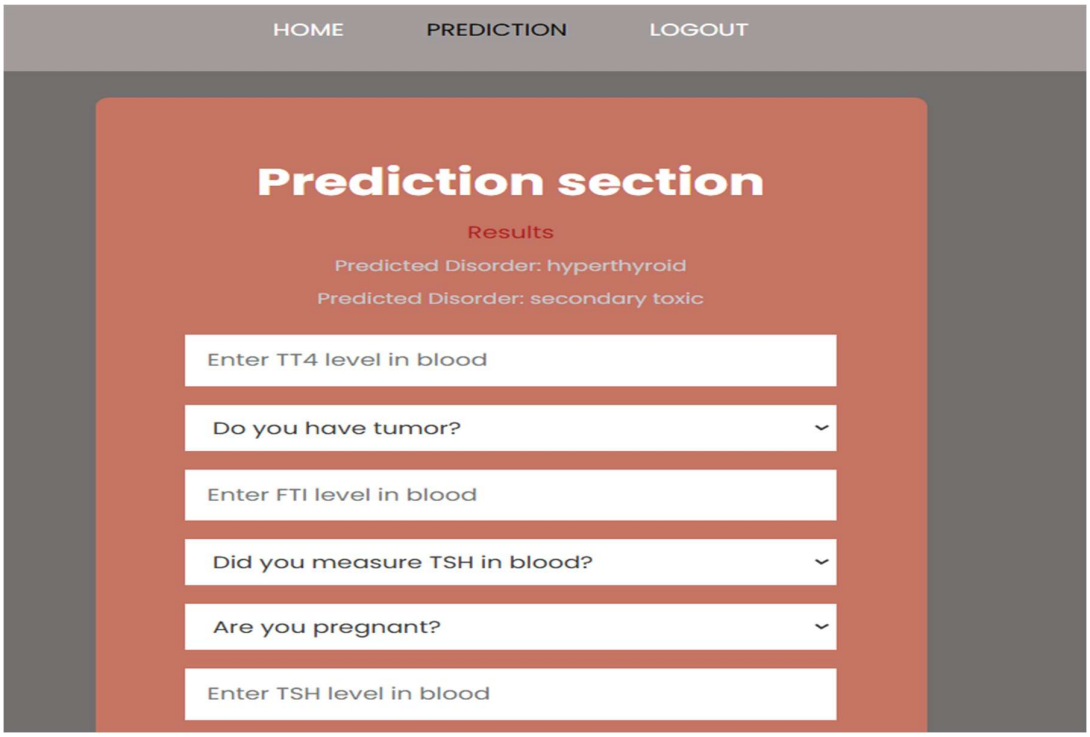
Predictions:disoreder–hyperthyroid, condition - T3 toxic.



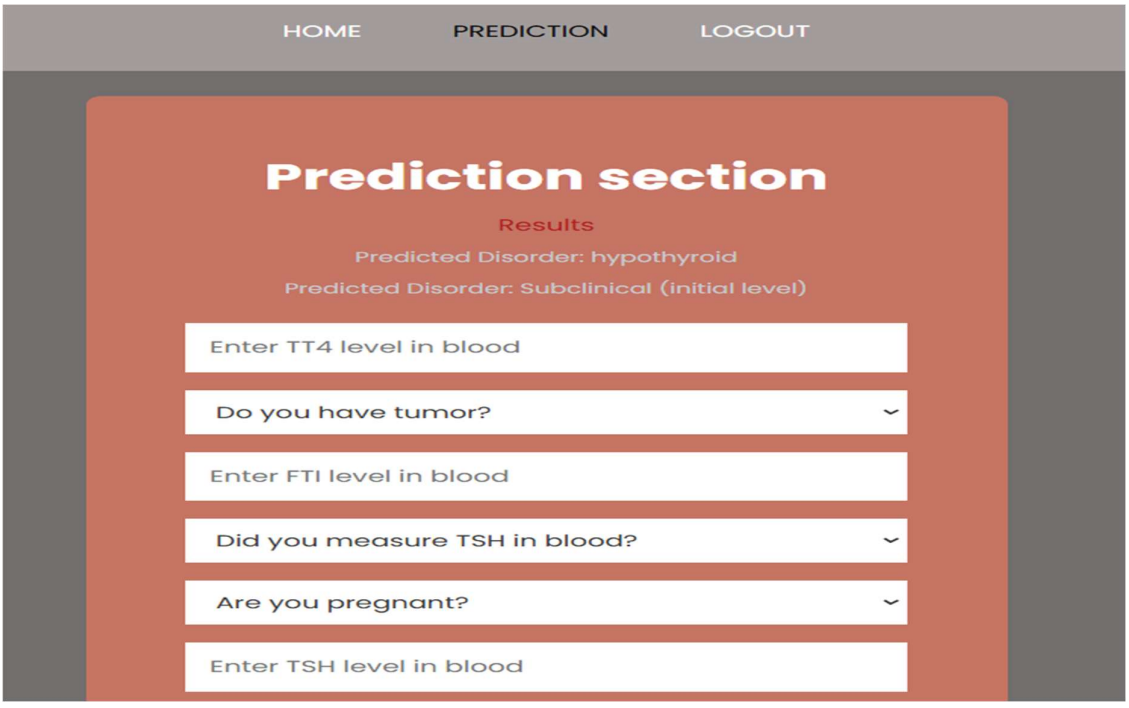
Predictions:disoreder–hyperthyroid, condition - toxic goitre.



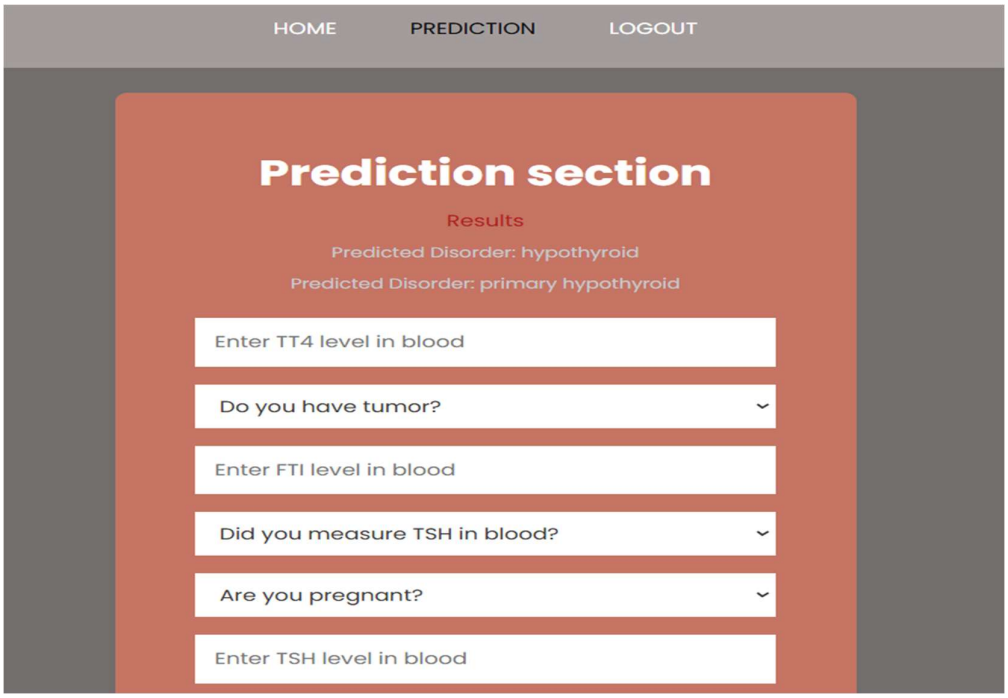
Predictions:disoreder–hyperthyroid, condition - secondary toxic.



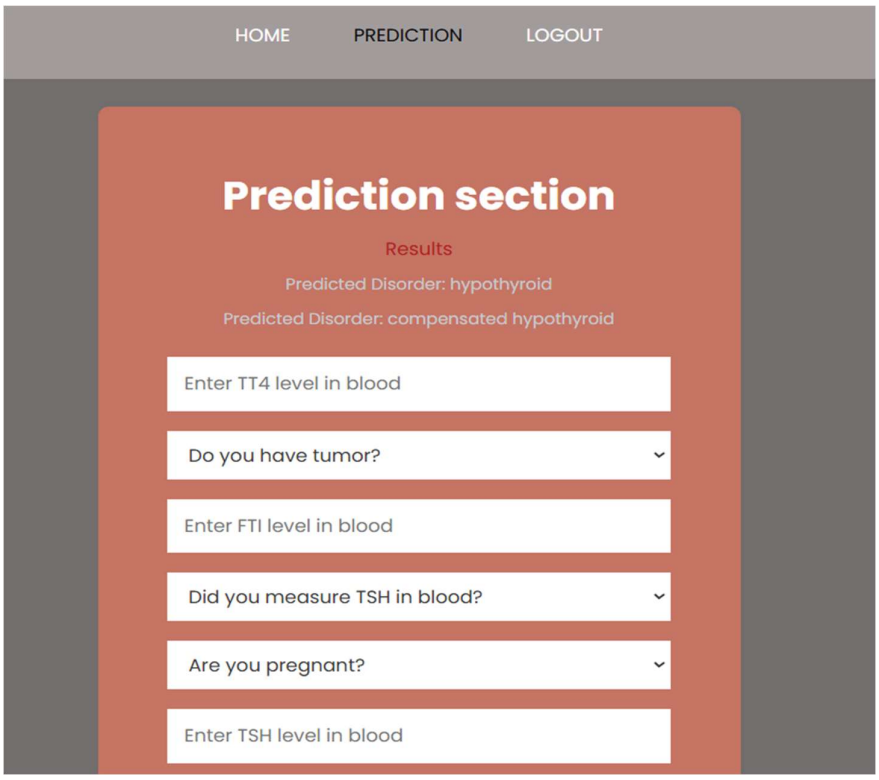
Predictions:disoreder–hypothyroid, condition - Subclinical (initial).



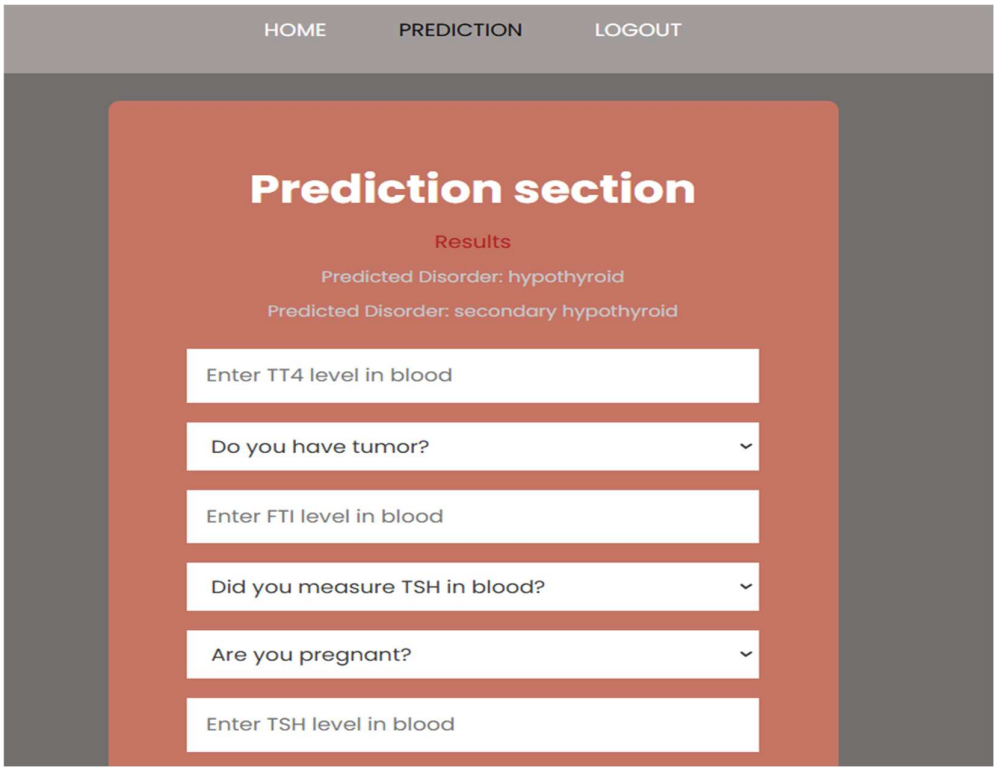
Predictions:disoreder–hypothyroid, condition - primary hypothyroid.



Predictions:disoreder–hypothyroid, condition - compensated hypothyroid.



Predictions:disoreder–hypothyroid, condition - secondary hypothyroid.



8. SYSTEM STUDY AND TESTING

The objective of system testing is to ensure that the thyroid disorder diagnosis system performs as expected and meets the specified requirements. The testing process is aimed at identifying and rectifying any errors or weaknesses in the system, ensuring reliable and accurate predictions for various thyroid conditions.

8.1 Feasibility Study

Technical Feasibility: The technical feasibility of this project is high due to the availability of advanced machine learning libraries and tools such as scikit-learn, Pandas, and Numpy, which support the various algorithms used in this project (Decision Tree, SVM, KNN, Random Forest, AdaBoost, and Gradient Boosting). The implementation of the BOOST Balancing Method is supported by these libraries, making it straightforward to handle imbalanced datasets. The tools required for data preprocessing, model building, and evaluation are well-established, ensuring compatibility with current technological infrastructure.

Economic Feasibility: The economic feasibility of this project is promising. By automating the diagnosis of thyroid disorders using machine learning, healthcare providers can reduce the time and cost associated with manual diagnosis, improving operational efficiency. The initial development costs are outweighed by the potential savings in diagnostic processes and the improved quality of care. This investment is likely to result in long-term cost benefits.

Operational Feasibility: Operationally, the system is feasible to implement. The development team has expertise in machine learning, software development, and medical data processing. Additionally, the system can be easily integrated into clinical workflows, providing real-time thyroid condition predictions. The project is designed to be user-friendly and adaptable, aligning well with operational requirements in medical settings.

Overall, the project is highly feasible from a technical, economic, and operational perspective, and holds great potential for enhancing the accuracy and speed of thyroid disorder diagnosis.

8.2 Types of Tests & Test Cases

- **Unit Testing**: Unit testing focuses on validating individual components of the thyroid diagnosis system to ensure that each unit performs as expected. This involves testing data preprocessing functions (e.g., handling missing values, label encoding), the implementation of the BOOST Balancing Method, and the accuracy of individual machine learning models. Each test case verifies specific input and output relationships, ensuring that the code behaves correctly.

  Test Cases:

  - Verify that missing values are handled correctly by the preprocessing pipeline.
  - Ensure that categorical features are correctly encoded into numerical values.
  - Check the correctness of the SelectKBest feature selection process.

- **Integration Testing**: Integration testing verifies that the various components of the system work together as intended. This includes the integration of data preprocessing with the machine learning models, and the interaction between the model predictions and the user interface. The goal is to ensure smooth data flow and proper functioning across all modules.

  Test Cases:

  - Ensure that preprocessed data is correctly passed to the classification models.
  - Validate that model outputs are integrated into the user interface for display.
  - Check that predictions are returned in real-time when new data is input.
- **Functional Testing**: Functional testing ensures that the system meets the specified business and technical requirements, including accurate predictions for the eight thyroid conditions. This involves validating the system's ability to handle different types of data inputs and produce accurate predictions.

  Test Cases:

  - Validate that the system correctly handles valid and invalid input data for thyroid parameters.
  - Ensure that the key functionalities, such as predicting thyroid conditions (e.g., hyperthyroidism or hypothyroidism), are operating as expected.
  - Check that the system can accurately classify thyroid conditions across the eight specific categories (A to H).
- **White Box Testing**: White box testing examines the internal workings of the code. In this project, the implementation of machine learning algorithms (such as Random Forest, AdaBoost, etc.) is tested to ensure that they function correctly. It involves checking the decision paths in models and verifying that the BOOST Balancing Method correctly handles imbalanced datasets.

  Test Cases:

  - Validate that the decision tree algorithm splits correctly and outputs the expected predictions.
  - Check the internal logic of the BOOST method to ensure correct application of SMOTE and Tomek Links.
- **Black Box Testing**: Black box testing evaluates the system without knowledge of its internal workings. The primary focus is on verifying the system's functionality through the user interface, ensuring that predictions for thyroid conditions are generated correctly and that user inputs are handled appropriately.

  Test Cases:

- Ensure the system responds correctly to user input (e.g., thyroid test results) and provides accurate predictions.
- Validate the correctness and responsiveness of the user interface elements, such as input forms and result displays.

- **Test Objectives**:
  - Verify that all input fields related to thyroid condition data are correctly validated and processed.
  - Ensure that navigation and interaction elements (e.g., submitting data, retrieving results) function correctly.
  - Confirm that the system provides accurate predictions for thyroid disorders in a timely manner.
- **Features to be Tested**:
  - **Input Validation**: Check that all input data (e.g., TSH, TT4 levels) is in the correct format and that invalid data is handled appropriately.
  - **Duplicate Entries**: Ensure that duplicate records are not allowed in the system to maintain data integrity.
  - **Prediction Accuracy**: Test the accuracy of the model's thyroid condition predictions across the eight categories.
  - **Model Training**: Ensure that the machine learning models are correctly trained and retrained as new data becomes available.
  - **Navigation**: Verify that all links and buttons in the user interface direct users to the correct functions, such as result retrieval or input pages.

By systematically conducting these tests, the project ensures that the thyroid disorder diagnosis system is accurate, reliable, and user-friendly, ultimately contributing to more effective and timely diagnosis of thyroid conditions.

8.3 Test cases

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the dataset.	Dataset path.	Dataset need to read successfully.	Dataset fetched successfully.	P
2	Performing Loading on the dataset	Data loading takes place	Data loading should be performed on system	Data loading successfully completed.	P
3	Performing data	Dataset is provided to	Processed data should	Processed data will	P

	preprocessing	process the data	predict either of two classes	successfully completed	
4	Model Building	Model Building for the clean data	Need to create model using required algorithms	Model Created Successfully.	P
5	Predict classification	Input is provided with numerical and categorical values	Based on the input data prediction is done	Predicted successfully	P

9. CONCLUSION

This project successfully developed a Dynamic Selection Hybrid Model for the accurate and efficient diagnosis of thyroid disorders, addressing the challenges of imbalanced medical data and the complexity of thyroid conditions. By integrating multiple machine learning algorithms such as Decision Trees, SVM, KNN, Random Forest, AdaBoost, Gradient Boosting,Xgboost and catboost the model is able to dynamically select the most effective classifier in real-time, ensuring higher diagnostic accuracy.

One of the key contributions of this project is the implementation of the BOOST Balancing Method, which combines Boosting with Sample Weighting (BS), SMOTE (Synthetic Minority Over-sampling Technique), and Tomek Links. This technique effectively handled the class imbalance problem often found in thyroid disorder datasets, improving classification performance, especially for minority classes such as rarer thyroid conditions. As a result, the system is capable of classifying thyroid disorders across eight specific categories of hyperthyroid and hypothyroid conditions with greater precision.

This project demonstrates the potential of machine learning models in healthcare, offering a scalable and efficient tool for medical professionals. By automating the diagnosis process, the system not only enhances the accuracy of predictions but also reduces the time required for diagnosis, ultimately improving patient outcomes. In conclusion, the Dynamic Selection Hybrid Model represents a significant advancement in thyroid care and lays the foundation for future applications of machine learning in the medical domain.

10. FUTURE ENHANCEMENT

As with any technological solution, the Dynamic Selection Hybrid Model for thyroid disorder diagnosis has potential for future enhancements to further improve accuracy, scalability, and usability. Below are some key areas where this project could be enhanced in the future:

- Incorporation of Additional Medical Data: Enhancing Dataset Diversity: The current model relies primarily on clinical test results and basic demographic information. Future versions could incorporate more comprehensive patient data, including genetic information, family medical history, and lifestyle factors, to improve predictive accuracy and provide more personalized diagnostic recommendations.
- Deep Learning Models: Implementation of Deep Learning: Although this project uses classical machine learning algorithms, future iterations could explore deep learning approaches such as Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) for enhanced prediction. These models could capture complex relationships in large datasets and provide even higher accuracy for diagnosing thyroid disorders.
- Incorporating More Classes of Thyroid Disorders: Expanding the Range of Disorders: Currently, the model focuses on eight specific thyroid conditions. Future enhancements could broaden the classification scope to cover other thyroid-related disorders, such as autoimmune thyroiditis or thyroid cancer. This would make the model more comprehensive in terms of thyroid disease diagnosis.
- Multi-Language Support and Global Expansion: Expanding the system to support multiple languages and adapting it to different healthcare environments globally would increase the system’s usability in diverse regions. This enhancement would involve translating the user interface and reports, as well as ensuring that local medical standards and practices are supported.
- Explainable AI (XAI): Future improvements could focus on making the system more interpretable by providing explanations for each prediction made by the model. This would build trust with healthcare professionals, allowing them to understand why certain predictions were made, especially in complex cases.

11. REFERENCES

1. G. Polat and S. Güneş, "An Expert System Approach Based on Principal Component Analysis and Adaptive Neuro-Fuzzy Inference System to Diagnosis of Thyroid Diseases," Expert Systems with Applications, vol. 32, pp. 1120–1128, 2007.
2. M. Singla, K. Tayal, and R. Kasana, "Thyroid Disease Diagnosis Using Machine Learning Algorithms," Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1488-1492, 2016.
3. D. V. Gaikwad, A. N. Yadav, and M. D. Patil, "Thyroid Disease Prediction Using Data Mining Techniques," International Journal of Engineering Research & Technology, vol. 7, no. 10, pp. 21-24, 2018.
4. P. Agarwal, S. Gupta, and M. P. Dave, "Ensemble Learning Model for Thyroid Disease Classification," Journal of Medical Systems, vol. 42, no. 4, 2018.
5. A. K. Singh and R. M. O. Almasi, "Handling Imbalanced Data for Thyroid Disease Diagnosis Using SMOTE and Tomek Links," International Journal of Computational Intelligence and Applications, vol. 19, no. 1, 2020.
6. N. Uddin, S. A. A. Baig, and S. M. Hassan, "Thyroid Disease Diagnosis Using Machine Learning Algorithms: A Comparative Study," IEEE Access, vol. 9, pp. 83182-83194, 2021.
7. A. Sharma, M. A. Zaki, and S. Pathak, "Thyroid Disease Classification Using Ensemble Methods," International Journal of Computer Applications, vol. 178, no. 7, pp. 23-28, 2019.
8. C. A. Al-Aidaros and M. M. Omar, "Improving Thyroid Disease Classification Using Deep Learning Approaches," Biomedical Signal Processing and Control, vol. 59, pp. 101-110, 2020.
9. R. Kumar and S. Aggarwal, "Thyroid Disorder Prediction Using Hybrid Machine Learning Models," Journal of King Saud University - Computer and Information Sciences, vol. 33, pp. 713-723, 2021.
10. M. S. Hossain and M. T. Rahman, "Predicting Thyroid Disease Using Data Mining and Machine Learning Approaches," Computers in Biology and Medicine, vol. 101, pp. 1-10, 2018.