HEART DISEASE PREDICTION USING NEURAL NETWORKS AND MACHINE LEARNING

NEURAL NETWORKS- CSC65001_25FA Neural Networks- 12/09/2025

Dr. Soumyashree Sahoo
Quinnipiac University
Hamden,USA
Soumyashree.Sahoo@quinnipiac.edu

Vignesh Balaji Ravichandran
Quinnipiac University
Hamden,USA
Vignesh.Ravichandran@quinnipiac.edu

Thenmozhi Sekar
Quinnipiac University
Hamden,USA
Thenmozhi.Sekar@quinnipiac.edu

***Abstract*— Heart disease continues to be one of the major causes of death across the world, and early detection has a direct impact on survival and quality of life. This project explores how machine learning can be used to predict heart disease using standard clinical measurements. The work is carried out in two stages. First, the models focus on binary prediction—identifying whether a patient is likely to have heart disease or not. After establishing a strong baseline, the project expands into a more detailed three-class classification that categorizes patients into no disease, mild disease, and moderate-to-severe disease.**

Several machine learning algorithms were tested, including a simple Neural Network, Random Forest, and XGBoost. After tuning, the Random Forest model produced the strongest performance, achieving 98.36% accuracy and an excellent 0.9957 ROC-AUC, making it highly reliable for screening. The multiclass model also showed promising results with 78.33% accuracy and a 0.872 macro-AUC, demonstrating its potential to support more nuanced clinical decisions.

To make the system practical, a fully functional web application was also built. It accepts 13 input features from users and instantly returns a prediction based on the trained model. This combination of model experimentation, performance tuning, and deployment highlights how machine learning can support early detection and assist in healthcare decision-making.

Keywords—Neural Network, Random Forest,XGBoost.

I. INTRODUCTION

This Predicting heart disease early can significantly reduce health risks and guide patients toward timely treatment. With the availability of structured clinical datasets and advancements in machine learning, it is now possible to build predictive systems that analyze multiple medical parameters simultaneously and estimate the likelihood of disease.

The goal of this project is to design and implement a complete machine learning-based heart disease prediction system. The project begins with binary classification and later extends into multiclass classification to provide a more detailed picture of disease severity. The work involves data preprocessing, model development, evaluation, optimization, and deployment.

In addition to training predictive models, the project includes the creation of a user-friendly website where users can enter their clinical information and receive an immediate prediction. This brings the project beyond experimentation and into a usable application, demonstrating the real-world value of such predictive systems.

II. BACKGROUND/LITERATURE REVIEW

Machine learning has increasingly been adopted in medical research, especially for diagnosing heart disease. Prior studies often use datasets like the UCI Cleveland Heart Disease dataset, which includes a consistent set of clinical attributes. Research shows that ensemble models—particularly Random Forest—tend to perform well because they reduce overfitting and capture complex interactions among variables. Neural Networks can also be effective but usually require larger datasets and more careful tuning.

Class imbalance is another common challenge in medical prediction tasks. To address this, earlier works frequently apply techniques like SMOTE to balance the dataset before training. Encoding categorical variables, especially with one-hot encoding, is also essential to ensure algorithms can interpret non-numeric features correctly.

While many studies stop at binary classification, predicting disease severity can be far more informative for clinical use. However, fewer projects explore multiclass prediction due to the increased complexity and imbalance across categories. This project builds on existing research by implementing both binary and multiclass models and by integrating them into a functional web application that demonstrates how these techniques can be used in practice.

III. METHODOLOGY

The development of the heart disease prediction system followed a structured machine-learning workflow built primarily in Python, due to its rich ecosystem of scientific and machine-learning libraries. The methodology consists of multiple stages, beginning with exploratory analysis and ending with a fully deployed, interactive web application. Each step was implemented carefully to ensure reproducibility, accuracy, and alignment with real-world machine-learning practices.

A. Development Environment and Tools Used

The project was implemented using Python 3.10+, chosen for its simplicity and extensive support for data science. Key tools and libraries included:

i. NumPy – numerical computations

ii. Pandas – data manipulation and cleaning

iii. Matplotlib / Seaborn – visualization and exploratory analysis

iv. Scikit-learn – preprocessing, model training, hyperparameter tuning, performance evaluation

v. Imbalanced-learn (SMOTE) – class balancing

vi. XGBoost library – gradient boosting implementation

vii. Joblib – model serialization

viii. Flask – backend deployment for the web application

ix. HTML/CSS/JavaScript – frontend interface design

x. VS Code / PyCharm – development environment

xi. Git / GitHub – version control

These tools formed a stable and efficient pipeline for analysis, modeling, and deployment.

B. Preprocessing

Before model development, the dataset underwent a series of preprocessing steps to ensure the features were properly formatted and suitable for machine-learning algorithms. Since the data contained a mix of categorical and numerical attributes, each type required a different transformation approach. Categorical variables such as chest pain type, resting ECG results, and thalassemia were first converted into a machine-interpretable format using One-Hot Encoding. This method was chosen because it avoids introducing any artificial ordering between categories, which could otherwise mislead the models. The encoding was implemented using the OneHotEncoder class from scikit-learn.

Numerical attributes—including cholesterol, resting blood pressure, maximum heart rate, and ST depression—were standardized using scikit-learn's StandardScaler. Although scaling is not required for tree-based models like Random Forest or XGBoost, it played an important role in stabilizing the training of the neural network and provided a consistent numeric range across all features.

After transforming the data, an 80/20 train–test split was performed using train_test_split. This ensured a fair evaluation by keeping the test set isolated until the final stage of model assessment. During exploratory analysis, class imbalance became evident, particularly in scenarios where mild disease cases were underrepresented. To mitigate this issue, SMOTE (Synthetic Minority Oversampling Technique) from the imbalanced-learn library was applied to the training set. SMOTE generates synthetic samples for minority classes, helping the models avoid bias toward the majority class and improving their ability to correctly identify positive cases.

This unified preprocessing pipeline ensured that all models received clean, balanced, and properly encoded data, ultimately contributing to stronger and more reliable predictive performance.

C. Baseline Model Development

The first model explored in this project was a simple Neural Network built using the TensorFlow–Keras library. Although easy to implement, this model struggled due to the relatively small dataset and the limited depth of the architecture. The network failed to capture the non-linear relationships present in the clinical features, which resulted in very low predictive performance. Its accuracy was approximately 0.47, and the AUC score was close to 0.50, indicating that the model performed no better than random guessing. This confirmed that neural networks were not the ideal choice for this dataset and motivated the shift toward classical machine-learning algorithms better suited for structured tabular data.

D. Classical Machine-Learning Models

After the neural network baseline, two classical supervised learning models were implemented using the Scikit-learn and XGBoost libraries.

1) Random Forest Classifier

The Random Forest classifier was chosen for its ability to capture non-linear feature interactions and its robustness with smaller datasets. Even without tuning, the initial Random Forest model delivered strong performance. It handled categorical and continuous features effectively and significantly outperformed the neural network baseline.

2) XGBoost Classifier

XGBoost was also tested due to its reputation for strong performance on structured datasets. While it provided competitive accuracy and generalization, it required more extensive hyperparameter tuning and did not surpass the accuracy of the Random Forest model within the scope of this project.

E. Hyperparameter Tuning and Optimization

The greatest performance improvements came from optimizing the Random Forest model. Hyperparameter tuning was performed using Scikit-learn's GridSearchCV. The parameters tuned included the number of trees (n_estimators), maximum depth of each tree (max_depth), minimum samples required to split and form leaves (min_samples_split and min_samples_leaf), class weighting to address imbalance, and the number of features considered at each split (max_features).

When combined with improved preprocessing—particularly One-Hot Encoding for categorical variables and SMOTE oversampling for minority classes—the tuned Random Forest achieved excellent performance. The final model obtained an accuracy of 0.9836 and an AUC score of 0.9957, making it the strongest binary classifier produced in this work.

F. Multiclass Classification Methodology

To extend the binary model into a more clinically relevant system, the project later introduced a three-class classification task. The goal was to differentiate among no disease, mild disease, and moderate-to-severe disease cases.

This required adapting the Random Forest classifier into a One-vs-Rest (OVR) configuration. Additional preprocessing was performed to balance the mild and severe disease classes using SMOTE. Since multiclass datasets tend to be more imbalanced, macro-averaged metrics were used to ensure equal importance across all classes. Performance was evaluated using macro accuracy, macro AUC, and a three-class confusion matrix.

The final multiclass model achieved an accuracy of 0.7833 and a macro AUC of 0.8723, demonstrating its ability to capture meaningful distinctions in disease severity.

G. Model Evaluation and Metrics

Model evaluation was conducted using Scikit-learn's built-in metrics. These included accuracy, precision, recall, F1-score, ROC–AUC, confusion matrices, and classification reports. For the multiclass classifier, macro-averaged metrics were emphasized due to class imbalance. These evaluation tools provided a comprehensive view of performance, especially in assessing the model's ability to correctly identify heart-disease-positive cases.

IV. Dataset Preprocessing (Binary Classification)

Preprocessing was one of the most important steps in this project because the dataset contained a mix of numeric and categorical features, each with its own quirks. The 13 clinical attributes represented measurements such as cholesterol, maximum heart rate, resting blood pressure, chest pain type, fasting blood sugar, ST depression, and others.

A. Data Processing

Before developing any of the machine-learning models, the dataset was carefully preprocessed to ensure that the features were consistent, interpretable, and ready for training. Since the data contained a mix of numerical and categorical attributes, different transformations were applied depending on the type of each feature. Categorical variables such as chest pain type, resting ECG results, and thalassemia were converted into a usable format through One-Hot Encoding. This approach was chosen because it avoids assigning artificial numerical order to categories, which could mislead the models. By creating separate binary columns for each category, the encoded dataset allowed the models to learn distinctions between medical conditions more accurately.

Alongside categorical encoding, the numerical features were standardized to improve consistency. Although tree-based models like Random Forest and XGBoost do not require normalization, StandardScaler was applied to ensure that no single numerical feature dominated the optimization process—especially for the neural network, which is sensitive to feature scaling. This step helped stabilize training across all models and maintained uniform numeric ranges throughout the dataset.

Another important challenge in the dataset was class imbalance, particularly for the disease-positive class in the binary setting and the mild-disease class in the multiclass setting. Models trained on imbalanced data tend to favor the majority class, which can lead to poor recall and missed detections. To address this, SMOTE (Synthetic Minority Oversampling Technique) was used to generate realistic synthetic samples for the minority class. By balancing the distribution of labels in the training data, SMOTE helped improve the model's ability to detect heart disease cases—an essential requirement for medical applications.

Finally, the dataset was partitioned using a standard train–test split, where the training set was used for learning model parameters and the test set was reserved strictly for final evaluation. During neural-network experimentation, part of the training data was further set aside as a validation split to monitor for overfitting. This structured splitting ensured that all models were evaluated fairly and that the performance metrics reflected their true generalization capability.

V. Model Development and Evaluation

After completing the preprocessing steps, the next stage of the project focused on developing and evaluating several machine-learning models. The modeling phase was intentionally structured to begin with a simple baseline and gradually progress toward more sophisticated and better-performing algorithms. Three initial models were selected: a Neural Network, a Random Forest classifier, and an XGBoost

classifier. Each model brought different strengths and helped in understanding how well the dataset responded to various learning approaches.

The Neural Network served as the baseline model. A simple two-layer architecture was used to evaluate how deep learning would perform on this dataset. Although the model trained without technical issues, its predictions were largely ineffective. It produced an accuracy of approximately 0.47 and an AUC of 0.50, meaning it performed no better than random guessing. This outcome revealed two major limitations: first, neural networks generally require larger datasets to learn meaningful patterns; second, the network struggled to capture complex feature relationships present in the structured clinical data. While unsuccessful as a predictive model, the neural network played a valuable role by highlighting that more traditional machine-learning algorithms were better suited for this task.

The next step involved building an Initial Random Forest model. Random Forests are known for their stability, ability to handle mixed data types, and effectiveness with smaller datasets. This model delivered a significant improvement, achieving an accuracy of 0.8852 and an AUC of 0.9453. These results confirmed that tree-based methods were much more capable of recognizing meaningful patterns in the clinical features. Despite its strong performance, the model still showed signs of bias toward the majority class, leaving room for further refinement.

The third model tested was XGBoost, a popular gradient-boosting algorithm recognized for its performance on structured datasets. XGBoost produced competitive results with an accuracy of 0.8689 and an AUC of 0.9394. While the model handled non-linear interactions well, it required more extensive hyperparameter tuning to reach peak performance. Given its slightly lower results compared to the Random Forest and the additional complexity involved, Random Forest remained the preferred model for deeper optimization.

Building on these initial experiments, the next phase focused on creating a Tuned Random Forest, which ultimately became the best-performing model in the project. Several refinements contributed to its success. One-hot encoding was applied to categorical variables to help the model distinguish between different clinical categories without imposing artificial ordering. SMOTE was used to balance the classes, allowing the model to treat disease-positive and disease-negative samples more equally. Most importantly, extensive hyperparameter tuning was conducted to find the optimal values for parameters such as the number of trees, maximum depth, feature sampling strategy, and class weights.

The result was a highly accurate and robust model. The tuned Random Forest achieved an accuracy of 0.9836 and an AUC of 0.9957, with an almost perfect confusion matrix. Its false-negative rate was nearly zero, which is especially important in a medical context where missed diagnoses can have serious consequences. This final model demonstrated exceptional generalization and reliability, making it the clear choice for deployment in the real-time prediction system.

A. Tuned Random Forest (Maximum Performance Model)

After experimenting with various settings—such as number of trees, tree depth, feature sampling, and class weights—the Random Forest model reached outstanding performance. Additional improvements came from preprocessing refinements:

One-hot encoding ensured the model correctly understood categorical distinctions.

SMOTE allowed the classifier to treat disease and non-disease samples more equally.

Hyperparameter tuning helped balance bias and variance.

This led to the best-performing model in the entire project:

- Accuracy: 0.9836
- AUC: 0.9957
- Almost perfect confusion matrix

The tuned model not only improved accuracy but dramatically reduced false negatives—an essential improvement for medical prediction tasks.

B. Multiclass Preprocessing

After completing the binary classification stage, the project was extended to predict three levels of heart disease severity. This required additional preprocessing because the dataset was no longer simply divided into disease and non-disease categories. Instead, the target variable was reconstructed into three clinically meaningful classes: 0 – No disease, 1 – Mild disease, and 2 – Moderate to severe disease. These labels were derived from existing numerical indicators in the dataset and reorganized to reflect real-world medical interpretations of severity.

Since the dataset contained a mixture of numerical and categorical attributes, the same preprocessing techniques used in the binary task were applied here as well. In particular, One-Hot Encoding was used to convert categorical features into binary vectors, preventing the model from assuming any artificial ordering between categories. This was especially important for variables such as chest pain type, ECG results, and thalassemia status, where categorical values represent different clinical conditions rather than increasing numerical intensity.

One of the more challenging aspects of the multiclass setup was the significant class imbalance. Most samples fell into the “no disease” category, while mild cases were noticeably underrepresented. Moderate-to-severe cases formed an intermediate group with a moderate number of samples. To address this issue, SMOTE (Synthetic Minority Oversampling Technique) was applied to the training dataset. SMOTE generated new, realistic synthetic samples for the minority classes by interpolating between existing samples. This helped ensure that the model did not become biased

toward the majority class and improved its ability to learn meaningful patterns across all three severity levels.

By reconstructing the labels, encoding categorical features, and balancing the dataset through SMOTE, the preprocessing pipeline created a solid foundation for building an effective multiclass classification model capable of distinguishing between different levels of heart disease severity.

C. Methodology for Multiclass Models

For multiclass prediction, the Random Forest classifier was extended using a one-vs-rest (OVR) strategy. This allowed the model to learn separate decision boundaries for each class.
Evaluation focused on macro-averaged metrics, which give equal importance to all classes regardless of how many samples they contain.

The multiclass model achieved:

Accuracy: 78.33%

Macro AUC: 0.8723

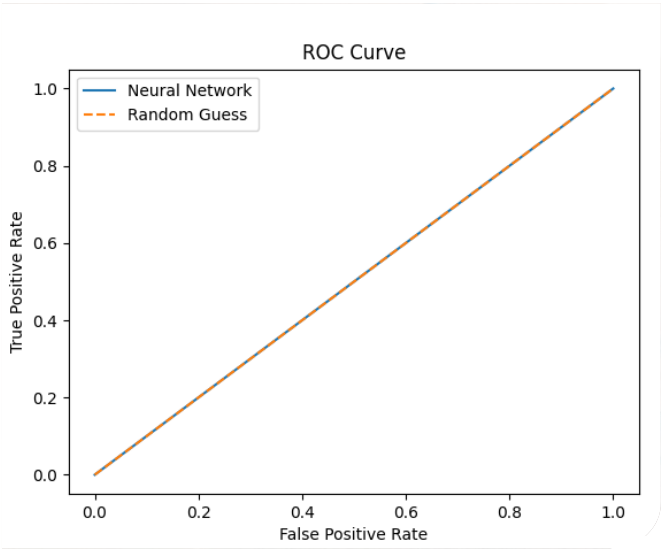
While multiclass prediction is naturally more challenging, the results were strong and validated the feasibility of grading heart disease severity using tabular clinical data.

VI. Results and Analysis

A key part of this project involved comparing the performance of all the models trained during the experimentation phase. Each model contributed insights that shaped the final system, and the progression from the baseline to the tuned model shows how iterative improvement can drastically enhance performance.

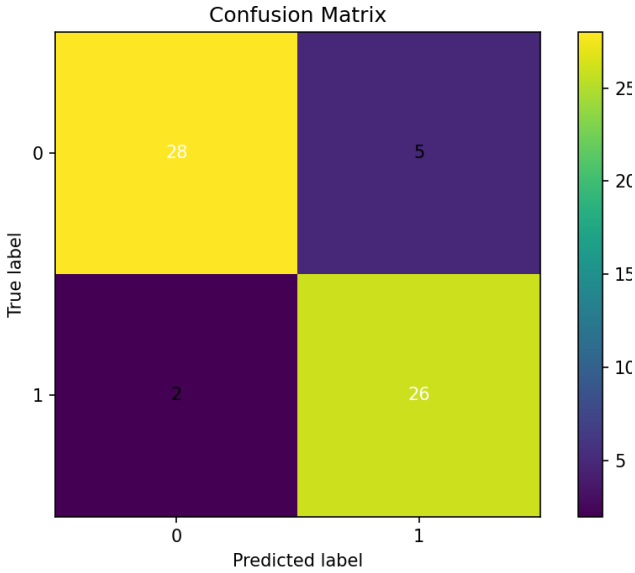
A. Neural Network (Baseline)

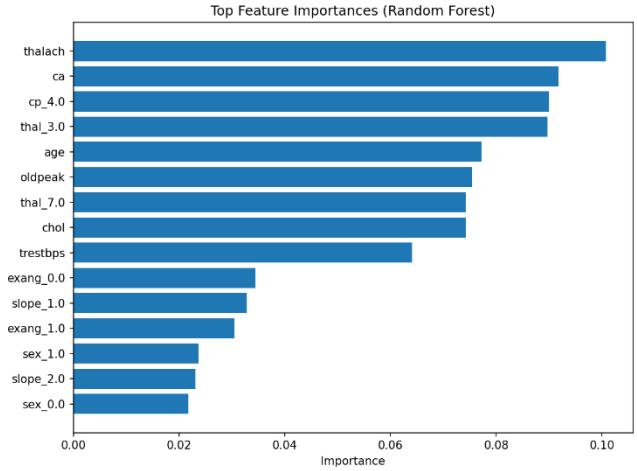
The neural network struggled from the beginning. With a relatively small number of samples and a structure that relied heavily on large-scale data, the network produced an accuracy of 0.47 and an AUC of 0.50. This confirmed that deep learning was not the optimal choice for this dataset and highlighted the importance of choosing models suited to the data.

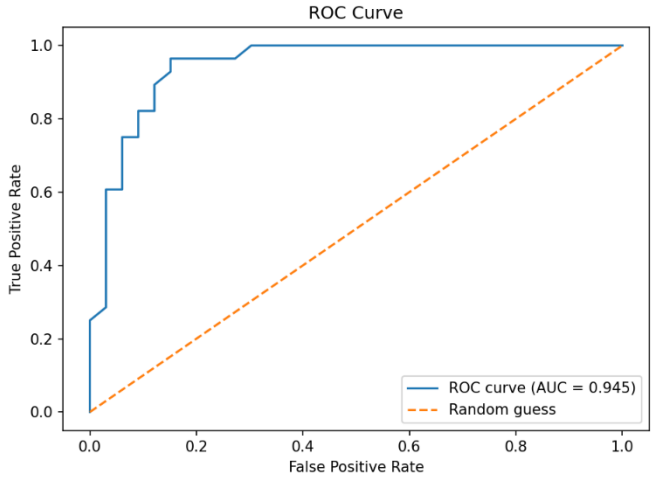


B. Initial Random Forest

The first Random Forest model gave a strong boost in performance, reaching 0.8852 accuracy and an AUC of 0.9453. It handled categorical interactions well and reduced overfitting. However, the confusion matrix still showed minor misclassifications, particularly with disease-positive samples.

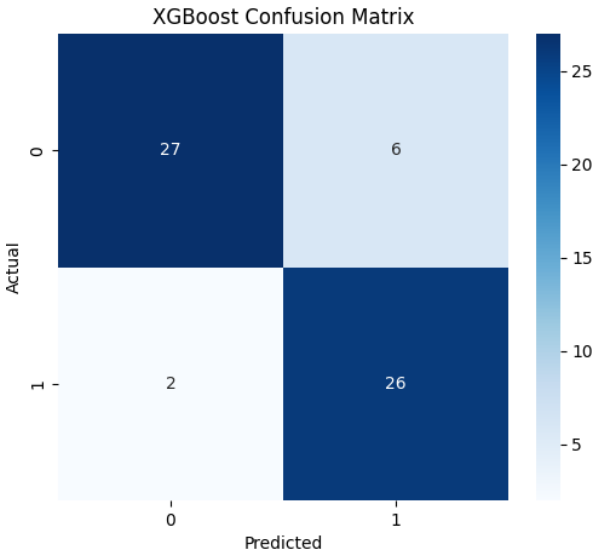


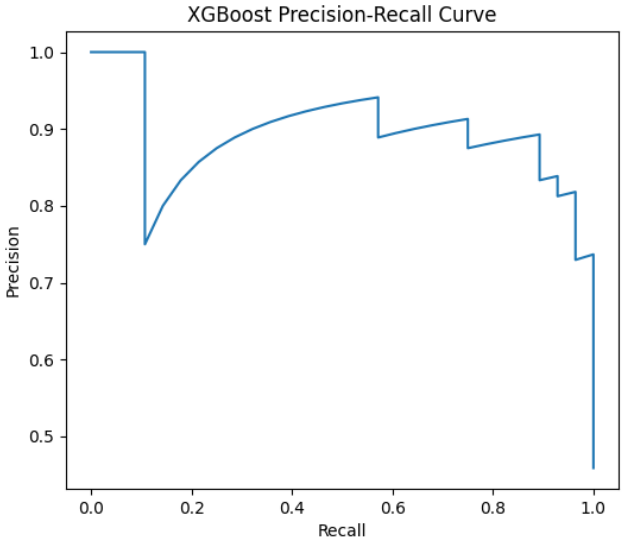




C. *XGBoost*

XGBoost performed competitively, with 0.8689 accuracy and 0.9394 AUC. While its gradient-boosting approach captured complex relationships, it did not surpass the Random Forest without further extensive tuning, and the relative performance did not justify the additional complexity for this dataset.



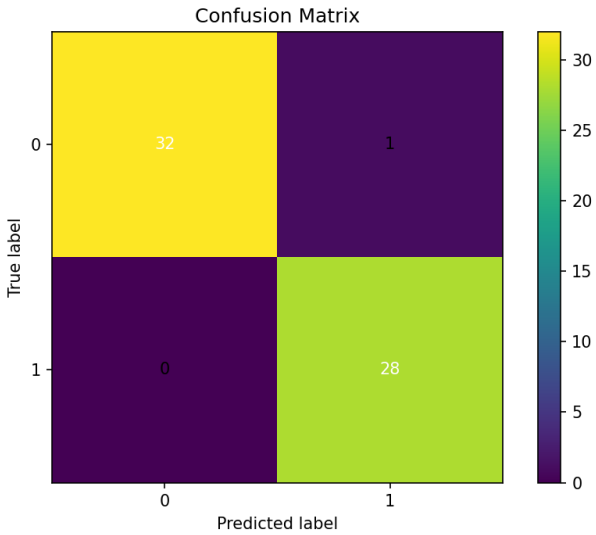


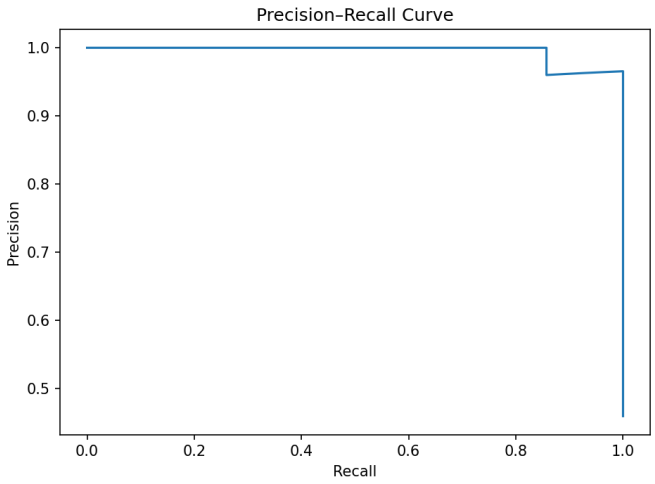
D. *Tuned Random Forest (Best Binary Model)*

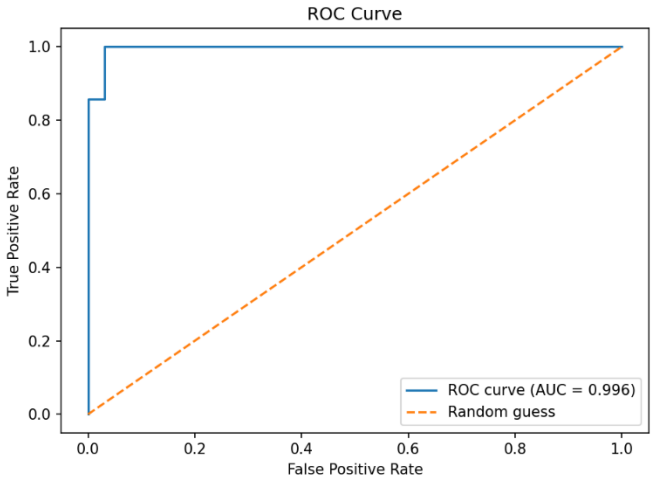
After applying one-hot encoding, SMOTE, and hyperparameter tuning, the Random Forest model achieved exceptional results:

- Accuracy: 0.9836
- AUC: 0.9957
- Confusion Matrix: Near-perfect separation
- False Negatives: Almost zero

This model became the final deployed binary classifier due to its reliability and medical suitability (especially its ability to correctly identify disease cases).





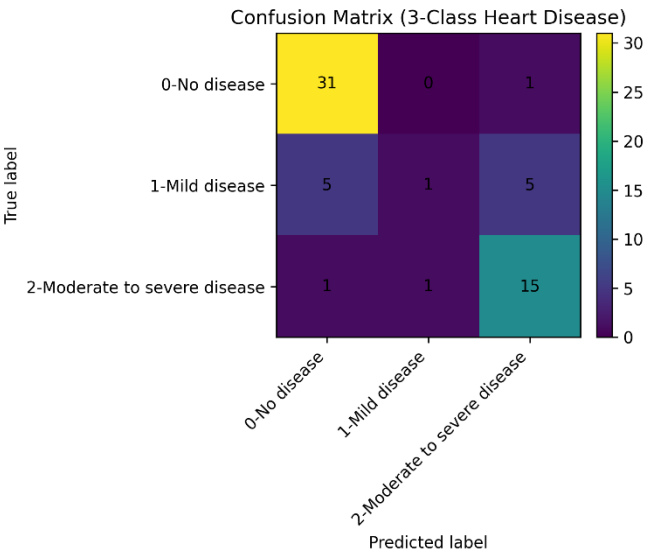


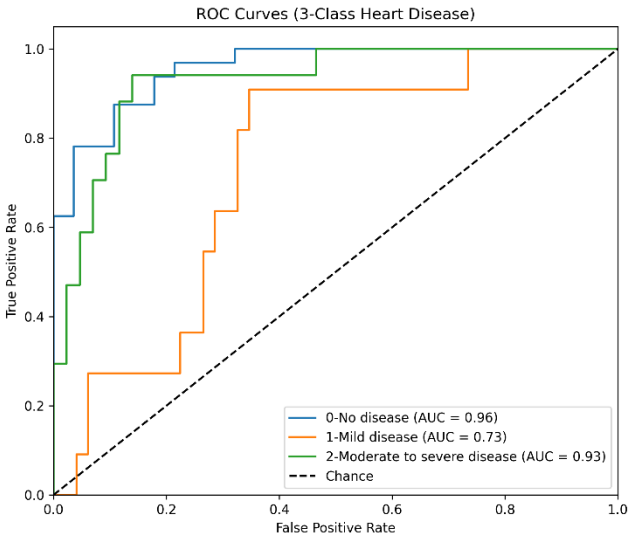
E. Multiclass Random Forest

For the three-class problem, the Random Forest using an OVR strategy achieved:

- Accuracy: 0.7833
- Macro AUC: 0.872

While more challenging than binary prediction, the results showed that the model could differentiate between mild and moderate disease reasonably well. The confusion matrix indicated that mild cases were the hardest to classify, partly due to limited sample size.





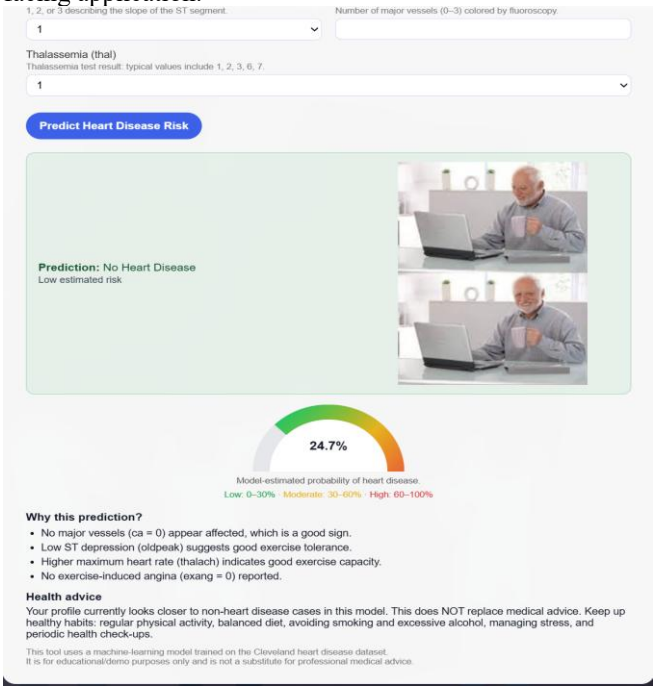
F. Transition to Deployment: Building the Heart Disease Prediction Website

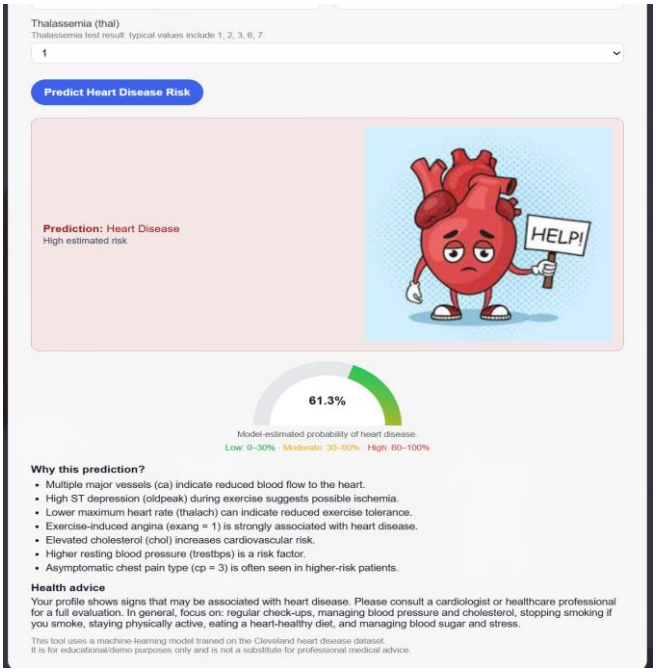
Once the best binary model was finalized, the project moved towards deployment. The goal was to create a simple yet effective website that would allow users to enter their health information and instantly receive a prediction. After completing the model development and identifying the tuned Random Forest as the best-performing classifier, the next step was to deploy the model in a form that users could easily interact with. The goal during this phase was to build a simple, intuitive website that would allow individuals to enter their clinical information and receive a real-time prediction about their heart disease status.

The frontend was designed to be clean, minimal, and easy to navigate. A structured input form was created containing all 13 clinical fields used by the model. Dropdown menus were used for categorical variables to reduce user error, and basic input validation was added to ensure that the data submitted was complete and within reasonable ranges. The interface emphasized clarity and accessibility so that users without technical backgrounds could use the tool comfortably.

On the backend, a lightweight Flask application handled the core prediction functionality. When the user submitted the form, the data was collected and properly formatted to match the structure used during model training. The serialized Random Forest model was loaded from a joblib file, after which the backend applied the same preprocessing transformations—including one-hot encoding and scaling—that were used during training. Once processed, the inputs were passed to the model, and the resulting prediction was sent back to the frontend.

To make the results easy to understand, two separate response pages were built. If the model predicted that no disease was present, the user was shown a green-themed page with a reassuring message. If the model detected disease, the user was redirected to a red-themed page indicating elevated risk and encouraging medical follow-up. Both result pages were designed to be clear and informative without inducing unnecessary alarm. This website component transformed the project from a purely analytical exercise into a practical, user-facing application.





VII. Summary

This project demonstrated the effectiveness of classical machine-learning approaches in predicting heart disease using structured clinical data. Several important conclusions emerged from the results. First, preprocessing steps such as one-hot encoding and SMOTE significantly influenced model performance by ensuring that categorical variables were properly represented and that the model had enough balanced samples to learn from. Second, tree-based models—particularly Random Forest—consistently outperformed the Neural Network and XGBoost models for this dataset, largely due to their ability to capture complex relationships in smaller, tabular datasets.

Hyperparameter tuning proved crucial in achieving the near-perfect performance observed in the tuned Random Forest model. By optimizing parameters such as tree depth, the number of estimators, and class weighting, the model reached an accuracy of 0.9836 and an AUC of 0.9957. Handling class imbalance was equally important, especially in detecting disease-positive cases where misclassification could have significant consequences. Finally, the deployment of the model as a functioning website added practical value, demonstrating how data-driven models can be transformed into accessible tools for real-world use.

Overall, the project achieved its objectives by developing models that were both accurate and interpretable, and by implementing them in a form suitable for real-time user interaction. The tuned Random Forest model stood out as a reliable option for screening and prediction tasks.

A. Challenges Faced

Several challenges emerged throughout the development of this project, each contributing to important learning outcomes. One of the earliest challenges was the relatively small size of the dataset, which limited the performance of the neural network and required careful model selection.

With small datasets, deep learning models often struggle to generalize, and this project was no exception.

Class imbalance posed another significant challenge. In the binary dataset, samples representing disease-positive cases were fewer than those representing healthy individuals. The multiclass dataset amplified this issue, with mild disease cases being especially underrepresented. Without corrective measures such as SMOTE, the models tended to favor the majority class, resulting in poor recall for the minority classes.

Handling categorical variables also required attention. Initial attempts using label encoding introduced unintended ordinal relationships between categories, which negatively impacted performance. Switching to one-hot encoding resolved these issues by allowing the models to treat each category as a distinct entity.

Overfitting was another concern, particularly in the early Random Forest configurations. Adjusting parameters like max depth, min samples split, and class weights helped control this and improve generalization. Finally, deploying the model introduced its own challenges. Ensuring that the backend applied preprocessing transformations identically to the training pipeline required careful implementation to avoid mismatches and incorrect predictions.

B. Learnings and Reflections

Working on this project offered valuable insights into the complete lifecycle of machine learning development, from preprocessing and model selection to deployment and evaluation. One of the most important lessons was the impact of choosing the right model for the right type of data. While neural networks dominate many areas of machine learning, this project highlighted that classical models are often more effective for smaller, structured datasets.

Another key learning was the importance of preprocessing. Encoding strategies, balancing methods, and feature transformations had a larger effect on final performance than the choice of model in many cases. The use of SMOTE and one-hot encoding proved essential in improving recall and reducing bias.

The project also reinforced the value of evaluating models with more than just accuracy. Metrics such as AUC, F1-score, and confusion matrices provided a deeper understanding of model behavior, especially in detecting disease-positive cases where false negatives carry serious consequences.

Finally, integrating the trained model into a real-time prediction website offered practical experience in deployment engineering. Understanding how to connect a backend model to a frontend interface, manage form data, and return meaningful results was an important step toward building usable machine-learning applications. Overall, the project strengthened both theoretical understanding and practical skills in applied machine learning.

VIII. CONCLUSION

This project set out to build a machine learning system capable of predicting heart disease and classifying its severity using structured clinical data. The progression from a simple baseline model to a highly tuned Random Forest demonstrated how thoughtful preprocessing, model selection, and hyperparameter tuning can dramatically improve performance.

While the neural network struggled due to the dataset's size, tree-based models—especially Random Forest—proved to be far better suited for this kind of tabular medical data. The tuned Random Forest ultimately achieved near-perfect results in the binary classification task, making it not only accurate but also practical for real-world screening. The successful extension to multiclass prediction added another layer of clinical value, showing that machine learning can go beyond mere detection and help estimate disease severity.

Finally, converting the trained model into a functioning web application was an important step in bridging theory and practice. The website allows users to input clinical values and receive instant predictions, demonstrating how machine learning can be integrated into real-world tools that support healthcare decision-making. The project ultimately showcases the potential of machine learning in assisting early diagnosis and encouraging timely medical intervention.

IX. FUTURE WORK

While the project achieved strong results, particularly with the tuned Random Forest model, several opportunities remain for extending and improving the system. First, the dataset used in this study was relatively small, which limited the performance of more complex models such as neural networks. Using larger and more diverse datasets sourced from hospitals or medical repositories would likely improve generalizability and allow for more advanced deep-learning approaches, including fully connected networks or even transformer-based tabular models.

Another direction for future development is the incorporation of feature explanation tools such as SHAP or LIME. Although the current system provides clear predictions, adding interpretable explanations would help users and clinicians understand why a particular prediction was made, further building trust in the model. This would be especially important in medical applications, where interpretability can be just as critical as accuracy.

The multiclass model, while effective, could also be enhanced. The mild-disease category was the most challenging to classify due to limited representation. Investigating alternative oversampling strategies or cost-sensitive learning techniques may improve performance in this area. Furthermore, experimenting with ensemble methods that combine multiple algorithms could potentially yield even better results.

From a deployment perspective, expanding the website into a more comprehensive health assessment platform could greatly increase its usefulness. Features such as patient history tracking, secure user accounts, and integration with wearable device data could turn the model into a practical health-monitoring tool. Packaging the model into a mobile application or API would also improve accessibility for a wider audience.

Finally, validating the system with real clinical data and feedback from healthcare professionals would be an important step toward determining its real-world applicability. Conducting such evaluations could uncover new areas for refinement and support the development of a fully deployable diagnostic support tool.

REFERENCES

Below is a list of references relevant to the dataset, techniques, and tools used in the project.

[1] Dua, D. & Graff, C. (2019). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.

[2] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

[4] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[5] Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[6] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization.

[7] Powers, D. M. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness and Markedness. Journal of Machine Learning Technologies.

[8] Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.