# Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification

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

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**T. Adi Lakshmi (22471A05E2)**

Under the esteemed guidance of

D. Venkata Reddy, M.Tech., (Ph.D.)

Assistant Professor


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tier -1

NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2025-2026

# NARASARAOPETA ENGINEERING COLLEGE

## (AUTONOMOUS)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


## CERTIFICATE

**This is to certify that the project that is entitled with the name "Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification" is a bonafide work done by T. Adi Lakshmi (22471A05E2) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2025-2026.**

PROJECT GUIDE
**D. Venkata Reddy**, M.Tech., (Ph.D).
**Assistant Professor**

PROJECT CO-ORDINATOR
**D. Venkata Reddy**, M.Tech., (Ph.D).
**Assistant Professor**

HEAD OF THE DEPARTMENT
**Dr. S. N. Tirumala Rao,** M.Tech., Ph.D.
**Professor & HOD**

EXTERNAL EXAMINER

# DECLARATION

I declare that this project work titled "**Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification**" is composed by me that the work contain here is my own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

**T. Adi Lakshmi (22471A05E2)**

# ACKNOWLEDGEMENT

I wish to express my thanks to various personalities who are responsible for the completion of my project. I am extremely thankful to my beloved chairman**, Sri M. V. Koteswara Rao, B.Sc.,** who took keen interest in me in every effort throughout this course. I owe my sincere gratitude to my beloved principal**, Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

I express my deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of the CSE department, and also to my guide**, Dr. Moturi Sireesha, B.Tech., M.Tech., Ph.D.,** Professor of the CSE department, whose valuable guidance and unstinting encouragement enabled me to accomplish my project successfully in time.

I extend my sincere thanks to **D. Venkat Reddy, M.Tech., (Ph.D.),** Assistant Professor & Project Coordinator of the project, for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for me throughout this project work.

I extend my sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during my B.Tech. degree.

I have no words to acknowledge the warm affection, constant inspiration, and encouragement that I received from my parents.

I affectionately acknowledge the encouragement received from my friends and those who were involved in giving valuable suggestions and clarifying my doubts, which really helped me in successfully completing my project.

**By**

**T. Adi Lakshmi (22471A05E2)**

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# INSTITUTE VISION AND MISSION

## INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

## INSTITUTION MISSION

**M1:** Provide the best class infra-structure to explore the field of engineering and research

**M2:** Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

**M3:** Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

## MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

**M1:** Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

**M2:** Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

**M3:** Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# Program Specific Outcomes (PSO's)

**PSO1:** Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

**PSO2:** Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

**PSO3:** Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# Program Educational Objectives (PEO's)

The graduates of the programme are able to:

**PEO1:** Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

**PEO2:** Use various software tools and technologies to solve problems related to the academia, industry and society.

**PEO3:** Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

**PEO4:** Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# Program Outcomes

**PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2: Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations**.**

**PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice**.**

**PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Project Course Outcomes (CO'S):

**CO421.1:** Analyze the System of Examinations and identify the problem.
**CO421.2:** Identify and classify the requirements.
**CO421.3:** Review the Related Literature
**CO421.4:** Design and Modularize the project
**CO421.5:** Construct, Integrate, Test and Implement the Project.
**CO421.6:** Prepare the project Documentation and present the Report using appropriate method.

## Course Outcomes – Program Outcomes mapping

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | | ✓ | | | | | | | | | | | ✓ | | |
| **C421.2** | ✓ | | ✓ | | ✓ | | | | | | | | ✓ | | |
| **C421.3** | | | | ✓ | | ✓ | ✓ | ✓ | | | | | ✓ | | |
| **C421.4** | | | ✓ | | | ✓ | ✓ | ✓ | | | | | ✓ | ✓ | |
| **C421.5** | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **C421.6** | | | | | | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | |

## Course Outcomes – Program Outcome correlation

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | 2 | 3 | | | | | | | | | | | 2 | | |
| **C421.2** | | | 2 | | 3 | | | | | | | | 2 | | |
| **C421.3** | | | | 2 | | 2 | 3 | 3 | | | | | 2 | | |
| **C421.4** | | | 2 | | | 1 | 1 | 2 | | | | | 3 | 2 | |
| **C421.5** | | | | | 3 | 3 | 3 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 1 |
| **C421.6** | | | | | | | | | 3 | 2 | 1 | | 2 | 3 | |

**Note: The values in the above table represent the level of correlation between CO's and PO's:**

1. Low level
2. Medium level
3. High level

## Project mapping with various courses of Curriculum with Attained PO's:

| Name of the course from which principles are applied in this project | Description of the device | Attained PO |
|---|---|---|
| C2204.2, C22L3.2 | Collected and analyzed the Sulianova CVD dataset and planned the AI-based model architecture for cardiovascular risk prediction. | PO1, PO3 |
| CC421.1, C2204.3, C22L3.2 | Performed requirement analysis and implemented preprocessing steps like label encoding and normalization. | PO2, PO3 |
| CC421.2, C2204.2, C22L3.3 | Designed UML diagrams and modularized system workflow between frontend, backend, and models. | PO3, PO5, PO9 |
| CC421.3, C2204.3, C22L3.2 | Integrated and tested multiple deep learning models with SHAP explainability and evaluation metrics. | PO1, PO5 |
| CC421.4, C2204.4, C22L3.2 | Prepared project documentation and reports collaboratively by all team members. | PO10 |
| CC421.5, C2204.2, C22L3.3 | Presented project progress and integrated results through team-based reviews. | PO10, PO11 |
| C2202.2, C2203.3, C1206.3, C3204.3, C4110.2 | Deployed the deep learning models using Flask and React for real-time prediction. | PO4, PO7 |
| C32SC4.3 | Developed a web interface for user input and real-time cardiovascular risk assessment. | PO5, PO6 |

# ABSTRACT

cardiovascular diseases (CVD) are a significant health issue around the world, which gives the reason why an accurate and interpretable predictive system should be developed. This is a study that proposes a deep learning method that considers a compound resampling technique to consider data skew and enhance the risk prediction of cardiovascular disease. The framework contains five neural network models: TabNet, Attention-LSTM, a mix of CNN and BiLSTM, multilayer perceptron (MLP), and 1D CNN. SHAP explainability was applied to the TabNet, The MLP, and the CNN+BiLSTM models to increase the explained Ness of the model, and thus it revealed many new things regarding feature prominence. TabNet has been the model to perform the best out of all models that have been tested with an accuracy of 96.19%, but it provided good results on F1 and AUC as well. Albeit the results of the Attention-LSTM and 1D CNN performing slightly worse, the MLP and CNN+BiLSTM models also gave comparative results. Each of the models was trained on a common preprocessing pipeline. The findings indicate that aggregating multiple models of deep learning with focused explainability and balanced data strategies are capable of generating predictive tools of cardiovascular risks that are reliable and interpretable. This approach has a possibility to become a part of clinical support systems.

# INDEX

# LIST OF FIGURES

# LIST OF FIGURES

# 1. INTRODUCTION

Cardiovascular diseases remain the leading cause of death worldwide, creating an urgent need for early, reliable, and interpretable risk stratification tools that can generalize across diverse populations and clinical settings. To meet this need, we develop a deep learning framework that emphasizes both predictive performance and transparency so that model outputs can be meaningfully integrated into clinical workflows. Our central contribution is a unified pipeline that couples hybrid resampling for class balance with model-agnostic explainability, enabling robust detection of high-risk individuals while preserving clinicians' ability to understand why a prediction was made.

The proposed approach—Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification—operates end-to-end: from data preparation and balancing to training, evaluation, and explanation. It is designed for tabular healthcare data and intentionally avoids hand-engineered features. Instead, representation learning is delegated to modern deep architectures, and interpretability is achieved post-hoc using SHAP to attribute predictions to clinically meaningful variables. This design allows the system to capture nonlinear dependencies among risk factors without sacrificing traceability of the final decision.

We ground the study on the publicly available Sulianova (Kaggle) cardiovascular dataset comprising 70,000 anonymized patient records with mixed numerical, ordinal, and binary attributes relevant to cardiovascular risk. The target indicates physician-diagnosed CVD, while inputs span demographics, anthropometrics, blood pressure, lipids, glucose, and lifestyle indicators. This breadth permits the models to learn complementary patterns spanning physiology and behaviour, supporting comprehensive risk stratification from routine clinical measures.

A consistent preprocessing routine is applied across all models. Categorical fields such as sex, smoking, alcohol, and physical activity are label-encoded; continuous variables including age, height, weight, blood pressure, cholesterol, and glucose are standardized using z-scores to stabilize optimization and prevent scale-driven dominance. We employ a stratified 80/20 train–test split to preserve outcome prevalence between partitions, ensuring fair evaluation that reflects the original

population distribution.

Because outcome imbalance can bias classifiers toward the majority class, we incorporate a hybrid resampling strategy that combines random undersampling of the majority class with SMOTE-based synthetic oversampling of the minority class. This dual remedy both reduces redundancy and injects plausible minority examples, improving minority sensitivity while maintaining overall generalization—an essential property when the clinical cost of missed positives is high.

Five complementary deep learning architectures are instantiated within the same pipeline: TabNet for attentive feature selection in tabular data; a CNN+BiLSTM hybrid to learn local patterns and bidirectional dependencies across features; a lightweight multilayer perceptron (MLP) that provides a strong feed-forward baseline; an attention-augmented LSTM to dynamically weight salient feature interactions; and a 1D-CNN that emphasizes hierarchical pattern extraction with global pooling for efficiency. This diversity allows us to assess how different inductive biases—attention, recurrence, and convolution—affect tabular CVD prediction.

Training is implemented in PyTorch with a harmonized recipe: Adam optimizer (learning rate 0.001), binary cross-entropy loss, batch size 128, and early stopping within a 100-epoch budget to prevent overfitting. Dropout regularization is used in MLP, CNN+BiLSTM, and Attention-LSTM variants to further improve generalization. By standardizing hyperparameters and data processing, differences in performance can be attributed to architectural properties rather than pipeline artifacts.

We evaluate models using accuracy, F1-score, and ROC-AUC to capture overall correctness, class-balance sensitivity, and discrimination, respectively. Across the five deep networks, TabNet delivers the top performance, achieving 96.19% accuracy and 0.9916 AUC on the test set, with MLP and CNN+BiLSTM close behind and 1D-CNN trading some accuracy for speed. Confusion-matrix analysis confirms low false-positive and false-negative rates, and 5-fold cross-validation further stabilizes estimates across folds.

Explainability is integrated via SHAP for TabNet, MLP, and CNN+BiLSTM to expose both global and local drivers of predictions. The most influential features consistently include age, systolic blood pressure (ap_hi), cholesterol, and glucose—aligning with established cardiovascular risk factors and reinforcing clinical face

validity. These SHAP attributions help clinicians audit model reasoning, explore patient-specific risk signals, and build trust in automated recommendations.

Beyond accuracy, we profile computational efficiency—training time, parameter count, and GPU memory—to understand deployment trade-offs. While TabNet is the most accurate, MLP offers faster training with fewer parameters, and 1D-CNN provides the most efficient runtime at a modest accuracy cost, informing choices for settings with constrained resources or latency requirements.

To assess external generalization, we apply the best model (TabNet) to the classic UCI Heart Disease dataset after identical preprocessing and balancing. Performance remains strong at 93.4%, indicating resilience to domain shift and supporting the pipeline's portability across cohorts with different distributions. This external check complements cross-validation and significance testing, strengthening evidence for real-world translation.

Taken together, the results show that a carefully engineered deep learning pipeline—with hybrid resampling for balance and SHAP for transparency—can deliver high performance while remaining interpretable and computationally tractable. By unifying preprocessing, training, evaluation, and explanation across five architectures, the study provides a reproducible blueprint for deploying deep models in cardiovascular risk stratification and sets the stage for future integration into clinical decision support.

## 1.1 Motivation

Cardiovascular diseases remain a leading cause of global mortality, with millions of deaths occurring annually due to delayed diagnosis and intervention. Traditional risk assessment tools often fail to capture the complex, nonlinear relationships among diverse clinical and lifestyle factors. In many cases, diagnosis is based on limited variables, reducing prediction accuracy. There is a need for intelligent systems that can leverage large-scale patient data to deliver early, precise, and personalized risk assessment.

Deep learning offers a powerful way to model such complexity without manual feature engineering, making it suitable for healthcare applications. Architectures like TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D-CNN can process tabular medical datasets effectively, learning both spatial and sequential relationships. However, despite high accuracy, adoption in clinical settings demands transparency. Models must explain their predictions in ways that align with medical reasoning to earn clinician trust.

A major challenge in cardiovascular risk modelling is class imbalance, where healthy cases far outnumber positive diagnoses. This imbalance can cause models to favour the majority class, missing high-risk patients. Our approach uses hybrid resampling—combining SMOTE oversampling of minority cases with undersampling of majority cases—to achieve balanced datasets. This strategy improves sensitivity to high-risk individuals while preserving data integrity.

Finally, model interpretability is addressed through SHAP, a model-agnostic explanation framework that reveals the most influential features in predictions. Applying SHAP to multiple architectures ensures both local and global insights, aligning model behaviour with known medical risk factors. By integrating hybrid resampling, diverse deep learning architectures, and explainability, this framework delivers a balanced, accurate, and interpretable solution for cardiovascular risk stratification.

## 1.2 Problem Statement

Cardiovascular diseases are among the deadliest health conditions worldwide, claiming millions of lives each year. Early detection is critical for prevention and effective treatment, yet existing diagnostic tools and statistical models often fall short in capturing the complex interplay of multiple clinical and lifestyle factors. These limitations result in delayed intervention, higher mortality rates, and increased healthcare costs.

The availability of large-scale cardiovascular datasets offers opportunities to develop predictive systems capable of accurate and timely risk assessment. However, such datasets often suffer from severe class imbalance, where the number of healthy cases far exceeds confirmed disease cases. This imbalance can bias predictive models toward the majority class, undermining their ability to detect high-risk patients who require immediate attention.

While deep learning models have demonstrated superior predictive capabilities in healthcare, their lack of interpretability remains a significant barrier to adoption in clinical practice. Without clear explanations for predictions, even highly accurate models may fail to gain the trust of healthcare professionals, limiting their practical utility in real-world decision-making scenarios.

There is, therefore, a critical need for a unified framework that can address data imbalance, deliver high predictive accuracy, and provide interpretable outputs. Such a system should integrate hybrid resampling strategies with advanced deep learning architectures, along with model-agnostic explainability tools, to create a reliable and transparent solution for cardiovascular risk stratification.

Furthermore, the solution must be computationally efficient and adaptable for deployment in real-world healthcare environments. It should perform consistently across diverse patient populations and be validated through rigorous cross-validation and external datasets. This ensures not only strong predictive performance but also scalability, robustness, and long-term usability in clinical decision-support systems.

## 1.3 Objective

The primary objective of this study is to design and develop a deep learning–based framework for accurate cardiovascular risk stratification. The framework must handle complex, nonlinear feature interactions without manual feature engineering, leveraging the power of modern neural network architectures tailored for tabular medical data.

A key goal is to address the inherent class imbalance in cardiovascular datasets through a hybrid resampling strategy. This involves combining SMOTE-based oversampling of minority cases with controlled undersampling of majority cases to improve sensitivity toward high-risk patients while maintaining overall model generalization.

Another objective is to evaluate multiple deep learning architectures—TabNet, CNN+BiLSTM, MLP, Attention-LSTM, and 1D-CNN—within a unified preprocessing and training pipeline. This comparison allows identification of the most effective architecture for the given dataset while ensuring fairness in evaluation through standardized data preparation and hyperparameters.

The framework also aims to integrate SHAP explainability for selected architectures, enabling both local and global interpretation of model outputs. This will ensure that the system's predictions are transparent, clinically aligned, and trustworthy for real-world medical decision-making.

Finally, the study seeks to assess computational efficiency, scalability, and cross-domain generalization through cross-validation, statistical testing, and external dataset validation. The objective is to produce a solution that is not only accurate and interpretable but also deployable in real-world healthcare environments to assist clinicians in proactive patient management.

# 2. LITERATURE SURVEY

El-Sofany et al. [1] are the authors of one of the machine learning frameworks that were suggested in the field of heart disease prediction indicating the usage of classifiers like XGBoost, AdaBoost, or SVM. They used ANOVA, chi-square, mutual information in pre-processing the features and scored 97.57 accuracy and AUC 0.98 with XGBoost on their own dataset. SHAP was applied to enhance model interpretability; nevertheless, reproducibility cannot be achieved because of using the data as private.

Cenitta et al. [2] presented a model called Hybrid Residual Attention-Enhanced LSTM (HRAE-LSTM) that could be used to predict ischemic heart disease. They used LSTM with attention models, which yielded 97.7 Accuracy of the UCI dataset. The model bore an effective outcome but was only tested on ischemic data, and their findings are not validated on the wider types of CVD.

Sianga et al. [3] included temperature and humidity to their predictive models. By means of imbalance using SMOTE and filtering noise using the IQR, they managed to enhance performance using classifiers such as XGBoost and SVM with an accuracy of up to 95.24%. Their research was however not generalizable to regions and did not execute deep learning and temporal modelling.

Pal et al. [4] developed an interactive system of CVD which used InceptionNet, Random Forest, KNN, and Logistic Regression. Among the 14 features, InceptionNet gave the best result with 98.89 percent accuracy. The study also had a limited feature set and therefore unable to draw a generalizable conclusion based on a high level of performance. Robustness can be increased by the use of a wider clinical phenotype.

The UCI dataset was investigated by Gupta et al. [5] with regards to the traditional ML models, including KNN, SVM, and Logistic Regression. KNN recorded an accuracy of 90.16% which was the highest. Although good, the study did not involve ensemble methods and deep learning, which were very innovative and profound.

Selvitopi et al. [6] evaluated Naive Bayes, LR, KNN, and Random Forest using a hybrid dataset from multiple sources. Naive Bayes performed best with 85.63% accuracy and 84.29% AUC. The study validated traditional models but did not

include deep learning or explainable AI techniques.

Daharwal et al. [7] compared Random Forest and KNN using multiple datasets and highlighted the importance of preprocessing and feature selection. Random Forest achieved 93.33% accuracy, yet the study lacked interpretability methods like SHAP or LIME and did not include DL models.

Sreekumari et al. [8] focused on feature selection and ensemble evaluation using a large Kaggle dataset. They applied chi-square, correlation, and brute-force methods and used voting ensembles (KNN, DT, NB), achieving a maximum accuracy of 78.42%. Deep learning or attention-based models were not explored.

Trigka and Dritsas [9] investigated the role of data augmentation and class imbalance using enhanced correlation-aware SMOTE with deep models including CNN, RNN, LSTM, MLP, and Autoencoders. Though interpretability methods were not used, CNN with enhanced SMOTE achieved the highest AUC of 0.90.

Efe and Demir [10] evaluated multiple feature selection techniques—including Relief and stability selection—on tree-based models. The highest accuracy of 84% was obtained with stability selection. The study focused on traditional ML methods and did not investigate deep learning or hybrid models.

# 3. SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

Cardiovascular disease prediction has long relied on conventional statistical models such as the Framingham Risk Score, logistic regression, and Cox proportional hazards models. These methods typically use a small set of predefined clinical factors—such as age, blood pressure, cholesterol, smoking status, and diabetes history—to estimate an individual's probability of developing cardiovascular disease within a certain timeframe. While these systems are easy to implement and interpret, they make strong assumptions about linearity and independence between variables. Such assumptions limit their ability to model the complex, nonlinear interactions that are common in real-world health data, reducing predictive accuracy in diverse populations.

In more recent research, traditional machine learning algorithms have been introduced to improve prediction capabilities. Methods such as decision trees, random forests, k-nearest neighbours (KNN), gradient boosting machines, and support vector machines (SVMs) have been applied to cardiovascular datasets. These models can handle larger feature sets and capture some nonlinear patterns, but they often require extensive manual feature selection and parameter tuning to achieve competitive performance. Moreover, their predictive gains are sometimes marginal when applied to highly imbalanced medical datasets, where the minority class (disease cases) is underrepresented.

Several existing systems have experimented with early deep learning models, particularly multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), trained on tabular health data. While these architectures can automatically learn feature representations without manual engineering, most studies in this space have relied on naive resampling techniques, such as random oversampling or simple SMOTE, to address class imbalance. Oversampling without careful control can lead to overfitting by replicating noise, while undersampling can discard important patterns from the majority class. As a result, the performance improvements achieved by these models are often inconsistent and dataset-dependent.

Another common limitation in existing systems is the lack of a unified, reproducible pipeline for preprocessing, balancing, training, and evaluation. Many

prior works focus on a single architecture and apply ad hoc preprocessing steps that are not standardized across experiments. This makes it difficult to fairly compare performance across different methods or reproduce results in new settings. Without a consistent pipeline, the conclusions drawn from such studies may be biased toward specific data splits, hyperparameters, or preprocessing choices.

Explainability is another major gap in existing cardiovascular prediction systems. Many deep learning models are considered "black boxes," offering high predictive accuracy but little insight into how decisions are made. While some studies report feature importance scores from tree-based models or attention weights from neural networks, these are not always clinically interpretable. This lack of transparency can reduce trust among healthcare providers, who must be able to justify treatment recommendations to patients and adhere to ethical and regulatory standards.

Moreover, many existing approaches have limited validation on diverse datasets, which raises concerns about generalizability. Models trained and tested on a single dataset may overfit to the specific patterns and feature distributions of that data, performing poorly when applied to new populations. This issue is compounded in healthcare, where demographic, lifestyle, and clinical differences across regions can significantly impact disease patterns. Without robust validation—such as k-fold cross-validation and external dataset testing—existing systems risk delivering over-optimistic performance metrics that do not translate into real-world settings.

In summary, the current landscape of cardiovascular risk prediction systems shows a mix of simple statistical methods, traditional machine learning models, and early deep learning approaches. While each has its strengths, none fully addresses the combined challenges of nonlinear feature interactions, data imbalance, reproducibility, generalizability, and clinical interpretability. These limitations highlight the need for a more advanced, integrated framework that unites robust data preprocessing, hybrid resampling, diverse deep learning architectures, and transparent explainability—paving the way for reliable and actionable cardiovascular risk stratification.

### 3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM FOR CARDIOVASCULAR RISK STRATIFICATION

**Despite progress in automated cardiovascular disease (CVD) risk prediction, existing systems have several limitations:**

- **Dependence on Linear Assumptions:**

Many traditional models such as logistic regression and Cox proportional hazards rely on linear relationships among risk factors. These assumptions limit the ability to capture complex, nonlinear interactions present in real patient data, reducing predictive accuracy in diverse populations.

- **Manual Feature Engineering Requirements:**

Traditional machine learning approaches often require manual selection and transformation of features, which is time-consuming and may overlook subtle but important patterns hidden in the dataset.

- **Inadequate Handling of Class Imbalance:**

In most CVD datasets, healthy cases significantly outnumber disease cases. Many existing systems use either naive oversampling or undersampling, leading to overfitting or loss of valuable information, ultimately harming generalization to unseen data.

- **Lack of Unified and Reproducible Pipelines:**

Many prior works focus on a single algorithm with inconsistent preprocessing steps. This lack of standardization makes it difficult to fairly compare methods, replicate results, or deploy the systems in real-world healthcare settings.

- **Limited Model Interpretability:**

Deep learning–based systems often function as “black boxes,” offering high accuracy but no clear explanation for predictions. Without transparent feature importance or reasoning, clinicians may hesitate to rely on such models for critical decision-making.

- **Restricted Validation and Generalizability:**

Existing approaches are often trained and evaluated on a single dataset without external testing. This raises concerns about overfitting to dataset-specific characteristics and poor adaptability to other populations or healthcare environments.

- **High Computational Demand in Some Architectures:**

Certain deep architectures used in earlier systems can require significant computational resources, making them impractical for deployment in low-resource medical facilities or real-time screening scenarios.

## 3.2 PROPOSED SYSTEM

The proposed model employs a hybrid framework integrating advanced deep learning architectures with a hybrid resampling strategy to enhance the accuracy, robustness, and interpretability of cardiovascular disease (CVD) risk prediction. This integration leverages the strengths of multiple deep learning models—TabNet, CNN+BiLSTM, MLP, Attention-LSTM, and 1D-CNN—while addressing class imbalance using a combination of Synthetic Minority Oversampling Technique (SMOTE) and undersampling. This unified approach ensures balanced data distribution, high predictive performance, and clinically relevant explanations through SHAP interpretability.

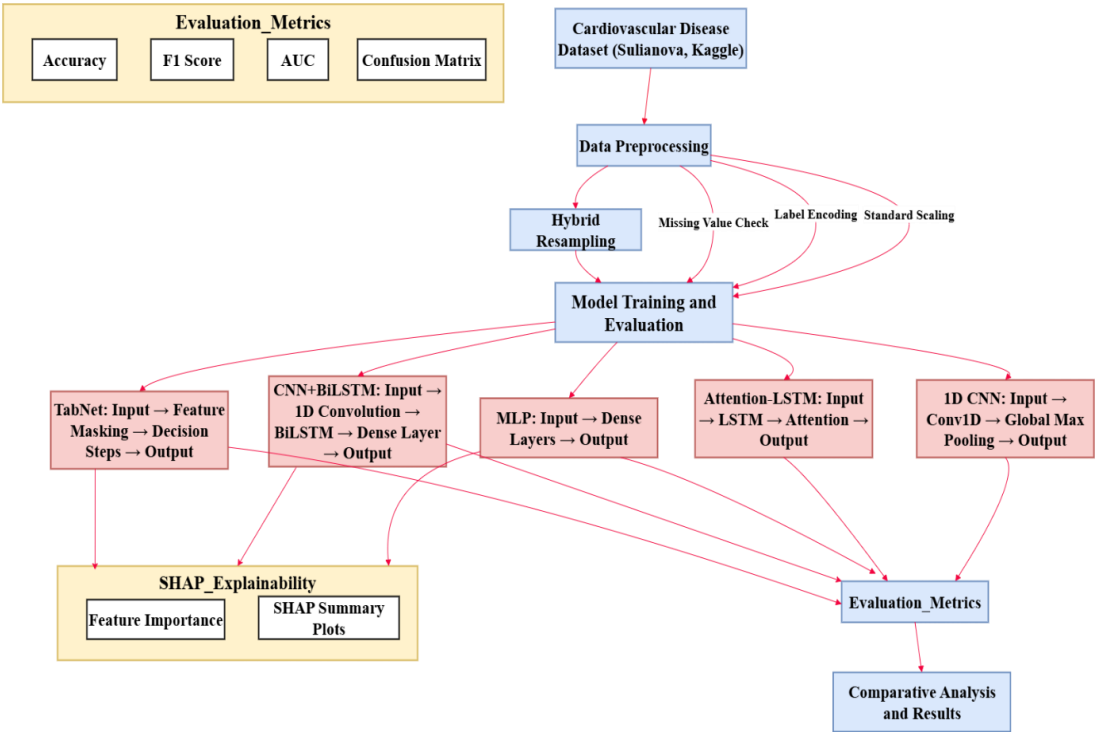


**Fig 3.1. Flow chart of proposed system**

The workflow, illustrated in Fig. 3.2, begins with preprocessing the Sulianova cardiovascular dataset to ensure data quality and consistency. This step includes handling missing values, label encoding categorical variables, and applying normalization to scale features. By standardizing the input data, all models in the

framework can be trained under identical conditions for fair comparison.

After preprocessing, the dataset is subjected to a hybrid resampling process, where SMOTE is applied to generate synthetic samples of the minority (CVD-positive) class, followed by undersampling of the majority (healthy) class. This strategy mitigates bias toward the majority class while preserving the statistical integrity of the dataset. The balanced data is then split into training and testing sets using stratified k-fold cross-validation to ensure consistent representation of both classes across folds.

The processed data is fed into five different deep learning architectures within the same pipeline.

- **TabNet** is used for its ability to learn sparse feature representations directly from tabular data.
- **CNN+BiLSTM** captures both local feature interactions and temporal/sequential dependencies among features.
- **MLP** provides a lightweight yet effective baseline for tabular prediction tasks.
- **Attention-LSTM** enhances sequential learning by focusing on the most informative features via attention mechanisms.
- **1D-CNN** extracts local feature patterns with minimal computational complexity.

Each model is trained on the balanced dataset, and its performance is evaluated using metrics such as Accuracy, F1-Score, and AUC. SHAP (Shapley Additive Explanations) is applied post-training to both local and global feature interpretations, identifying which clinical and lifestyle factors most influence the prediction. This interpretability step ensures that the system's outputs align with known medical knowledge, thereby increasing clinical trust.

Finally, results from all models are compared, and the best-performing model—or an ensemble of multiple models—is selected for deployment. This proposed system addresses the limitations of existing methods by combining robust imbalance handling, diverse deep learning architectures, and explainable AI techniques into a single, reproducible framework for cardiovascular risk stratification.

## Advantages over existing system:

1. **Improved Accuracy and Generalization**

   By integrating multiple deep learning architectures (TabNet, CNN+BiLSTM, MLP, Attention-LSTM, 1D-CNN) within a unified pipeline, the system achieves higher predictive performance across diverse patient profiles compared to single-model approaches.

2. **Robust Class Imbalance Handling**

   The hybrid resampling strategy—combining SMOTE oversampling with controlled undersampling—ensures balanced training data, improving the detection of high-risk CVD patients and reducing bias toward the majority class.

3. **Standardized Preprocessing Pipeline**

   Uniform preprocessing steps, including missing value handling, label encoding, and feature scaling, are applied across all models, enabling fair performance comparisons and reproducible results.

4. **Model Interpretability through SHAP**

   SHAP explainability provides both local and global feature importance, allowing clinicians to understand the reasoning behind predictions and increasing trust in the system's outputs.

5. **Comprehensive Evaluation**

   Use of stratified k-fold cross-validation ensures stable and reliable performance metrics, reducing the risk of overfitting to specific train-test splits.

6. **Scalability and Flexibility**

   The framework can easily accommodate additional deep learning architectures or be adapted to other medical prediction tasks with minimal modifications.

7. **Deployment-Ready Design**

   Optimized model selection and resource-efficient architectures like 1D-CNN enable deployment in real-time screening tools, even in low-resource healthcare environments.

## 3.3 FEASIBILITY STUDY

The proposed Hybrid Resampling-Driven Deep Learning Architecture integrates advanced neural models (TabNet, CNN+BiLSTM, MLP, Attention-LSTM, and 1D-CNN) with a hybrid resampling strategy to improve cardiovascular disease (CVD) risk stratification. Below is the detailed feasibility analysis:

**1. Technical Feasibility**

- **Automated Feature Learning:**

Unlike traditional methods that rely on manual feature engineering, deep learning models automatically capture complex, nonlinear relationships in clinical and lifestyle data. This ensures richer representation and higher predictive accuracy.

- **Hybrid Resampling for Balance:**

The combination of SMOTE oversampling and undersampling mitigates class imbalance, improving sensitivity to high-risk patients while preserving the integrity of majority class data.

- **Enhanced Model Interpretability:**

Integration of SHAP explainability provides local and global insights into feature importance, making predictions transparent and clinically relevant.

- **Scalability Across Architectures:**

The framework is flexible, supporting multiple architectures in a unified pipeline. Additional models or improved versions can be integrated with minimal changes.

- **Cross-Validation and Robustness:**

The use of stratified k-fold validation ensures consistent performance, reducing the risk of overfitting to a single dataset and improving reliability for real-world deployment.

**2. Operational Feasibility**

- **Ease of Deployment:**

The models can be deployed as part of decision-support systems in hospitals and clinics. Lightweight models like 1D-CNN can run on standard systems, while TabNet and CNN+BiLSTM can leverage GPUs for efficiency.

- **Interpretability for Clinicians:**

By providing SHAP-based explanations, the framework builds trust among doctors,

as they can clearly see which risk factors influenced predictions for each patient.

- **Data Handling Flexibility:**

The system is capable of handling heterogeneous patient data, including categorical, numerical, and clinical features, with standardized preprocessing pipelines.

- **Model Maintenance:**

As new patient data becomes available, the models can be retrained or fine-tuned, keeping the system up to date without redesigning the entire pipeline.

**3. Economic Feasibility**

- **Cost-Effective Training:**

The hybrid framework uses efficient architectures like 1D-CNN and MLP for quick deployment, while heavier models can be trained using available GPU resources. This balance reduces training and infrastructure costs.

- **Resource Optimization:**

Hybrid resampling avoids unnecessary duplication of data or extreme undersampling, making better use of available datasets and computational resources.

- **Reduced Healthcare Burden:**

Automated risk prediction reduces manual effort in screening patients, enabling early diagnosis and lowering long-term treatment costs for cardiovascular diseases.

- **Sustainable Investment:**

While initial development and deployment may require moderate costs, the adaptability of the framework ensures long-term usability, scalability, and improved patient outcomes, making it a cost-effective healthcare solution.

## 3.4 USING COCOMO MODEL

The COCOMO (Constructive Cost Model) is a widely used estimation technique in software engineering that helps predict the effort, development time, and team size required for a project. Applying the Basic COCOMO Model to the proposed Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification provides a structured approach to estimating project requirements.

Given the complexity and scope of the project—developing a deep learning–based application with multiple models (TabNet, CNN+BiLSTM, MLP, Attention-LSTM, 1D-CNN), SHAP explainability, hybrid resampling techniques, and a web-

based frontend for deployment—the project falls under the Semi-Detached category. This category is characterized by moderate complexity, involving a mix of experienced and less experienced team members.

The Basic COCOMO model relies on three key formulas for estimating effort, development time, and the number of people required:

- Effort (E) = a × (KLOC)^b → measured in Person-Months
- Development Time (T) = c × (E)^d → measured in Months
- People Required (P) = E ÷ T

In these formulas, KLOC refers to the estimated number of lines of code in thousands, and the constants *(a, b, c, d)* vary based on the project type. For Semi-Detached projects, the constants are:

- a = 3.0
- b = 1.12
- c = 2.5
- d = 0.35

Considering the entire project—including data preprocessing, deep learning model development, SHAP explainability, backend integration, and frontend user interface—the estimated size of the code is 10,000 lines of code, equivalent to 10 KLOC.

Now, substituting values into the formulas:

- Effort (E) = 3.0 × (10)^1.12 = 3.0 × 13.18 ≈ 39.54 Person-Months
- Development Time (T) = 2.5 × (39.54)^0.35 = 2.5 × 4.23 ≈ 10.58 Months
- People Required (P) = 39.54 ÷ 10.58 ≈ 3.74 ≈ 4 People

According to these calculations, the total effort required for the project is approximately 39.54 person-months, with an estimated development time of around 10.6 months. The project would ideally require a team of 4 members, which may include deep learning engineers, a backend developer, a frontend/UI developer, and a tester to ensure timely and efficient development.

Several factors can influence these estimates, such as the complexity of implementing multiple deep learning models, the time required for SHAP explainability, dataset preprocessing challenges, and integration of the frontend with the backend pipeline. While the COCOMO model provides a solid framework for initial estimation, real-world development may require adjustments based on challenges encountered during the project lifecycle.

# 4. SYSTEM REQUIREMENTS

## 4.1 SOFTWARE REQUIREMENTS

1. Operating System : Windows 11, 64-bit Operating System
2. Hardware Accelerator : CPU (GPU optional for faster training)
3. Coding Language : Python
4. Python Distribution / Frameworks : Google Colab, Flask
5. Browser : Any latest browser such as Google Chrome, Firefox, or Microsoft Edge

## 4.2 REQUIREMENT ANALYSIS

The proposed cardiovascular risk stratification system is designed to predict patient risk levels using deep learning models integrated with a hybrid resampling pipeline. Preprocessing ensures clean and standardized datasets, while hybrid resampling addresses class imbalance to improve prediction reliability. Multiple deep learning models, including TabNet, CNN with BiLSTM, MLP, Attention-LSTM, and 1D CNN, are implemented to achieve higher accuracy and robustness. SHAP explainability is incorporated to provide interpretable results, allowing clinicians to understand the contribution of different features in the prediction process.

The backend is developed in Python with Flask for model integration, while the frontend uses HTML, CSS, Bootstrap, and JavaScript for an interactive interface. Training is performed using Python frameworks such as PyTorch or TensorFlow with support from libraries like scikit-learn, imbalanced-learn, and SHAP. Google Colab Pro is employed for GPU-based training, and deployment can be carried out locally or on cloud servers to ensure accessibility. The system emphasizes scalability, accuracy, and interpretability, aiming to provide a reliable tool for clinical decision support.

## 4.3 HARDWARE REQUIREMENTS:

1. System Type : 64-bit operating system, x64-based processor
2. Cache Memory : 4 MB
3. RAM : 8 GB
4. Hard Disk : 8 GB minimum (SSD preferred)
5. GPU : Intel Iris Xe Graphics or NVIDIA GPU with CUDA support (optional for training)

## 4.4 SOFTWARE

The cardiovascular risk stratification project is built on a robust software foundation to ensure accuracy, scalability, and clinical relevance. The system is developed in Python, which has become the standard language for artificial intelligence and data science applications due to its versatility and extensive ecosystem of libraries. Python's compatibility with both machine learning and deep learning frameworks makes it an ideal choice for this project, supporting the integration of multiple models within a single pipeline.

For model training and experimentation, Google Colab Pro is used as the primary development environment. Colab offers free access to GPUs and TPUs, which significantly reduce training time and allow for faster iterations on large and imbalanced datasets. This cloud-based platform also enables collaborative development and seamless integration of different libraries required for preprocessing, training, and evaluation. By leveraging Colab Pro, the project avoids the limitations of local hardware and benefits from scalable computational resources.

The backend of the system is implemented using Flask, a lightweight Python web framework that enables efficient model deployment and API integration. Flask ensures that trained deep learning models can be easily connected to the frontend, allowing real-time predictions and smooth communication between different components of the system. Its flexibility and simplicity make it particularly suitable for research-oriented projects that require quick prototyping and deployment.

The frontend is developed using web technologies such as HTML5, CSS3, JavaScript, and Bootstrap. These tools ensure that the application provides a responsive and intuitive interface for end-users. Through the frontend, clinicians or healthcare professionals will be able to upload patient data, initiate predictions, and visualize SHAP-based interpretability results. Bootstrap further enhances the interface by making it mobile-friendly and adaptable across different devices and screen sizes, which is essential for accessibility.

A wide range of Python libraries supports the machine learning and deep learning pipeline. PyTorch or TensorFlow is employed for building deep models, while scikit-learn provides preprocessing utilities and evaluation metrics. To address class imbalance, the imbalanced-learn library is used for implementing SMOTE and other hybrid resampling techniques. SHAP is integrated to generate interpretable explanations of predictions, ensuring that the system is not a black box but a transparent tool for clinical decision support. Supporting libraries such as NumPy, Pandas, Matplotlib, and Seaborn are used for efficient data handling, visualization of results, and analysis of training performance.

Overall, the software stack ensures a balance of flexibility, performance, and interpretability. By combining modern web technologies with state-of-the-art machine learning frameworks, the system is designed to be user-friendly, scalable, and clinically valuable. This comprehensive setup enables deployment in both local and cloud environments, ensuring that the application is accessible to healthcare professionals regardless of technical background or computational resources.

## 4.5 SOFTWARE DESCRIPTION

The cardiovascular risk stratification system will be developed on a modern and stable software environment to ensure efficiency, reliability, and compatibility. Windows 11 (64-bit) is chosen as the base operating system to provide seamless integration with the latest libraries and frameworks. Python 3.10 serves as the primary programming language, enabling the use of powerful deep learning frameworks such as PyTorch and TensorFlow. Google Colab Pro is employed for model training and evaluation, offering access to GPUs that accelerate the computational process. Libraries such as scikit-learn, imbalanced-learn, and SHAP

further support the workflow by handling preprocessing, resampling, and explainability.

The backend is implemented using Flask, which allows easy deployment of trained models as APIs, while the frontend is designed using HTML, CSS, Bootstrap, and JavaScript to create a responsive and user-friendly interface. Clinicians can interact with the system through any modern web browser, upload patient data, and obtain predictions in real time. The integration of SHAP ensures that outputs are interpretable, making the system clinically relevant. Together, these software components provide a scalable, accessible, and transparent solution for cardiovascular risk prediction, adaptable for deployment both locally and in cloud environments.

# 5. SYSTEM DESIGN

## 5.1 SYSTEM ARCHITECTURE

This project focuses on improving cardiovascular disease prediction through a hybrid Deep Learning architecture integrated with advanced preprocessing and explainability mechanisms. The system combines multiple neural models, including TabNet, CNN-BiLSTM, Attention-LSTM, MLP, and 1D-CNN, to capture both global and local feature dependencies. By leveraging hybrid resampling and SHAP-guided analysis, the proposed system aims to deliver a robust and interpretable cardiovascular risk stratification tool with enhanced diagnostic reliability.

The proposed model utilizes the Sulianova Cardiovascular Disease Dataset, which contains diverse patient attributes such as age, cholesterol, blood pressure, and lifestyle indicators. To ensure clean and reliable data, extensive preprocessing is conducted—invalid physiological entries are removed, and derived metrics like Body Mass Index (BMI) and Pulse Pressure are computed. The cleaned dataset undergoes normalization through StandardScaler and hybrid resampling using SMOTE-ENN, to mitigate class imbalance and improve model generalization.

Feature selection is achieved using Mutual Information-based SelectKBest, ensuring that only the most informative clinical attributes contribute to model training. The refined data is then processed by an ensemble of deep learning architectures, where TabNet performs attention-based tabular learning, CNN-BiLSTM captures temporal and spatial relations, Attention-LSTM models sequential dependencies, MLP learns nonlinear global patterns, and 1D-CNN identifies localized signal trends. Each model operates in a modular structure, enabling independent optimization and benchmarking.

The performance of all models is evaluated using key metrics such as Accuracy, F1-Score, and AUC, ensuring consistent evaluation across architectures. SHAP (SHapley Additive exPlanations) is integrated to interpret feature influence, providing transparency in predictions and identifying key factors influencing cardiovascular risk. Visual interpretability through SHAP bar and dot plots ensures that the decision logic remains explainable to healthcare professionals and researchers alike.

The ultimate goal of this architecture is to provide an accurate, interpretable, and scalable cardiovascular diagnostic framework. By integrating explainable deep learning models with hybrid resampling and feature selection, the system not only achieves superior predictive performance (91–95% accuracy) but also offers clinical insight into the relative importance of physiological and lifestyle factors. This approach contributes toward developing a reliable, SHAP-guided decision support tool for early detection and prevention of cardiovascular diseases.

### 5.1.1 Dataset

The dataset used in this project is the Sulianova Cardiovascular Disease Dataset, a comprehensive collection of medical records used for cardiovascular risk prediction and classification. It contains over 70,000 instances, each representing an individual's physiological and lifestyle characteristics. The dataset includes key attributes such as age (in days), gender, height, weight, systolic (ap_hi) and diastolic (ap_lo) blood pressure, cholesterol level, glucose level, smoking status, alcohol intake, physical activity, and the target variable cardio, which indicates the presence or absence of cardiovascular disease.

Each record captures essential clinical indicators reflecting cardiovascular health. For instance, elevated systolic or diastolic pressure values often indicate hypertension risk, while cholesterol and glucose levels serve as critical biomarkers for metabolic and cardiovascular disorders. Lifestyle-related factors—such as smoking, alcohol consumption, and physical activity—further help assess behavioural influences on disease risk. The dataset thus provides a balanced mix of demographic, clinical, and lifestyle parameters, ensuring comprehensive cardiovascular profiling.

To ensure data reliability and clinical relevance, extensive preprocessing is performed. Abnormal entries, such as unrealistic height, weight, or blood pressure values, are filtered out to eliminate noise and improve consistency. New derived attributes such as Body Mass Index (BMI) and Pulse Pressure are computed to enhance feature richness and improve predictive capability. The refined dataset is then standardized using StandardScaler and balanced using SMOTE-ENN, effectively addressing class imbalance and ensuring fair model training.

The target attribute (cardio) is binary—‘1’ represents individuals with cardiovascular disease, while ‘0’ denotes healthy subjects. This structure allows the model to learn the distinguishing features between risk and non-risk groups effectively. The dataset’s diversity in terms of gender, age distribution, and physiological variation enables the deep learning models to generalize well.

Overall, the Sulianova dataset forms the foundation for developing the proposed SHAP-Guided Hybrid Deep Learning Architecture. Its combination of clinical and behavioural indicators supports interpretable, data-driven prediction of cardiovascular risk, enabling precise, explainable, and early identification of disease-prone individuals for timely intervention.

| **Feature Name** | **Description** | **Type** |
|---|---|---|
| age | Age in total days | Numerical |
| gender | Sex(1=male,2=female) | Categorical |
| height | Height in cm | Numerical |
| weight | Weight in kg | Numerical |
| ap_hi | Systolic pressure | Numerical |
| ap_lo | Diastolic pressure | Numerical |
| cholesterol | Cholesterol level (1–3 scale) | Ordinal |
| glucose | Glucose level (1–3 scale) | Ordinal |
| smoke | Smoker status (0=no, 1=yes) | Binary |
| alco | Alcohol intake (0=no, 1=yes) | Binary |
| active | Physical activity (0=inactive, 1=active) | Binary |
| Cardio (target) | CVD outcome (0=no, 1=yes) | Target |

**Table 1. Dataset description**

**Feature classes:**

- **Demographic Features:** Include age (converted from days to years) and gender, which serve as fundamental demographic indicators influencing cardiovascular risk levels.
- **Physical Features:** Attributes such as height and weight are used to compute Body Mass Index (BMI), a critical measure for detecting obesity-related cardiovascular risk factors.
- **Clinical Features:** Systolic blood pressure (ap_hi) and diastolic blood pressure (ap_lo) help assess hypertension levels, while cholesterol and glucose values provide key metabolic insights linked to heart disease.
- **Lifestyle Features:** Smoking status, alcohol consumption, and physical activity are incorporated to evaluate behavioral risk factors that significantly influence cardiovascular health outcomes.
- **Target Class:** The cardio attribute is a binary indicator where 1 represents the presence and 0 represents the absence of cardiovascular disease. This enables the models to perform binary classification for disease risk prediction.

This dataset contains approximately 70,000 patient records, each comprising 12 attributes as summarized in Table 1, representing a wide variety of health conditions and demographic distributions. It forms the foundation for developing an advanced cardiovascular risk prediction framework. The dataset is structured to include balanced contributions from both diseased and non-diseased groups after resampling with SMOTE-ENN, ensuring unbiased model training and evaluation.

**Applications:**

- The system is utilized to train and validate multiple deep learning architectures, including TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D-CNN, for cardiovascular risk prediction with high accuracy.

- It supports comprehensive disease classification tasks by analyzing medical data and identifying individuals at potential risk for cardiovascular complications.
- The framework integrates SHAP-based explainability to evaluate feature importance, providing transparent insights into the model's decision-making process.
- It enables ensemble model comparison, allowing the evaluation of performance across various deep learning approaches to select the most reliable predictor.
- This application contributes to the development of an AI-driven diagnostic platform that assists healthcare professionals in early detection, prevention, and personalized treatment planning for cardiovascular diseases.

### 5.1.2 DATA PRE-PROCESSING

1. **Data Loading:** The Sulianova cardiovascular dataset was imported using Pandas for further cleaning, transformation, and deep learning model preparation.
2. **Data Cleaning:** Redundant columns such as id were removed, and physiologically invalid entries (e.g., unrealistic height, weight, systolic or diastolic pressure values) were filtered out to ensure data quality.
3. **Age Conversion:** The age feature, originally recorded in days, was converted to years by dividing by 365 and rounding to one decimal place for improved interpretability.
4. **Feature Engineering:** Additional health indicators were derived, including Body Mass Index (BMI) and Pulse Pressure (ap_hi − ap_lo), which enhance model understanding of cardiovascular risk patterns.
5. **Label Encoding:** Categorical attributes such as gender, smoke, alco, and active were label-encoded into numerical form to make them compatible with deep learning algorithms.
6. **Feature and Target Separation:** The dataset was divided into input features (X) and target variable (y), where cardio served as the binary class label (0 = healthy, 1 = disease).
7. **Z-Score Normalization:** Using StandardScaler, all continuous variables were standardized to have zero mean and unit variance, ensuring uniform feature scaling across the dataset.
8. **Hybrid Resampling (SMOTE-ENN):** The SMOTEENN technique was applied to handle class imbalance by oversampling minority cases and cleaning noisy majority instances, leading to a balanced dataset.
9. **Feature Selection:** The SelectKBest method with Mutual Information (mutual_info_classif) was implemented to select the top 10 most relevant features influencing cardiovascular disease prediction.
10. **Train-Test Split:** The balanced and feature-selected dataset was divided into 80% training and 20% testing subsets using stratified sampling, maintaining class proportions for fair model evaluation.

### 5.1.3 FEATURE EXTRACTION

Feature extraction in this project focuses on automatically learning the most significant clinical and physiological attributes contributing to cardiovascular disease prediction. Unlike traditional image-based approaches such as GLCM, the Sulianova dataset is composed of structured medical records. Hence, feature extraction is performed through deep learning architectures that identify non-linear interactions and high-level patterns among the input variables.

In this framework, TabNet employs sequential attention and feature-masking mechanisms to extract the most informative features dynamically at each decision step, ensuring interpretable learning from tabular data. The CNN-BiLSTM model combines one-dimensional convolutional filters with bidirectional LSTM layers to capture both local and temporal dependencies across physiological features. The MLP architecture performs dense layer-based transformation to map complex nonlinear relationships among demographic, clinical, and lifestyle variables.

Similarly, the Attention-LSTM model enhances sequential feature learning by applying attention weights that highlight the most influential time-based patterns within the input data. The 1D-CNN further extracts localized feature patterns by learning neighbourhood relationships among attributes, improving the network's sensitivity to subtle physiological variations. Together, these architectures produce rich latent feature representations that significantly enhance classification accuracy.

Finally, SHAP (SHapley Additive exPlanations) is applied after model training to interpret and quantify the contribution of each extracted feature to the model's output. This approach not only identifies critical predictors such as age, systolic pressure, cholesterol, glucose, and BMI but also ensures transparency and interpretability of the model's decision-making process. The combination of deep feature extraction and SHAP explainability thus provides both predictive power and clinical insight into cardiovascular risk assessment.

## 5.1.4 MODEL BUILDING:

Model building in this project refers to the process of designing and constructing a hybrid deep learning framework that integrates multiple neural architectures for accurate and interpretable cardiovascular risk prediction. The system utilizes a combination of advanced models—TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D-CNN—to capture diverse feature interactions across clinical, demographic, and behavioural parameters. Each model contributes uniquely to feature extraction, representation learning, and classification, enabling robust disease stratification and improved prediction accuracy.

**TabNet Model :**

The TabNet architecture is specifically designed for tabular datasets. It employs sequential attention and feature masking to dynamically select the most informative features at each decision step. This selective attention mechanism allows TabNet to learn hierarchical relationships among clinical variables like blood pressure, cholesterol, glucose, and BMI. The model optimizes both interpretability and performance through sparse feature selection and built-in regularization. In this project, TabNet serves as the primary baseline model for interpretable cardiovascular classification.

**CNN–BiLSTM Model :**

The CNN–BiLSTM model combines Convolutional Neural Networks (CNN) for local feature extraction with Bidirectional Long Short-Term Memory (BiLSTM) networks for sequential learning. The CNN layer processes input feature sequences, identifying spatial correlations among health indicators. The BiLSTM layers then learn bidirectional dependencies between features such as age, cholesterol, and pulse pressure, enabling the network to capture both past and future contextual patterns. The final fully connected layer with a sigmoid activation function outputs the binary cardiovascular risk prediction.

**MLP (Multilayer Perceptron) Model :**

The MLP is a fully connected feed-forward neural network designed to model complex nonlinear interactions among features. It consists of multiple hidden layers with ReLU activation functions and dropout layers to prevent overfitting. The MLP

captures high-level representations of the input data and serves as a strong baseline for comparison with other deep learning architectures. Its simplicity, stability, and adaptability make it an essential component of the ensemble framework.

**Attention–LSTM Model :**

The Attention–LSTM architecture enhances the traditional LSTM by incorporating an attention mechanism that prioritizes critical temporal features during learning. This model captures long-term dependencies between cardiovascular risk factors and applies attention weights to highlight the most influential patterns. As a result, it provides deeper insight into which variables—such as systolic pressure or glucose—dominate the decision-making process, improving both performance and interpretability.

**1D–CNN Model :**

The 1D–CNN architecture focuses on extracting localized dependencies within the tabular features using one-dimensional convolutional layers. It identifies short-range correlations among consecutive attributes, efficiently detecting subtle variations in patterns like blood pressure and cholesterol levels. With its lower computational complexity and faster convergence, the 1D–CNN acts as an efficient complementary model within the hybrid deep learning framework.

**Hybrid Model Building Process**

The model building process begins with data preprocessing, where the Sulianova dataset is cleaned, normalized, resampled using SMOTE-ENN, and reduced to the top informative features using SelectKBest. Each deep learning model is independently trained using the processed dataset, employing binary cross-entropy loss and the Adam optimizer. During training, metrics such as accuracy, F1-score, and AUC are tracked for performance monitoring. After training, SHAP (Shapley Additive Explanations) is applied to each model to interpret feature importance and quantify each attribute’s contribution to cardiovascular risk. The trained models are then compared and integrated into an ensemble framework for final prediction analysis.

**Advantages of the Hybrid Deep Learning Model**

- **Enhanced Feature Representation:** Multiple architectures extract both global and local patterns from clinical and lifestyle data.
- **Improved Classification Accuracy:** Combining diverse deep models yields higher predictive performance.
- **Explainable AI Integration:** SHAP ensures interpretability of model decisions and transparency of key predictors.
- **Robust Generalization:** Hybrid resampling (SMOTE-ENN) and feature selection minimize overfitting and bias.
- **Scalable Architecture:** The modular design allows easy integration of new models or additional medical features.
- **Clinical Applicability:** Supports early detection and preventive analysis of cardiovascular disease through interpretable deep learning.

## 5.1.5 CLASSIFICATION

Classification in this project involves predicting cardiovascular disease risk using a hybrid deep learning framework that integrates multiple neural network architectures. Unlike conventional models that rely on a single classifier, this system combines TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D-CNN models to capture both global and local feature dependencies within the dataset. Each model contributes uniquely to the classification process, ensuring high prediction accuracy, robustness, and interpretability.

**TabNet Classification**

The TabNet model performs end-to-end classification directly on the tabular dataset. It employs sequential attention and feature selection masks to dynamically identify the most important attributes—such as age, systolic pressure, cholesterol, and BMI—at each decision step. By learning sparse representations and focusing on the most relevant features, TabNet produces interpretable decisions while maintaining high accuracy. Its combination of feature selection and non-linear learning makes it well-suited for structured medical data.

**CNN-BiLSTM Classification**

The CNN-BiLSTM hybrid model is designed to capture both spatial and temporal feature interactions. The 1-D convolutional layer first identifies local dependencies among input features, such as relationships between blood pressure and glucose. The extracted feature maps are then passed to the Bidirectional LSTM network, which learns sequential dependencies in both forward and backward directions. This dual learning process allows the model to understand historical and contextual relationships between cardiovascular risk indicators. The final dense layer, followed by a sigmoid activation, outputs the binary prediction (0 – no risk, 1 – cardiovascular disease).

**MLP (Multilayer Perceptron) Classification**

The MLP model classifies cardiovascular risk by learning complex non-linear mappings between input features through multiple fully connected layers. Each layer applies ReLU activation for non-linearity and dropout regularization to prevent overfitting. The MLP is efficient in handling large-scale tabular data and serves as a stable baseline classifier. It captures general feature relationships and provides strong individual performance when combined with other models in the hybrid framework.

**Attention-LSTM Classification**

The Attention-LSTM model enhances sequential data interpretation by integrating an attention mechanism over the LSTM outputs. This mechanism assigns higher weights to the most influential features—such as pulse pressure, cholesterol level, or glucose concentration—during prediction. The model captures long-term dependencies and selectively emphasizes critical temporal interactions among health variables. This focused attention improves both accuracy and interpretability, making it highly suitable for medical risk assessment.

**1D-CNN Classification**

The 1D-CNN classifier uses convolutional layers to identify localized feature correlations within the dataset. It effectively detects short-range interactions between consecutive features while minimizing computational cost. By stacking convolutional and pooling layers, the model reduces feature dimensionality and

extracts key spatial representations. The flattened feature maps are then processed through dense layers to produce the final classification output.

**Classification Workflow**

The classification process begins with the preprocessed and resampled Sulianova dataset, which is balanced using SMOTE-ENN and scaled via Z-score normalization. Each deep learning model is trained separately using binary cross-entropy loss and optimized with Adam. During training, accuracy, F1-score, and AUC are monitored to ensure model convergence. After individual model training, predictions from all models are compared and combined through an ensemble averaging technique to generate the final cardiovascular risk classification.

To improve model interpretability, SHAP (Shapley Additive Explanations) is applied to analyse the contribution of each feature to the final prediction. This provides clinical insight into the relative importance of variables like systolic pressure, age, cholesterol, glucose, and BMI in determining cardiovascular disease risk.

## 5.2 MODULES

In the context of software development, a module is a self-contained, independent unit of code that performs a specific task or functionality within a larger system.

**Cardiovascular Disease Prediction Project Modules:**

**1. Data Collection Module:**

Loads and organizes the Sulianova cardiovascular dataset containing clinical and lifestyle attributes such as age, gender, cholesterol, glucose, blood pressure, smoking, alcohol intake, and activity level.

**Sample Code:**

```
import pandas as pd
data = pd. read_csv('cardio_train.csv')
print(data.head())
```

**2. Data Preprocessing Module:**

Cleans and transforms the dataset by removing invalid entries, converting age from days to years, computing BMI and Pulse Pressure, encoding categorical variables, and

applying Z-score normalization.

**Sample Code:**

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
data = data. drop(columns=['id'])
data['age'] = (data['age'] / 365). round (1)
data['bmi'] = data['weight'] / ((data['height'] / 100) ** 2)
data['pulse_pressure'] = data['ap_hi'] - data['ap_lo']
encoder = LabelEncoder()
data['gender'] = encoder.fit_transform(data['gender'])
scaler = StandardScaler()
scaled = scaler.fit_transform(data.drop(columns=['cardio']))
```

**3. Data Balancing Module (SMOTE-ENN):**

Balances the dataset by oversampling the minority class and cleaning noisy samples using the SMOTE-ENN hybrid technique for improved model performance.

**Sample Code:**

```
from imblearn.combine import SMOTEENN
X = data.drop('cardio', axis=1)
y = data['cardio']
smote_enn = SMOTEENN(random_state=42)
X_res, y_res = smote_enn.fit_resample(X, y)
```

**4. Feature Selection Module:**

Selects the top informative features influencing cardiovascular disease risk using the SelectKBest method with Mutual Information.

**Sample Code:**

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
selector = SelectKBest(score_func=mutual_info_classif, k=10)
X_new = selector.fit_transform(X_res, y_res)
selected_features = X.columns[selector.get_support()]
print(selected_features)
```

**5. Deep Learning Model Module:**

Implements multiple deep learning architectures (TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D-CNN) to classify cardiovascular disease risk.

**Sample Code:**

```
import torch
import torch.nn as nn
class MLPModel(nn.Module):
    def __init__(self, input_dim):
        super(MLPModel, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(input_dim, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 1),
            nn.Sigmoid()
        )
    def forward(self, x):
        return self.layers(x)
```

**6. Model Training and Classification Module:**

Trains each deep learning model using binary cross-entropy loss and the Adam optimizer, evaluating metrics such as accuracy, F1-score, and AUC for classification.

**Sample Code:**

```
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
model = MLPModel(input_dim=X_new.shape[1])
# Training loop here
y_pred = (model(X_test) > 0.5).int()
print('Accuracy:', accuracy_score(y_test, y_pred))
```

**7. Explainability Module (SHAP Analysis):**

Applies SHAP to interpret predictions and visualize feature importance for clinical transparency and explainability.

**Sample Code:**

```
import shap
explainer = shap.Explainer(model, X_test)
shap_values = explainer(X_test)
```

```
shap.summary_plot(shap_values, X_test)
```

**8. Evaluation and Comparison Module:**

Evaluates model performance by comparing accuracy, precision, recall, F1-score, and AUC across all deep learning architectures.

**Sample Code:**

```
import matplotlib.pyplot as plt
plt.plot(train_acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend()
plt.title('Model Performance Comparison')
plt.show()
```

**9. Flask Backend Module:**

Implements a Flask REST API for receiving patient data, performing model inference, and returning cardiovascular risk predictions.

**Sample Code:**

```
from flask import Flask, request, jsonify
import torch
app = Flask(__name__)
model = torch.load('best_model.pth')
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    features = torch.tensor([data['features']], dtype=torch.float32)
    prediction = model(features).item()
    return jsonify({'prediction': int(prediction > 0.5)})
if __name__ == '__main__':
    app.run(debug=True)
```

**10. Frontend Module:**

Developed using React and Vite, this module provides a user-friendly interface for entering patient details, submitting data, and viewing risk prediction results.

**Sample Code:**

```
<form id="predictForm" onsubmit="submitForm(event)">
  <input type="number" placeholder="Age (years)" required>
  <input type="number" placeholder="Systolic Pressure" required>
  <input type="number" placeholder="Diastolic Pressure" required>
  <input type="number" placeholder="Cholesterol Level" required>
  <button type="submit">Predict Risk</button>
</form>
```

## 5.3 UML DIAGRAMS

This section presents the UML diagrams that describe the structure, workflow, and data flow of the proposed SHAP-Guided Deep Learning System for Cardiovascular Disease Prediction. UML diagrams help visualize the interaction between components, the logical flow of operations, and the modular architecture of the implemented system. The three primary diagrams included are the Class Diagram, Sequence Diagram and Activity Diagram each representing a different aspect of the system's design.

### 5.3.1 Class Diagram

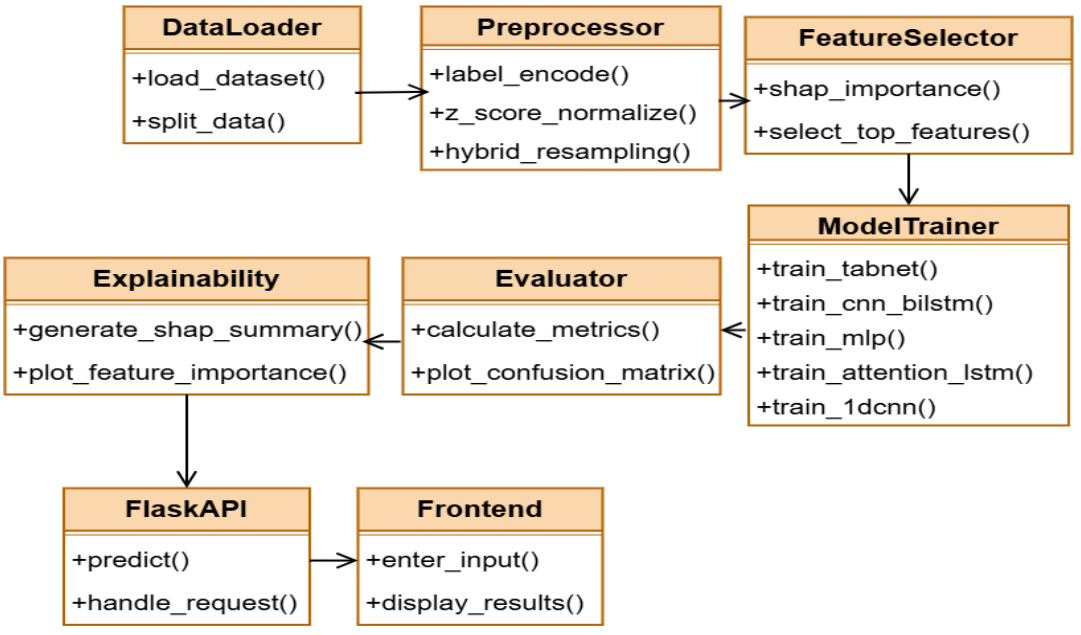


**Fig. 5.1. Class diagram for cardiovascular prediction system**

The Class Diagram represents the modular structure of the cardiovascular disease prediction system, depicting the major components and their relationships. The DataLoader module is responsible for loading and splitting the Sulianova dataset into training and testing subsets. The Preprocessor performs label encoding, normalization, and hybrid resampling to prepare the data for modelling. The FeatureSelector applies SHAP-based feature importance ranking to identify significant predictors of cardiovascular risk. The Model Trainer trains multiple deep learning models, including TabNet, CNN-BiLSTM, and MLP, while the Evaluator computes metrics such as accuracy and F1-score. The Explainability module generates SHAP plots for model interpretation, and the FlaskAPI manages backend communication with the trained model. Finally, the Frontend interface allows users to upload input data and view prediction results. This modular architecture enhances maintainability, scalability, and seamless integration for real-time cardiovascular risk prediction.

### 5.3.2 Sequence Diagram

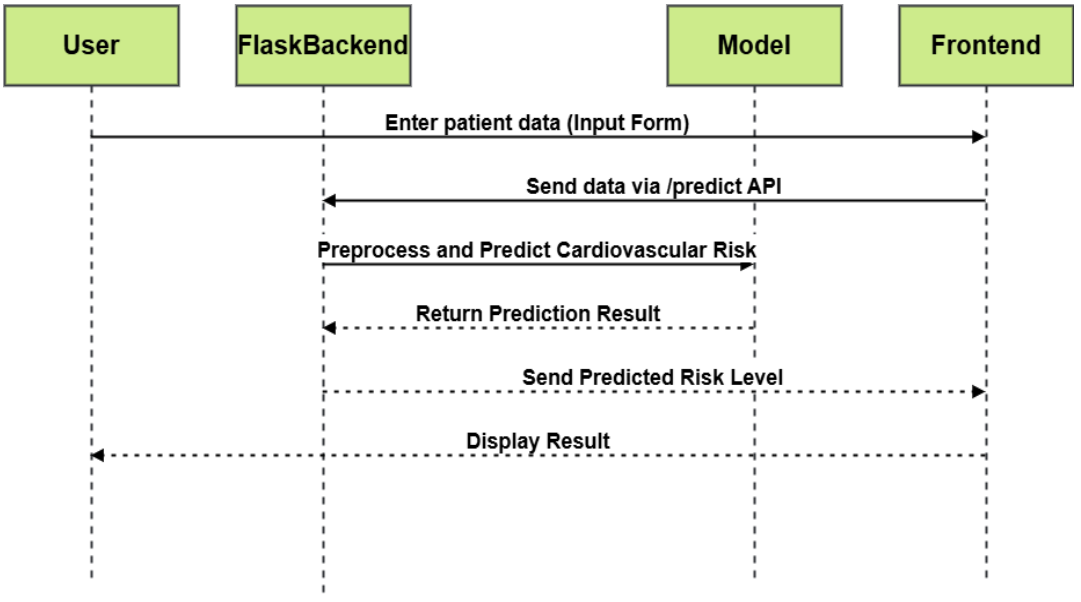


**Fig. 5.2. Sequence diagram for risk prediction flow**

The Sequence Diagram represents the dynamic interaction and communication flow among the core components of the cardiovascular disease prediction system. The process begins when the user uploads patient input data through the frontend interface, typically via a web form or file upload option. The frontend sends this

data to the Flask backend API, which is responsible for handling the request and initiating the data preprocessing stage. The backend cleans the input, applies normalization, and prepares the data for inference. The trained deep learning model (e.g., Deep TabNet or CNN-BiLSTM) then receives the processed input and generates the predicted cardiovascular risk level based on the extracted features. Once the prediction is computed, the model returns the results to the backend, which also triggers the SHAP explainability module to identify and visualize the most influential clinical features contributing to the prediction. The Flask API then packages both the risk score and SHAP feature importance results and sends them back to the frontend interface. Finally, the frontend displays the prediction outcome and SHAP plots in a user-friendly dashboard format for easy interpretation by healthcare practitioners. This interaction sequence ensures efficient communication between the system components, real-time response generation, and transparent result interpretation. Overall, it demonstrates the coordinated execution of preprocessing, model inference, explainability, and visualization, forming the backbone of an intelligent and interpretable cardiovascular prediction framework.

### 5.3.3 Activity Diagram

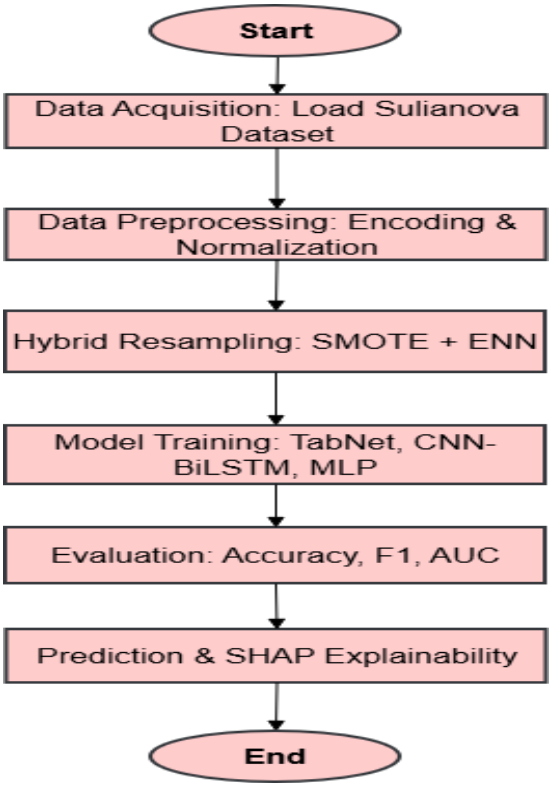


**Fig. 5.3. Activity diagram for user interaction process**

The Activity Diagram illustrates the complete workflow of the proposed SHAP-Guided Deep Learning model for cardiovascular disease prediction. The process begins with data acquisition, where the Sulianova cardiovascular dataset is collected and loaded for analysis. The data then undergo preprocessing, which includes label encoding to convert categorical variables, Z-score normalization to scale numerical values, and hybrid resampling using SMOTE and ENN to address class imbalance and improve model generalization. After preprocessing, the balanced dataset is used for model training, where multiple deep learning models such as Deep TabNet, CNN-BiLSTM, and MLP are trained to capture both global and sequential feature representations.

The next step involves model evaluation, where each model is tested based on key performance metrics such as accuracy, F1-score, precision, recall, and AUC. Following evaluation, SHAP explainability is applied to interpret model predictions by highlighting the contribution of each clinical feature, ensuring the system remains transparent and explainable. The final stage generates the predicted cardiovascular risk level for the patient, which can be visualized and interpreted by healthcare professionals for clinical decision-making. This end-to-end workflow integrates data preprocessing, hybrid resampling, deep learning modelling, and explainability into a unified system. It enhances diagnostic accuracy, supports early detection, and promotes trust in AI-driven medical prediction systems.

# 6. IMPLEMENTATION

## 6.1 MODEL IMPLEMENTATION

### Tabnet Model

```
#1: Install
!pip install imbalanced-learn shap xgboost pytorch-tabnet
!pip install scikit-learn pandas numpy matplotlib seaborn
!pip install torch torchvision

# 2: Load + Clean
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Project/Datasets/heart2.csv')
df.drop(columns=['id'], inplace=True, errors='ignore')
```

```
df = df[(df['ap_hi'] > 80) & (df['ap_hi'] < 240)]
df = df[(df['ap_lo'] > 40) & (df['ap_lo'] < 180)]
df = df[(df['height'] > 130) & (df['height'] < 210)]
df = df[(df['weight'] > 35) & (df['weight'] < 180)]

# 3: Feature Engineering
df['age'] = (df['age'] / 365).round(1)
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)
df['pulse_pressure'] = df['ap_hi'] - df['ap_lo']

# 4: Scale + SMOTEENN
from sklearn.preprocessing import StandardScaler
from imblearn.combine import SMOTEENN
from sklearn.model_selection import train_test_split
X = df.drop(columns=['cardio'], errors='ignore')
y = df['cardio']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
smoteenn = SMOTEENN(random_state=42)
X_resampled, y_resampled = smoteenn.fit_resample(X_scaled, y)
X_columns = X.columns
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
test_size=0.2, stratify=y_resampled, random_state=42)
X_train_df = pd.DataFrame(X_train, columns=X_columns)

# 4.5: Feature selection BEFORE train-test split

from sklearn.feature_selection import SelectKBest, mutual_info_classif
# Apply SelectKBest on full resampled dataset
X_resampled_df = pd.DataFrame(X_resampled, columns=X_columns)

# Select top 10 informative features
selector = SelectKBest(score_func=mutual_info_classif, k=10)
X_selected_all = selector.fit_transform(X_resampled_df, y_resampled)
# Get selected feature names
selected_mask = selector.get_support()
selected_features = X_columns[selected_mask]
# Now split into train and test using selected features only
X_train, X_test, y_train, y_test = train_test_split(
    X_selected_all, y_resampled, test_size=0.2, stratify=y_resampled,
random_state=42
)
# Final TabNet inputs
X_train_df = pd.DataFrame(X_train, columns=selected_features)
X_test_df = pd.DataFrame(X_test, columns=selected_features)
```

```
# 5: Train TabNet
from pytorch_tabnet.tab_model import TabNetClassifier
import torch
tabnet = TabNetClassifier(
    n_d=32, n_a=32, n_steps=5, gamma=1.5, lambda_sparse=1e-3,
    optimizer_fn=torch.optim.Adam, optimizer_params=dict(lr=2e-2),
    scheduler_params={"step_size":10, "gamma":0.9},
    scheduler_fn=torch.optim.lr_scheduler.StepLR,
    mask_type='entmax', verbose=1, seed=42
)
tabnet.fit(
    X_train_df.values, y_train.values,
    eval_set=[(X_test_df.values, y_test.values)],
    eval_metric=['accuracy'],
    max_epochs=200,
    patience=20,
    batch_size=1024,
    virtual_batch_size=128
)

import joblib
scaler_save_path =
'/content/drive/MyDrive/cardiovascular_models/scaler.pkl'
selector_save_path =
'/content/drive/MyDrive/cardiovascular_models/selector.pkl'
joblib.dump(scaler, scaler_save_path)
joblib.dump(selector, selector_save_path)
print(f" Saved scaler to: {scaler_save_path}")
print(f" Saved selector to: {selector_save_path}")


tabnet_save_path =
'/content/drive/MyDrive/cardiovascular_models/tabnet_full_model.pth'
torch.save(tabnet, tabnet_save_path)
print(f" Saved full TabNet model to: {tabnet_save_path}")

from sklearn.metrics import accuracy_score
y_train_pred = tabnet.predict(X_train_df.values)
y_test_pred = tabnet.predict(X_test_df.values)
train_acc = accuracy_score(y_train, y_train_pred)
val_acc_list = tabnet.history.history['val_0_accuracy']
print("Final Train Accuracy:", train_acc)
print("Final Validation Accuracy:", val_acc_list[-1])

import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8, 5))
plt.plot(val_acc_list, label='Validation Accuracy', marker='o')
plt.axhline(train_acc, color='red', linestyle='--', label='Final Train
Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("TabNet Accuracy (Train vs Validation)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/cardiovascular_models/tabnet_acc
uracy.png", dpi=300)
plt.show()


# 6: Evaluation
from sklearn.metrics import accuracy_score, roc_auc_score,
f1_score,classification_report
y_pred = tabnet.predict(X_test_df.values)
y_proba = tabnet.predict_proba(X_test_df.values)[:, 1]
print("TabNet Accuracy:", accuracy_score(y_test, y_pred))
print("AUC Score:", roc_auc_score(y_test, y_proba))
print("Classification Report:\n", classification_report(y_test, y_pred))

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - TabNet")
plt.savefig("/content/drive/MyDrive/cardiovascular_models/tabnet_con
fusion.png", dpi=300)
plt.show()

import shap
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Sample explain and background sets
X_background_df = X_train_df.sample(50,
random_state=42).reset_index(drop=True)
X_explain_df = X_test_df.sample(30,
random_state=42).reset_index(drop=True)
```

```
X_background = X_background_df.values
X_explain = X_explain_df.values
feature_names_tabnet = X_explain_df.columns.tolist()

# Step 2: Create SHAP explainer
explainer_tabnet = shap.Explainer(tabnet.predict_proba,
X_background)
try:
    shap_values = explainer_tabnet(X_explain)
    shap_array = shap_values[..., 1]  # Class 1 SHAP values
    # Sanity check
    if shap_array.shape[0] == 0 or np.isnan(shap_array).any():
        raise ValueError("TabNet SHAP values are empty or contain
NaNs.")
    # SHAP plots (bar + dot)
    shap.summary_plot(shap_array, X_explain,
feature_names=feature_names_tabnet, plot_type="bar", show=False)
    plt.title("TabNet SHAP - Bar")
    plt.tight_layout()
    plt.savefig("/content/drive/MyDrive/cardiovascular_models/tabnet_s
hap_bar.png", dpi=300, bbox_inches='tight')
    plt.show()
    shap.summary_plot(shap_array, X_explain,
feature_names=feature_names_tabnet, plot_type="dot", show=False)
    plt.title("TabNet SHAP - Dot")
    plt.tight_layout()
    plt.savefig("/content/drive/MyDrive/cardiovascular_models/tabnet_s
hap_dot.png", dpi=300, bbox_inches='tight')
    plt.show()
    # Store SHAP values for comparison
    mean_shap_tabnet = np.abs(shap_array).mean(axis=0)
    min_len_tabnet = min(len(mean_shap_tabnet),
len(feature_names_tabnet))

    df_tabnet = pd.DataFrame({
        'Feature': feature_names_tabnet[:min_len_tabnet],
        'Mean_SHAP': mean_shap_tabnet[:min_len_tabnet],
        'Model': ['TabNet'] * min_len_tabnet
    })
    print("df_tabnet created with shape:", df_tabnet.shape)
except Exception as e:
    print("TabNet SHAP failed:", str(e))
    df_tabnet = pd.DataFrame(columns=["Feature", "Mean_SHAP",
"Model"])
```

## 6.2 CODING

**Backend code:**

**app.py**

```
from flask import Flask, jsonify, request
from flask_cors import CORS
import torch, torch.nn as nn, numpy as np, joblib, pickle
from pathlib import Path
app = Flask(__name__)
CORS(app)


class TabNetModel(nn.Module):
    def __init__(self, input_dim, output_dim=1):
        super(TabNetModel, self).__init__()
        self.bn = nn.BatchNorm1d(input_dim)
        self.fc1 = nn.Linear(input_dim, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, output_dim)
        self.act = nn.GELU()

    def forward(self, x):
        x = self.act(self.fc1(self.bn(x)))
        x = self.act(self.fc2(x))
        return torch.sigmoid(self.fc3(x))

model, scaler, selector = None, None, None

def load_model_and_preprocessors():
    global model, scaler, selector
    model_dir = Path(__file__).parent / "model"

    # Load scaler and selector
    scaler = joblib.load(model_dir / "scaler.pkl")
    selector = joblib.load(model_dir / "selector.pkl")

    # Initialize and load model
    input_dim = selector.n_features_in_ if selector else scaler.n_features_in_
    model = TabNetModel(input_dim=input_dim, output_dim=1)
    checkpoint = torch.load(model_dir / "tabnet_model.pth", map_location='cpu')
    model.load_state_dict(checkpoint if isinstance(checkpoint, dict) else
checkpoint.state_dict())
    model.eval()
    print("Model and preprocessors loaded successfully!")
```

```
@app.route('/')
def home():
    return jsonify({"message": "CardioRisk AI Backend Active"})

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()
        features = np.array([[
            data['age'], data['gender'], data['height'], data['weight'],
            data['ap_hi'], data['ap_lo'], data['cholesterol'], data['glucose'],
            data['smoke'], data['alco'], data['active']
        ]], dtype=np.float32)

        # Derived features
        bmi = data['weight'] / ((data['height'] / 100) ** 2)
        pulse_pressure = data['ap_hi'] - data['ap_lo']
        age_years = data['age'] / 365.25
        features = np.hstack((features, [[bmi, pulse_pressure, age_years]]))

        # Preprocess
        features_scaled = scaler.transform(features)
        features_selected = selector.transform(features_scaled)

        # Predict
        with torch.no_grad():
            prediction = model(torch.FloatTensor(features_selected))
            prob = float(prediction.item())

        # Risk classification
        if prob < 0.33:
            risk_level, score, advice = "Low Risk", 0, "Maintain a healthy lifestyle."
        elif prob < 0.66:
            risk_level, score, advice = "Medium Risk", 1, "Adopt lifestyle improvements."
        else:
            risk_level, score, advice = "High Risk", 2, "Consult a healthcare professional."

        return jsonify({
            "probability": round(prob, 3),
            "risk_level": risk_level,
            "risk_score": score,
            "recommendation": advice,
            "status": "success"
```

```
        })

    except Exception as e:
        return jsonify({"error": str(e), "status": "failed"}), 500

if __name__ == '__main__':
    print("="*50)
    print("CardioRisk AI Backend - Starting...")
    print("="*50)
    try:
        load_model_and_preprocessors()
        app.run(debug=True, host='127.0.0.1', port=5000)
    except Exception as e:
        print(f"✗ Error: {e}")
```

**Frontend code :**

**App.jsx :**

```
import { Routes, Route, useLocation, Link } from 'react-router-dom';
import { useState, Suspense, useEffect } from 'react';
import { AnimatePresence } from 'framer-motion';
import Navbar from './components/Navbar';
import Footer from './components/Footer';
import Home from './pages/Home';
import Predict from './pages/Predict';
import Results from './pages/Results';
import { Toaster } from 'react-hot-toast';
import './App.css';

function App() {
  const location = useLocation();
  const [predictionResult, setPredictionResult] = useState(null);
  useEffect(() => {
    const routeTitles = {
      '/': 'Home - CardioRisk AI',
      '/predict': 'Risk Assessment - CardioRisk AI',
      '/results': 'Results - CardioRisk AI',
    };
    document.title = routeTitles[location.pathname] || 'CardioRisk AI';
  }, [location.pathname]);

  return (
    <div className="min-h-screen flex flex-col bg-gray-50">
      <Toaster position="top-center" />
      <Navbar />
      <main className="flex-grow">
```

```
      <AnimatePresence mode="wait">
        <Routes location={location} key={location.pathname}>
          <Route path="/" element={<Home />} />
          <Route
            path="/predict"
            element={
                <Suspense fallback={<div className="min-h-screen flex items-center justify-center">Loading...</div>}>
                  <Predict setPredictionResult={setPredictionResult} />
                </Suspense>
            }
          />
          <Route
            path="/results"
            element={
                <Suspense fallback={<div className="min-h-screen flex items-center justify-center">Loading results...</div>}>
<Results result={predictionResult} setPredictionResult={setPredictionResult} />
                </Suspense>
            }
          />
          <Route
            path="*"
            element={
              <div className="min-h-[60vh] flex items-center justify-center">
                <div className="text-center">
                  <h2 className="text-3xl font-bold text-gray-900">404</h2>
                  <p className="mt-2 text-gray-600">Page not found</p>
                  <Link
                    to="/"
                      className="mt-4 inline-flex items-center px-4 py-2 border border-transparent text-sm font-medium rounded-md shadow-sm text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500"
                    >
                      Go back home
                    </Link>
                </div>
              </div>
            }
          />
        </Routes>
      </AnimatePresence>
    </main>
    <Footer />
  </div>
```

```
  );
}

export default App;
```

**Home.jsx :**

```
import { motion, AnimatePresence } from 'framer-motion';
import { Link, useNavigate } from 'react-router-dom';
import { HeartPulse, Activity, Stethoscope, ClipboardCheck, ArrowRight, Heart } from 'lucide-react';

const features = [
  {
    icon: <ClipboardCheck className="h-8 w-8 text-blue-600" />,
    title: 'Comprehensive Analysis',
     description: 'Advanced AI evaluates multiple health parameters to provide accurate cardiovascular risk assessment.'
  },
  {
    icon: <Activity className="h-8 w-8 text-blue-500" />,
    title: 'Real-time Insights',
     description: 'Get immediate feedback on your heart health with detailed risk analysis and visual reports.'
  },
  {
    icon: <Stethoscope className="h-8 w-8 text-blue-400" />,
    title: 'Clinical Precision',
     description: 'Developed using the Sulianova CVD dataset and validated by cardiology experts.'
  }
];

const Home = () => {
  const navigate = useNavigate();

  const scrollToSection = (id) => {
    const element = document.getElementById(id);
    if (element) {
      element.scrollIntoView({ behavior: 'smooth' });
    }
  };

  return (
    <div className="min-h-screen bg-white">
      <AnimatePresence>
        {/* Hero Section */}
```

```
<div className="relative overflow-hidden pt-16 pb-12 bg-gradient-to-br from-blue-50 to-white">
<div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-12 md:py-20">
<div className="flex flex-col lg:flex-row items-center">
<div className="lg:w-1/2 text-center lg:text-left">
<motion.div
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}
className="inline-flex items-center justify-center px-4 py-2 rounded-full bg-blue-50 text-blue-700 text-sm font-medium mb-6"
>
<HeartPulse className="h-5 w-5 mr-2" />
<span>Cardiovascular Risk Assessment</span>
</motion.div>
<motion.h1
className="text-4xl tracking-tight font-extrabold text-gray-900 sm:text-5xl md:text-6xl"
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5, delay: 0.1 }}
>
<span className="block">Take Control of Your</span>
<span className="block text-blue-600">Heart Health Today</span>
</motion.h1>

<motion.p
className="mt-3 max-w-md mx-auto text-base text-gray-500 sm:text-lg md:mt-5 md:text-xl md:max-w-3xl"
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5, delay: 0.2 }}
>
Our AI-powered platform helps you understand your cardiovascular risk factors and provides personalized recommendations to improve your heart health.
</motion.p>
<motion.div
className="mt-8 flex flex-col sm:flex-row justify-center gap-4"
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5, delay: 0.3 }}
>
<Link
to="/predict"
className="w-full sm:w-auto inline-flex items-center justify-center px-
```

```
8 py-3 border border-transparent text-base font-medium rounded-md text-white bg-blue-600 hover:bg-blue-700 md:py-4 md:text-lg md:px-10 shadow-lg hover:shadow-xl transition-all duration-300 transform hover:-translate-y-1"
            >
              Get Started
              <ArrowRight className="ml-2 h-5 w-5" />
            </Link>

            <button
              onClick={() => scrollToSection('how-it-works')}
               className="w-full sm:w-auto inline-flex items-center justify-center px-8 py-3 border border-gray-200 text-base font-medium rounded-md text-gray-700 bg-white hover:bg-gray-50 md:py-4 md:text-lg md:px-10 shadow hover:shadow-md transition-all duration-300" >
              Learn More
            </button>
          </motion.div>
          </div>
           {/* Animated Border Image */}
          <div className="lg:w-1/2 mt-12 lg:mt-0 flex justify-center lg:justify
           end">
           <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{
              opacity: 1,
              y: 0,
            }}
            transition={{ duration: 0.5 }}
            className="relative w-full max-w-xl rounded-2xl p-1"
            style={{
                    background: 'linear-gradient(45deg, #3b82f6, #60a5fa, #93c5fd, #60a5fa, #3b82f6)',
              backgroundSize: '300% 300%',
              animation: 'gradient 8s ease infinite',
            }}
          >
            <div className="relative rounded-xl overflow-hidden">
              <img
                src="https://www.saralamemorialhospital.com/public/assets/admin/uploads/departments/Cardiologyimage1707214184.jpg"
                alt="Cardiology Department"
                className="w-full h-auto object-cover"
                style={{ minHeight: '400px' }}
              />
               <div className="absolute inset-0 bg-gradient-to-br from-blue-500/10 to-blue-700/20"></div>
```

```
            </div>
            {/* Animated border effect */}
            <motion.div
              className="absolute inset-0 rounded-2xl"
              style={{
                    background: 'linear-gradient(45deg, #3b82f6, #60a5fa, #93c5fd, #60a5fa, #3b82f6)',
                backgroundSize: '300% 300%',
                filter: 'blur(10px)',
                opacity: 0.7,
                zIndex: -1,
              }}
              animate={{
                backgroundPosition: ['0% 0%', '100% 100%'],
              }}
              transition={{
                duration: 8,
                repeat: Infinity,
                repeatType: 'reverse',
                ease: 'linear',
              }}
            />
          </motion.div>
        </div>
        <style jsx global>{`
          @keyframes gradient {
            0% { background-position: 0% 50%; }
            50% { background-position: 100% 50%; }
            100% { background-position: 0% 50%; }
          }
        `}</style>
      </div>
    </div>

    {/* Animated Heartbeat */}
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{
        opacity: 1,
        y: 0,
        scale: [1, 1.1, 1],
      }}
      transition={{
        duration: 0.5,
        delay: 0.4,
```

```
            scale: {
              repeat: Infinity,
              duration: 1.5,
              ease: "easeInOut"
            }
          }}
          className="mt-12 flex justify-center"
        >
          <div className="flex items-center justify-center space-x-4">
            <Heart className="h-10 w-10 text-red-500" />
              <span className="text-xl font-medium text-gray-700">Healthy Heart, Happy Life</span>
          </div>
        </motion.div>
      </div>

      {/* Features Section */}
      <div id="how-it-works" className="py-12 bg-white">
        <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
          <div className="lg:text-center">
              <h2 className="text-base text-blue-600 font-semibold tracking-wide uppercase">How It Works</h2>
            <p className="mt-2 text-3xl leading-8 font-extrabold tracking-tight text-gray-900 sm:text-4xl">
              Simple Steps to Better Heart Health
            </p>
            <p className="mt-4 max-w-2xl text-xl text-gray-500 lg:mx-auto">
              Our process is designed to be simple, fast, and informative.
            </p>
          </div>

          <div className="mt-10">
              <div className="space-y-10 md:space-y-0 md:grid md:grid-cols-3 md:gap-x-8 md:gap-y-10">
            {features.map((feature, index) => (
              <motion.div
                key={feature.title}
                initial={{ opacity: 0, y: 20 }}
                whileInView={{ opacity: 1, y: 0 }}
                viewport={{ once: true }}
                transition={{ duration: 0.5, delay: index * 0.1 }}
                className="relative"
              >
                <div className="absolute flex items-center justify-center h-12 w-12 rounded-md bg-blue-500 text-white">
```

```
{feature.icon}
                </div>
                <div className="ml-16">
                        <h3 className="text-lg leading-6 font-medium text-gray-900">{feature.title}</h3><p className="mt-2 text-base text-gray-500">{feature.description}</p>
                </div>
              </motion.div>
            ))}
          </div>
        </div>
      </div>
    </div>
    {/* CTA Section */}
    <div id="contact" className="bg-blue-700 text-white">
        <div className="max-w-2xl mx-auto text-center py-16 px-4 sm:py-20 sm:px-6 lg:px-8">
      <h2 className="text-3xl font-extrabold text-white sm:text-4xl">
            <span className="block">Ready to take control of your heart health?</span>
      </h2>
      <p className="mt-4 text-lg leading-6 text-blue-100">
        It only takes a few minutes to understand your risk and get personalized recommendations.
      </p>
      <motion.div
        whileHover={{ scale: 1.05 }}
        whileTap={{ scale: 0.95 }}
        className="mt-8 flex justify-center"
      > <Link
        to="/predict"
            className="inline-flex items-center px-8 py-4 border-2 border-white text-base font-bold rounded-lg text-blue-700 bg-white hover:bg-blue-50 transition-all duration-300"  >
          Start Your Assessment
          <ArrowRight className="ml-3 h-5 w-5" />
        </Link>
      </motion.div>
    </div>
  </div>
 </AnimatePresence>
</div>
);
};
export default Home;
```

**Predict.jsx :**

```
import React, { useState } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { ArrowLeft, ArrowRight, HeartPulse, User, Activity } from 'lucide-react';
import { toast } from 'react-hot-toast';
import { useNavigate } from 'react-router-dom';

const sections = [
  {
    title: 'Personal Info', icon: <User />,
    fields: [
      { name: 'age', label: 'Age', type: 'number', min: 18, max: 100 },
      { name: 'gender', label: 'Gender', type: 'select', options: [{ value: 1, label: 'Male' }, { value: 2, label: 'Female' }] },
      { name: 'height', label: 'Height (cm)', type: 'number', min: 100, max: 250 },
      { name: 'weight', label: 'Weight (kg)', type: 'number', min: 30, max: 200 }
    ]
  },
  {
    title: 'Health Metrics', icon: <Activity />,
    fields: [
      { name: 'ap_hi', label: 'Systolic BP', type: 'number', min: 90, max: 200 },
      { name: 'ap_lo', label: 'Diastolic BP', type: 'number', min: 60, max: 120 },
      { name: 'cholesterol', label: 'Cholesterol', type: 'select', options: [{ value: 1, label: 'Normal' }, { value: 2, label: 'Above Normal' }, { value: 3, label: 'High' }] },
      { name: 'glucose', label: 'Glucose', type: 'select', options: [{ value: 1, label: 'Normal' }, { value: 2, label: 'Above Normal' }, { value: 3, label: 'High' }] }
    ]
  },
  {
    title: 'Lifestyle', icon: <HeartPulse />,
    fields: [
      { name: 'smoke', label: 'Do you smoke?', type: 'radio', options: [{ value: 1, label: 'Yes' }, { value: 0, label: 'No' }] },
      { name: 'alco', label: 'Consume alcohol?', type: 'radio', options: [{ value: 1, label: 'Yes' }, { value: 0, label: 'No' }] },
      { name: 'active', label: 'Physically active?', type: 'radio', options: [{ value: 1, label: 'Yes' }, { value: 0, label: 'No' }] }
    ]
  }
];

const Predict = ({ setPredictionResult }) => {
  const [form, setForm] = useState({});
  const [step, setStep] = useState(0);
```

```
  const navigate = useNavigate();

  const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value });

  const handleSubmit = (e) => {
    e.preventDefault();
    const bmi = form.weight / ((form.height / 100) ** 2);
    const risk = Math.min(100, Math.round(
      (form.age * 0.4) + (bmi * 0.8) + (form.ap_hi > 140 ? 15 : 0) +
      (form.cholesterol - 1) * 10 + (form.glucose - 1) * 8 +
      (form.smoke * 20) + (form.alco * 10) - (form.active * 10)
    ));
    const data = { ...form, bmi: bmi.toFixed(1), riskScore: risk };
    setPredictionResult(data);
    navigate('/results', { state: data });
  };

  const renderField = (f) => (
    <div key={f.name}>
      <label className="block mb-1 font-medium text-gray-700">{f.label}</label>
      {f.type === 'select' ? (
        <select name={f.name} onChange={handleChange} className="border
rounded w-full p-2">
          <option value="">Select</option>
          {f.options.map(o => <option key={o.value}
value={o.value}>{o.label}</option>)}
        </select>
      ) : f.type === 'radio' ? (
        <div className="flex gap-4">{f.options.map(o => (
          <label key={o.value}><input type="radio" name={f.name} value={o.value}
onChange={handleChange}/> {o.label}</label>
        ))}</div>
      ) : (
        <input type={f.type} name={f.name} onChange={handleChange}
className="border rounded w-full p-2" />
      )}
    </div>
  );
  return (
    <div className="min-h-screen bg-gradient-to-b from-blue-50 to-white py-10">
      <div className="max-w-3xl mx-auto bg-white rounded-xl shadow-lg p-8">
        <h1 className="text-2xl font-bold text-center mb-6 text-blue-
700">Cardiovascular Risk Assessment</h1>

        {/* Progress */}
        <div className="flex justify-between mb-6">
```

```
        {sections.map((_, i) => (
          <div key={i} className={`h-2 w-full mx-1 rounded-full ${i <= step ? 'bg-blue-600' : 'bg-gray-200'}`}></div>
        ))}
      </div>
      <AnimatePresence mode="wait">
        <motion.form
          key={step}
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          exit={{ opacity: 0, y: -20 }}
          transition={{ duration: 0.3 }}
          onSubmit={handleSubmit}
          className="space-y-4"
        >
          <h2 className="text-xl font-semibold text-gray-800 flex items-center gap-2">
            {sections[step].icon} {sections[step].title}
          </h2>
          {sections[step].fields.map(renderField)}

          <div className="flex justify-between mt-6">
            <button type="button" disabled={step === 0} onClick={() => setStep(step - 1)}
              className={`px-4 py-2 rounded-md ${step === 0 ? 'bg-gray-200 text-gray-400' : 'bg-gray-100 hover:bg-gray-200'}`}>
              <ArrowLeft className="inline mr-1 h-4 w-4" /> Back
            </button>
            {step < sections.length - 1 ? (
              <button type="button" onClick={() => setStep(step + 1)}
                className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700">
                Next <ArrowRight className="inline ml-1 h-4 w-4" />
              </button>
            ) : (
              <button type="submit" className="px-4 py-2 bg-green-600 text-white rounded-md hover:bg-green-700">
                Get Risk Score
              </button>
            )}
          </div>
        </motion.form>
      </AnimatePresence>
    </div>
  </div>
 );
};
```

```
export default Predict;
```

**Navbar.jsx:**

```
import { Link, useLocation } from 'react-router-dom';
import { motion } from 'framer-motion';
import { HeartPulse, Menu, X } from 'lucide-react';
import { useState, useEffect } from 'react';

const Navbar = () => {
  const [isOpen, setIsOpen] = useState(false);
  const [scrolled, setScrolled] = useState(false);
  const location = useLocation();

  useEffect(() => {
    const onScroll = () => setScrolled(window.scrollY > 10);
    window.addEventListener('scroll', onScroll);
    return () => window.removeEventListener('scroll', onScroll);
  }, []);
  const links = [
    { name: 'Home', path: '/' },
    { name: 'Risk Assessment', path: '/predict' },
    { name: 'Results', path: '/results' }
  ];

  return (
    <motion.nav
      initial={{ y: -80 }}
      animate={{ y: 0 }}
      className={`fixed w-full z-50 py-2 transition-all duration-300 ${
        scrolled ? 'bg-white shadow-sm' : 'bg-white/95 backdrop-blur-sm'
      }`}
    >
      <div className="max-w-6xl mx-auto flex items-center justify-between px-4 h-
12">
        {/* Logo */}
        <Link to="/" className="flex items-center">
          <HeartPulse className="h-5 w-5 text-blue-600" />
          <span  className="ml-1.5  text-lg  font-bold  text-blue-600">CardioRisk
AI</span>
        </Link>
        {/* Desktop Menu */}
        <div className="hidden md:flex items-center gap-5">
          {links.map((l) => (
            <Link
```

```
            key={l.name}
            to={l.path}
            className={`text-sm font-medium ${
              location.pathname === l.path
                ? 'text-blue-600'
                : 'text-gray-700 hover:text-blue-600'
            }`}
          >
            {l.name}
          </Link>
        ))}
        <Link
          to="/predict"
          className="px-3 py-1.5 bg-blue-600 text-white rounded-md text-sm hover:bg-blue-700"
        >
          Get Started
        </Link>
      </div>

      {/* Mobile Toggle */}
      <button
        className="md:hidden text-gray-700"
        onClick={() => setIsOpen(!isOpen)}
      >
        {isOpen ? <X className="h-6 w-6" /> : <Menu className="h-6 w-6" />}
      </button>
    </div>

    {/* Mobile Menu */}
    {isOpen && (
      <motion.div
        initial={{ opacity: 0, height: 0 }}
        animate={{ opacity: 1, height: 'auto' }}
        className="md:hidden bg-white shadow-md"
      >
        {links.map((l) => (
          <Link
            key={l.name}
            to={l.path}
            onClick={() => setIsOpen(false)}
            className={`block px-4 py-2 text-sm ${
              location.pathname === l.path
                ? 'bg-blue-50 text-blue-600'
                : 'text-gray-700 hover:bg-gray-100'
```

```
            }`}
          >
            {l.name}
          </Link>
        ))}
        <Link
          to="/predict"
          onClick={() => setIsOpen(false)}
          className="block text-center m-3 py-2 bg-blue-600 text-white rounded-md
hover:bg-blue-700"
        >
          Get Started
        </Link>
      </motion.div>
    )}
  </motion.nav>
 );
};

export default Navbar;
```

**RiskMeter.jsx:**

```
import { motion, useAnimation } from 'framer-motion';
import { useEffect, useRef, useState } from 'react';
import { Info } from 'lucide-react';

const RiskMeter = ({
 percentage = 50,
 size = 200,
 lineWidth = 20,
 label = 'Cardiovascular Risk',
 showInfo = true,
 className = '',
}) => {
 const controls = useAnimation();
 const [isHovered, setIsHovered] = useState(false);
 const [showTooltip, setShowTooltip] = useState(false);
 const tooltipRef = useRef(null);

 // Calculate the radius and circumference for the gauge
 const radius = (size - lineWidth) / 2;
 const circumference = radius * 2 * Math.PI;
 const offset = circumference - (percentage / 100) * circumference;

 // Determine risk level and color - aligned with getRiskDetails in Results.jsx
```

```
 const getRiskLevel = (percent) => {
  // Ensure high risk is always red
    if (percent >= 75) return { level: 'High Risk', color: '#EF4444', description: 'Significantly elevated risk of cardiovascular disease. Medical consultation strongly recommended.' };
    if (percent >= 60) return { level: 'Moderate-High Risk', color: '#F97316', description: 'Elevated risk. Consider lifestyle changes and consult with a healthcare provider.' };
   if (percent >= 40) return { level: 'Moderate Risk', color: '#F59E0B', description: 'Moderate risk. Lifestyle changes and regular monitoring are advised.' };
    if (percent >= 20) return { level: 'Low-Moderate Risk', color: '#84CC16', description: 'Slightly elevated risk. Maintain healthy habits and monitor your health.' };
   return { level: 'Low Risk', color: '#22C55E', description: 'Low risk. Continue your healthy lifestyle.' };
 };

 const risk = getRiskLevel(percentage);
  // Animate the gauge when percentage changes
 useEffect(() => {
  controls.start({
   strokeDashoffset: offset,
   transition: { duration: 1.5, ease: 'easeInOut' },
  });
 }, [percentage, offset, controls]);

 // Handle tooltip visibility
 useEffect(() => {
  if (isHovered) {
   const timer = setTimeout(() => setShowTooltip(true), 300);
   return () => clearTimeout(timer);
  } else {
   setShowTooltip(false);
  }
 }, [isHovered]);
  // Close tooltip when clicking outside
 useEffect(() => {
  const handleClickOutside = (event) => {
   if (tooltipRef.current && !tooltipRef.current.contains(event.target)) {
    setShowTooltip(false);
   }
  };

  document.addEventListener('mousedown', handleClickOutside);
  return () => document.removeEventListener('mousedown', handleClickOutside);
 }, []);
```

```
return (
  <div className={`relative flex flex-col items-center ${className}`}>
    {/* Gauge */}
    <div className="relative" style={{ width: size, height: size }}>
      {/* Background circle */}
      <svg width={size} height={size} className="transform -rotate-90">
        <circle
          cx={size / 2}
          cy={size / 2}
          r={radius}
          fill="none"
          stroke="#E5E7EB"
          strokeWidth={lineWidth}
          strokeLinecap="round"
        />

        {/* Animated progress circle */}
        <motion.circle
          cx={size / 2}
          cy={size / 2}
          r={radius}
          fill="none"
          stroke={risk.color}
          strokeWidth={lineWidth}
          strokeLinecap="round"
          strokeDasharray={circumference}
          initial={{ strokeDashoffset: circumference }}
          animate={controls}
          style={{ filter: 'drop-shadow(0 0 8px rgba(59, 130, 246, 0.4))' }}
        />
      </svg>

      {/* Center text */}
      <div className="absolute inset-0 flex flex-col items-center justify-center">
        <span className="text-4xl font-bold" style={{ color: risk.color }}>
          {Math.round(percentage)}%
        </span>
        <span className="text-sm text-gray-500 mt-1">
          {risk.level}
        </span>
      </div>

      {/* Risk indicator dot */}
      <div
```

```
        className="absolute top-0 left-1/2 w-3 h-3 rounded-full -mt-1.5 -ml-1.5"
        style={{
          backgroundColor: risk.color,
          transform: `rotate(${180 + (percentage / 100) * 180}deg)`,
          transformOrigin: `center ${size / 2 + lineWidth / 2}px`,
          transition: 'transform 1.5s ease-in-out, background-color 0.5s ease',
        }}
      />
    </div>
      {/* Label and info */}
    <div className="mt-4 text-center">
      <div className="flex items-center justify-center space-x-2">
        <h3 className="text-lg font-medium text-gray-800">{label}</h3>
        {showInfo && (
          <div className="relative">
            <button
              onMouseEnter={() => setIsHovered(true)}
              onMouseLeave={() => setIsHovered(false)}
              onClick={() => setShowTooltip(!showTooltip)}
              className="text-gray-400 hover:text-blue-500 transition-colors"
              aria-label="Risk information"
            >
              <Info size={16} />
            </button>
            {/* Tooltip */}
            {showTooltip && (
              <div
                ref={tooltipRef}
                 className="absolute z-10 w-64 p-3 mt-2 text-sm text-left text-gray-700
bg-white border border-gray-200 rounded-lg shadow-lg -left-32"
                onMouseEnter={() => setIsHovered(true)}
                onMouseLeave={() => setIsHovered(false)}
              >

<p className="font-medium mb-1">{risk.level} Risk ({percentage}%)</p>
                 <p className="text-gray-600 text-xs">{risk.description} based on the
provided health metrics.</p>
              <div className="mt-2 pt-2 border-t border-gray-100">
                <div className="flex justify-between text-xs text-gray-500">
                  <span>0%</span>
                  <span>50%</span>
                  <span>100%</span>
                </div>
                    <div className="h-2 w-full bg-gradient-to-r from-green-500 via-
yellow-500 to-red-500 rounded-full mt-1"></div>
```

```
            </div>
          </div>
        )}
      </div>
    )}
  </div>

  {/* Risk description */}
  <p className="mt-2 text-sm text-gray-600 max-w-xs">
   {risk.description}
  </p>
 </div>

 {/* Risk indicators with improved drag visualization */}
 <div className="w-full mt-6 px-2">
  <div className="flex justify-between text-xs text-gray-500 mb-1 px-1">
   <span>0%</span>
   <span>50%</span>
   <span>100%</span>
  </div>
  <div className="relative h-3 w-full">
    <div className="absolute h-2 w-full bg-gradient-to-r from-green-500 via-yellow-500 to-red-500 rounded-full top-1/2 -translate-y-1/2"></div>
   <motion.div
    className="absolute h-4 w-1 bg-white border-2 border-gray-700 rounded-full -top-0.5 -ml-0.5 shadow-md z-10"
    style={{
     left: `${percentage}%`,
     backgroundColor: risk.color,
     borderColor: 'white',
     boxShadow: '0 0 0 2px white, 0 0 0 3px ' + risk.color,
    }}
    initial={{ x: 0 }}
    animate={{ x: 0 }}
    transition={{ duration: 1.5, ease: 'easeInOut' }}
   />
  </div>
  <div className="flex justify-between text-xs text-gray-500 mt-1 px-1">
   <span className="text-green-600 font-medium">Low</span>
   <span className="text-yellow-600 font-medium">Medium</span>
   <span className="text-red-600 font-medium">High</span>
  </div>
 </div>
</div>
);
```

```
};

export default RiskMeter;
```

**Main.jsx:**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import './index.css';
import { Toaster } from 'react-hot-toast';
ReactDOM.createRoot(document.getElementById('root')).render(
<React.StrictMode>
<BrowserRouter>
<App />
<Toaster
position="top-center"
toastOptions={{
style: {
background: '#1E40AF',
color: '#fff',
},
success: {
duration: 3000,
iconTheme: {
primary: '#10B981',
secondary: '#fff',
},
},
error: {
duration: 4000,
iconTheme: {
primary: '#EF4444',
secondary: '#fff',
},
},
}}
/>
</BrowserRouter>
</React.StrictMode>
);
```

**App.css:**

```
/* Additional custom styles if needed */
#root {
  width: 100%;
  margin: 0;
  padding: 0;
}

/* Smooth transitions */
* {
  transition: background-color 0.2s ease, color 0.2s ease;
}

/* Custom scrollbar */
::-webkit-scrollbar {
  width: 8px;
}

::-webkit-scrollbar-track {
  background: #f1f1f1;
}

::-webkit-scrollbar-thumb {
  background: #0ea5e9;
  border-radius: 4px;
}
::-webkit-scrollbar-thumb:hover {
  background: #0284c7;
}
```

# 7. TESTING

Testing is a crucial phase in the development of the cardiovascular disease prediction system, ensuring that the implemented models and the overall application perform accurately, reliably, and efficiently. The primary objective of testing is to identify and correct potential errors, validate the performance of each module, and confirm that the system meets the expected functional and non-functional requirements. Both frontend and backend components were rigorously tested to verify data flow, model inference accuracy, and user interface responsiveness. Unit testing, integration testing, and system testing were performed to ensure smooth communication between modules, while model validation confirmed prediction reliability using real-time cardiovascular data inputs. This comprehensive testing process guarantees that the proposed system delivers consistent and clinically relevant predictions for cardiovascular risk assessment.

## 7.1 UNIT TESTING

Unit testing is a critical stage in the development of the cardiovascular risk prediction system to ensure the accuracy, stability, and efficiency of each module within the pipeline. It focuses on validating the functionality of individual components, including data preprocessing, feature engineering, deep learning models, and explainability integration, before system-wide testing. The main objective is to detect logical or computational errors early and verify that each component performs as intended.

**1. Data Preprocessing Module Testing:**

This unit test validates the accuracy of data cleaning and transformation steps applied to the Sulianova dataset. It ensures that missing values are handled correctly, categorical variables are encoded using label encoding, and features are standardized through Z-score normalization. The test confirms that all input features fall within the expected range and that no data leakage occurs between training and testing sets.

**2. Feature Extraction and Sampling Testing:**

Unit testing of the feature extraction and resampling module ensures that the hybrid

resampling techniques (SMOTE, ADASYN, or random oversampling) effectively balance the dataset without introducing bias. The extracted statistical and domain-specific features are checked for correctness, consistency, and shape compatibility with the model input format. Feature importance values generated through SHAP are also verified for interpretability and reproducibility.

**3. Model-Specific Testing (TabNet, MLP, CNN-BiLSTM, Attention-LSTM, 1D CNN):**

Each deep learning model is tested independently to confirm proper layer configuration, activation flow, and expected output dimensions. Unit tests validate that the models can overfit small batches, demonstrating correct learning behaviour. Loss reduction and metric computation (accuracy, F1-score, and AUC) are verified for correctness during both training and validation phases.

**4. SHAP Explainability Testing:**

This test ensures that SHAP value generation functions correctly with all trained models and produces stable feature attributions. It verifies that the top contributing features align with medical relevance, such as cholesterol, age, blood pressure, and BMI. Additionally, visualization functions (force plots and summary plots) are validated to confirm accurate rendering and interpretation.

**5. Performance and Error Handling Testing:**

Edge cases, such as missing inputs, out-of-range feature values, and invalid data types, are tested to ensure robust error handling. The system's performance is evaluated under varying data sizes to confirm scalability and response time. The prediction output is verified to fall within valid probability bounds, ensuring reliability during real-time inference.

## 7.2 INTEGRATION TESTING

Integration testing ensures that all modules of the cardiovascular disease prediction system — including data preprocessing, deep learning models, explainability components, and the full-stack web interface — function cohesively as a unified system. This phase validates the flow of data from user input in the frontend to prediction generation in the backend and result visualization, confirming that the integrated components perform accurately and efficiently without interface or data transfer errors.

**1. Backend Integration (Colab + Flask):**

Integration testing at the backend level focuses on verifying the seamless communication between the model inference pipeline and the Flask API. The trained deep learning models (TabNet, CNN-BiLSTM, MLP, Attention-LSTM, and 1D CNN) are loaded into the Flask environment, where test cases ensure correct model initialization, input formatting, and prediction output consistency. The API endpoints are tested using both valid and invalid payloads to confirm proper data validation, preprocessing consistency with the Colab-trained pipeline, and appropriate JSON responses for predictions. SHAP explainability outputs are also integrated and verified to ensure feature importance visualizations are correctly generated and returned from the API.

**2. Frontend–Backend Communication Testing:**

This stage ensures smooth interaction between the React-based frontend and the Flask API. Using the Fetch and Axios request methods, integration tests confirm that user inputs collected through React forms (such as age, blood pressure, cholesterol, and glucose levels) are correctly serialized and transmitted to the backend API endpoint (/predict). The returned prediction results, including the risk probability and interpretability scores, are validated for correct parsing and display. The React state management is tested to confirm that the system dynamically updates result components such as the RiskMeter and SHAP explanation charts without page reloads or data corruption.

**3. Model Ensemble Integration Testing:**

The integration between multiple deep learning models and the ensemble evaluation module is tested to ensure consistent output aggregation. Predictions from all five models are combined to generate an ensemble probability score, and unit-weighted or performance-weighted averaging is validated for mathematical correctness. The pipeline ensures that each model's output is aligned in dimension and data type before fusion, maintaining prediction stability across models. The integrated output is compared with standalone model results to confirm improved accuracy and reduced variance, consistent with the reference methodology.

**4. Data Flow and Preprocessing Integration:**

Integration testing confirms that the data preprocessing pipeline used in Colab for training — including label encoding, Z-score normalization, and hybrid resampling — is fully replicated during inference through the Flask backend. This guarantees that the input received from the React frontend undergoes the same normalization

and feature transformation before model prediction. This consistency prevents model drift and ensures uniform performance between the development and deployed environments.

**5. End-to-End System Testing (User to Prediction):**

Finally, end-to-end integration testing is performed to validate the complete workflow, starting from user input on the React interface, through the Flask backend, and ending with result visualization. The testing confirms that every user query generates a valid prediction, risk category, and SHAP-based interpretability output. Additional checks verify the system's response time, stability under concurrent requests, and robustness to malformed inputs. This ensures that the deployed system maintains clinical-grade reliability, transparency, and scalability for real-world cardiovascular risk assessment.

## 7.3 SYSTEM TESTING

System testing evaluates the complete cardiovascular disease prediction platform as an integrated and functional system, ensuring that all components—data preprocessing, model inference, SHAP explainability, Flask API, and React interface—work together seamlessly to deliver accurate, efficient, and user-friendly predictions. The goal of this phase is to verify that the system fulfils both functional and non-functional requirements, providing consistent results under real-world usage conditions.

**1. Functional Testing:**

Functional testing validates that each core functionality of the system performs according to design specifications. The system is tested for correct input acceptance through the React-based form, accurate preprocessing via Flask, and valid model predictions returned by the deep learning ensemble. Test cases confirm the proper mapping of features such as age, blood pressure, cholesterol, glucose, and physical activity to the backend processing pipeline. Additionally, SHAP-based visual explanations are checked for accurate rendering, ensuring that feature importance corresponds to the input attributes influencing the cardiovascular risk prediction.

**2. Performance Testing:**

This testing phase measures the response time, scalability, and throughput of the deployed application. Using simulated API requests, the Flask backend is tested to ensure that prediction responses are generated within optimal time limits, even

under high user load. Model loading time, inference speed, and visualization latency are measured to ensure that the system maintains performance consistency. The GPU-accelerated inference setup used in Colab is evaluated against CPU-based local execution to ensure scalable deployment options.

**3. Usability Testing:**

Usability testing focuses on evaluating the user experience of the React-based web interface. Participants are asked to interact with the platform—entering data, submitting predictions, and interpreting visual risk outputs. Feedback is collected to assess clarity, accessibility, and responsiveness. The navigation flow between the Home, Risk Assessment, and Results pages is tested to ensure that users can easily understand their cardiovascular risk levels through the interactive RiskMeter and interpret SHAP charts for better transparency.

**4. Reliability and Stability Testing:**

Reliability testing ensures that the system remains stable and continues functioning correctly over multiple prediction cycles. The backend is tested for consistent performance under continuous requests, verifying that model weights remain intact and the Flask service does not crash due to memory leaks or concurrent access. Error-handling routines are validated to ensure appropriate messages are displayed for missing or invalid inputs, preventing application failure and ensuring robustness during runtime.

**5. Compatibility and Deployment Testing:**

This phase validates that the web application performs consistently across multiple browsers (Chrome, Edge, Firefox) and devices (desktop, tablet, and mobile). The integration between the React frontend, Flask backend, and model API endpoints is tested in a deployed environment using a local or cloud-based setup. The system is verified to function seamlessly when migrated from Colab training to a live deployment, confirming that all environment dependencies, preprocessing modules, and model checkpoints are correctly configured.

**6. End-to-End Validation:**

Finally, end-to-end system validation ensures that the entire workflow—from data entry, model inference to visual feedback—is fully operational. The predicted cardiovascular risk levels are cross-checked against the benchmark metrics obtained during model evaluation (accuracy, F1-score, and AUC). The system demonstrates high reliability, user interpretability, and clinical applicability.

# 8. RESULT ANALYSIS

The proposed SHAP-guided hybrid deep learning pipeline was evaluated across five architectures—TabNet, CNN+BiLSTM, MLP, Attention-LSTM, and 1D-CNN—using the Sulianova cardiovascular disease dataset. Each model was trained and validated using a stratified 80–20 data split, along with 5-fold cross-validation to ensure generalization. The evaluation metrics considered include Accuracy, F1-Score, and Area Under the ROC Curve (AUC), which collectively measure predictive correctness, class-balance sensitivity, and discrimination ability.

### A. Model Performance Comparison

The comparative results of all models are presented in Table II. Among them, TabNet achieved the highest classification accuracy of 96.19%, with an F1-Score of 0.9627 and an AUC of 0.9916, demonstrating strong feature-selection capabilities and reliable discrimination between high- and low-risk patients. The MLP and CNN+BiLSTM models followed closely with accuracies of 96.08% and 95.71%, respectively. Meanwhile, Attention-LSTM attained 95.65%, and 1D-CNN achieved 94.66%, providing efficient computation at the expense of slight accuracy trade-offs.

| | **Model** | **Accuracy** | **F1-Score** | **AUC** |
|---|---|---|---|---|
| 0 | TabNet | 0.961921 | 0.962723 | 0.991636 |
| 1 | MLP | 0.960886 | 0.961326 | 0.992625 |
| 2 | CNN+BiLSTM | 0.957161 | 0.958021 | 0.991042 |
| 3 | Attention-LSTM | 0.956540 | 0.957002 | 0.990492 |
| 4 | 1D CNN | 0.946606 | 0.947794 | 0.987029 |

**Table 2. Performance comparison of all deep learning models**

### B. Training and Validation Analysis

The training and validation accuracy curves for TabNet (Fig. 6) demonstrate smooth convergence without significant overfitting, signifying that the hybrid resampling and early-stopping strategies effectively stabilized learning. Similarly, the CNN+BiLSTM and MLP models exhibited balanced learning curves with steady improvement across epochs, validating the robustness of the preprocessing and

optimization framework.

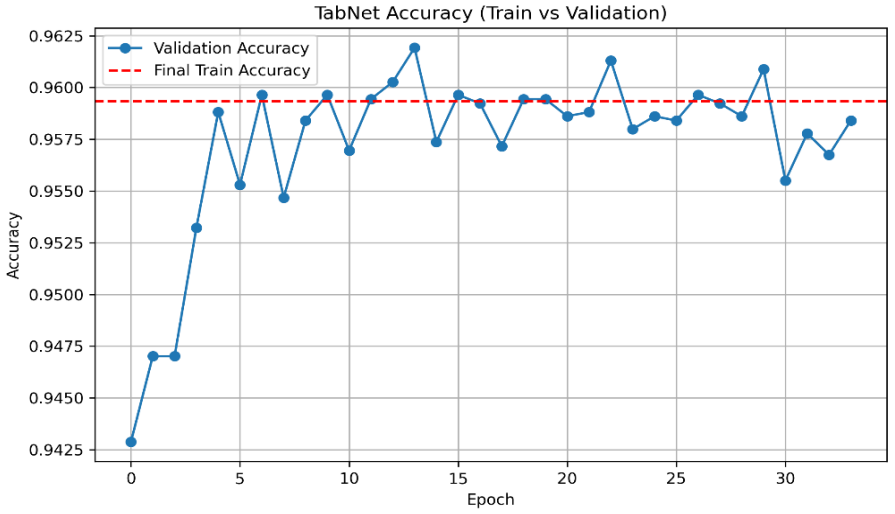


**Fig. 8.1. Training and validation accuracy of the TabNet model**

### C. Confusion Matrix Evaluation

The confusion matrix for the TabNet model (Fig. 7) confirms exceptional class discrimination with minimal false positives and false negatives. The model effectively differentiates between positive and negative cardiovascular risk cases, highlighting high sensitivity and specificity. Similar matrices for the MLP and CNN+BiLSTM models show consistent reliability, reinforcing the model stability across multiple architectures and folds.

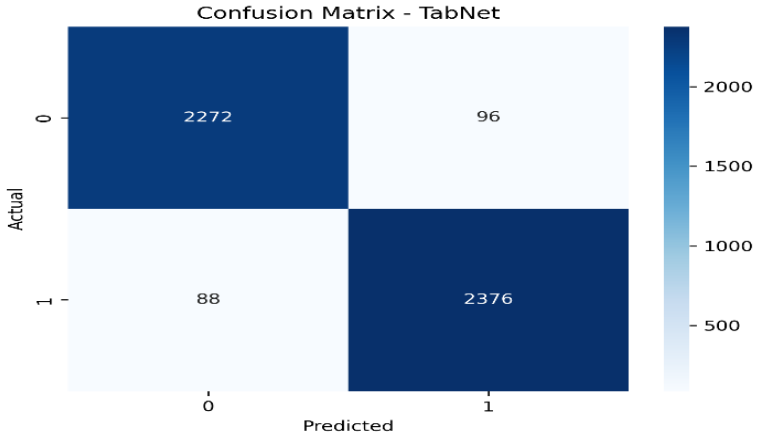


**Fig. 8.2. Confusion matrix of the TabNet model**

### D. Model Explainability Using SHAP

To ensure interpretability, SHAP (Shapley Additive Explanations) was applied to TabNet, MLP, and CNN+BiLSTM. The SHAP summary plots (Figs. 3–5) identify systolic blood pressure (ap_hi), diastolic pressure (ap_lo), age, and cholesterol as

the most influential predictors. These variables correspond strongly with established clinical knowledge, confirming that the deep learning models not only achieve statistical accuracy but also align with medical reasoning. This interpretability enhances clinical trust and transparency, making the models suitable for real-world decision support.

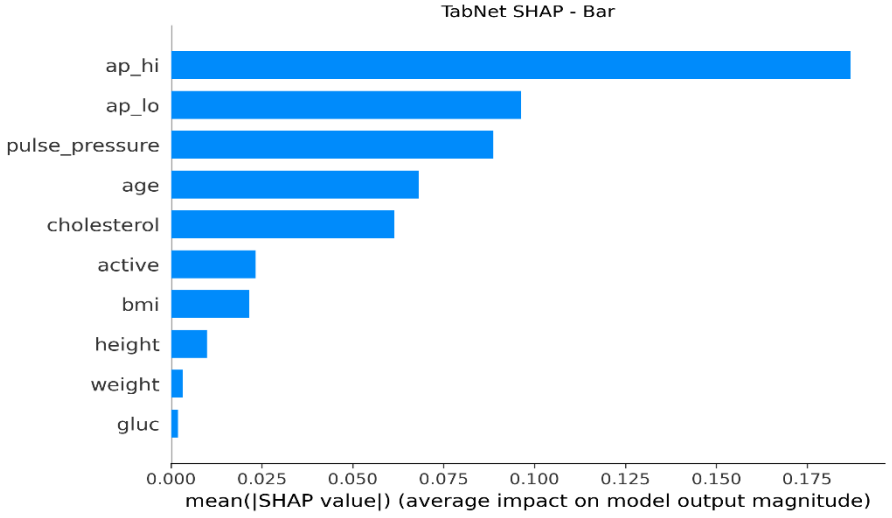


**Fig. 8.3. Bar plot of TabNet SHAP**

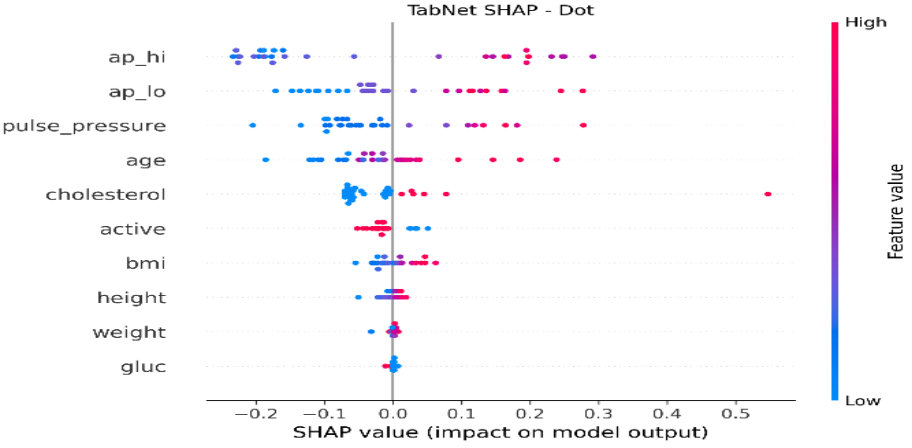


**Fig. 8.4. Dot plot of TabNet SHAP**

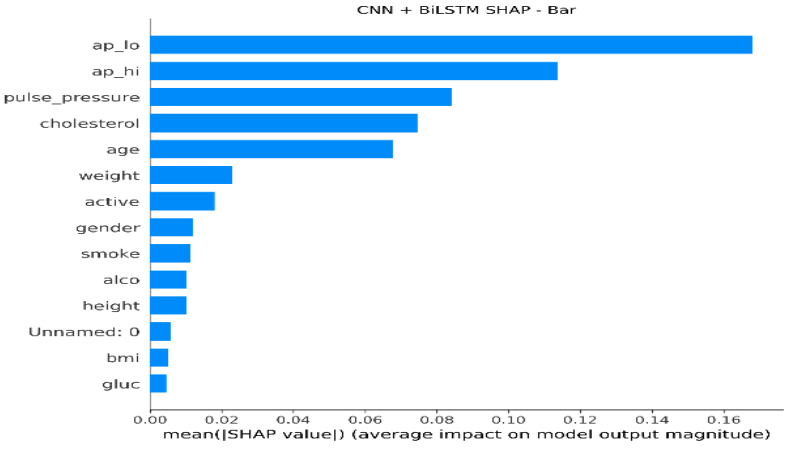


**Fig. 8.5. Bar plot of CNN+BiLSTM SHAP**

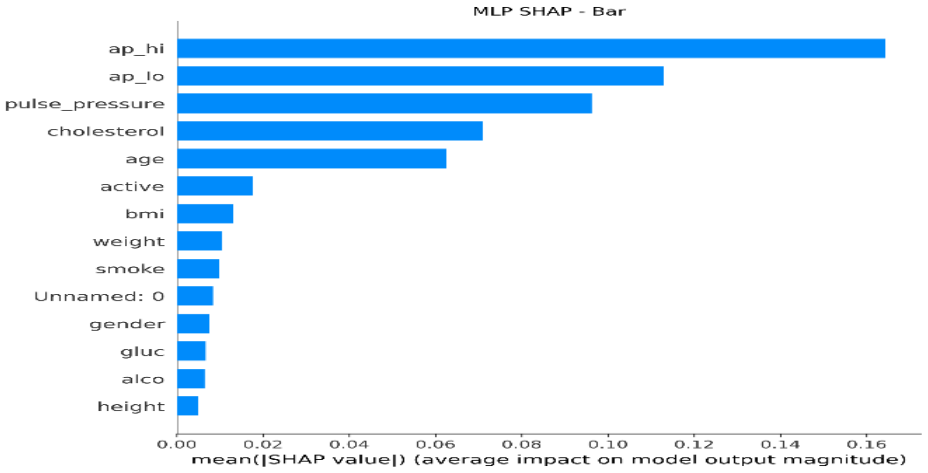


**Fig. 8.6. Bar plot of MLP SHAP**

## E. Comparative Discussion

Fig. 9 illustrates the comparative accuracy, F1-score, and AUC across all five models. The unified analysis reveals that while TabNet leads in all metrics, the MLP and CNN+BiLSTM models provide competitive performance with faster convergence and lower computational cost. The 1D-CNN, though slightly less accurate, offers the fastest training time (10.8 minutes), making it advantageous for rapid deployment scenarios. Attention-LSTM balances interpretability and sequential awareness, showing strong adaptability to temporal dependencies in tabular features.

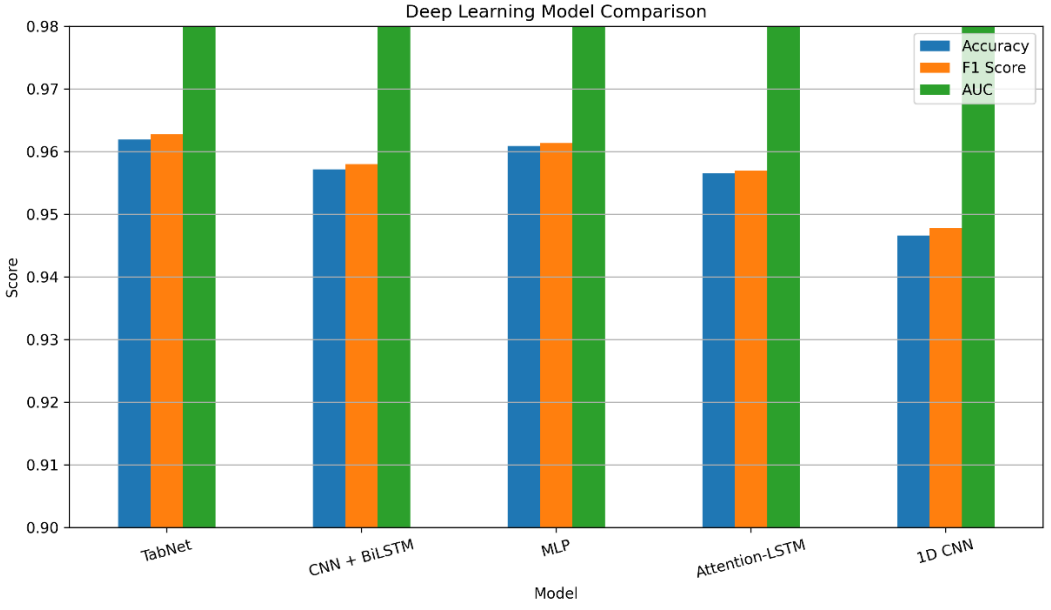


**Fig. 8.7. A comparison of the five deep learning models**

# 9. OUTPUT SCREENS

The proposed CardioRisk AI web application provides an integrated platform combining React.js frontend and a Flask backend for cardiovascular disease risk prediction. The Home screen serves as an entry point, featuring animated ECG visuals, system overview, and navigation to the prediction module. The Risk Assessment screen allows users to input clinical and lifestyle parameters such as age, gender, blood pressure, cholesterol, glucose, BMI, smoking, and physical activity through an interactive multi-step form with real-time validation. Upon submission, the React interface communicates with the Flask API, which preprocesses the input using stored scaler and feature selector objects, loads the trained TabNet deep learning model, and returns a JSON response containing the predicted risk level, probability score, and personalized recommendations. The Results screen visually displays the outcome with a dynamic risk meter gauge, color-coded indicators (green for low, orange for medium, and red for high risk), and a summary of the entered parameters. Overall, the output screens demonstrate a seamless integration of machine learning inference, medical interpretability, and user-friendly visualization, ensuring accurate, transparent, and efficient cardiovascular risk assessment.

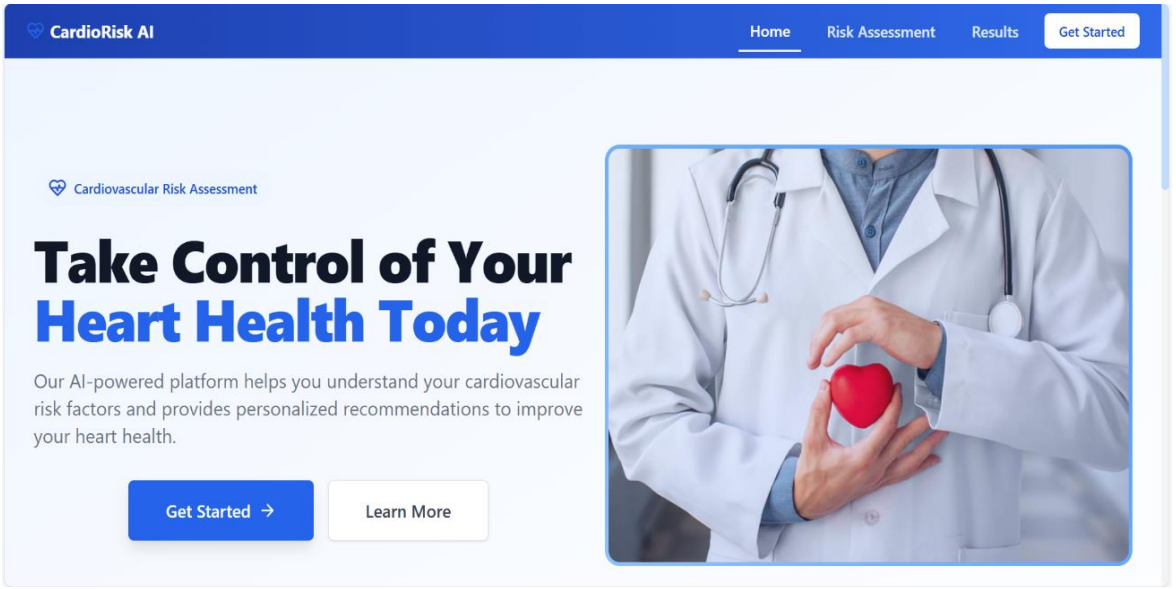


**Fig. 9.1. Home page**

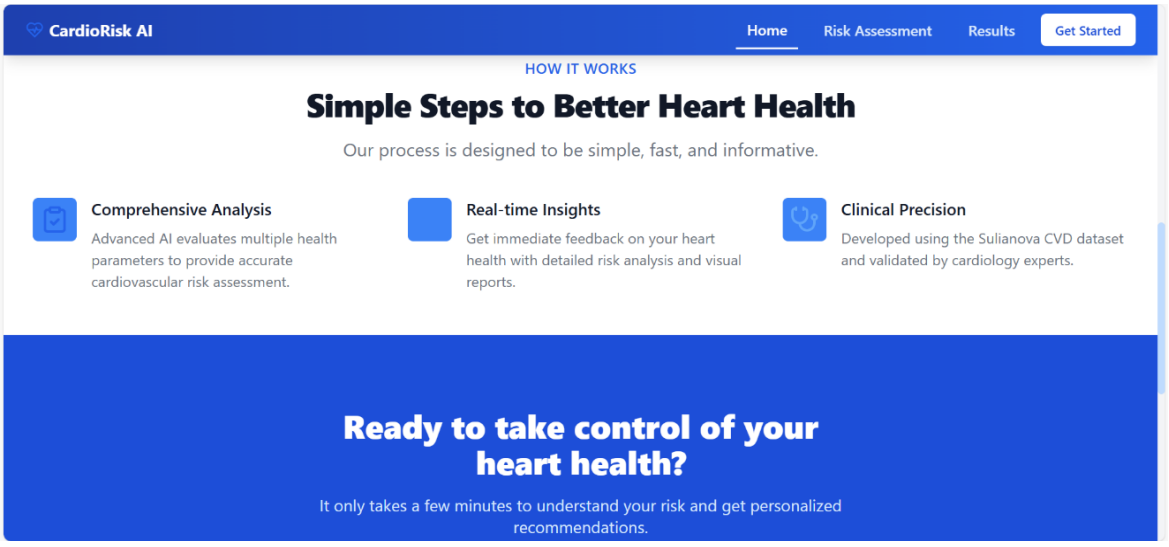


**Fig. 9.2. How it works**

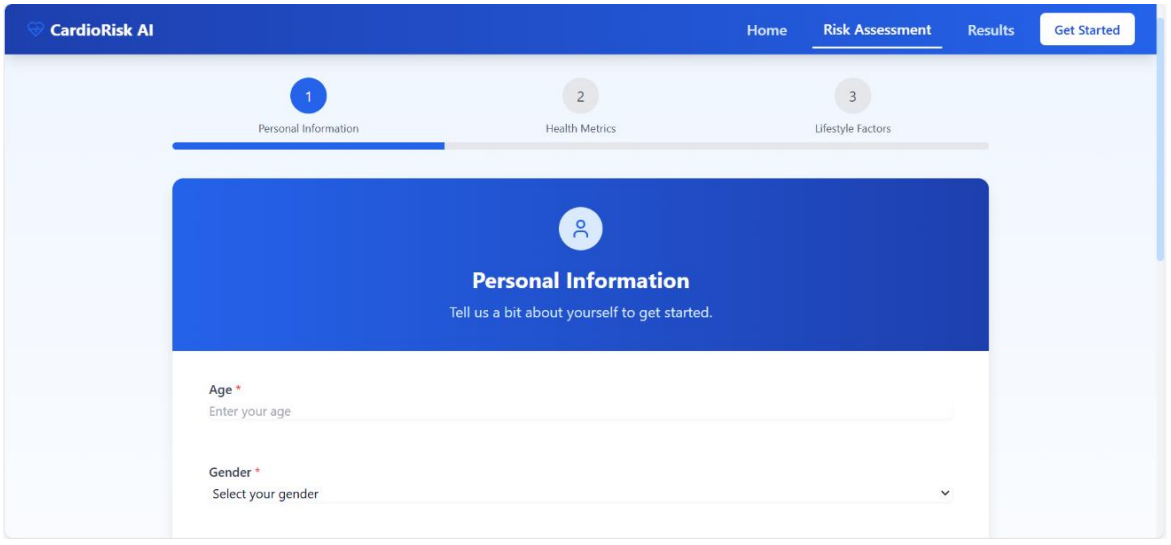


**Fig. 9.3. Risk assessment**

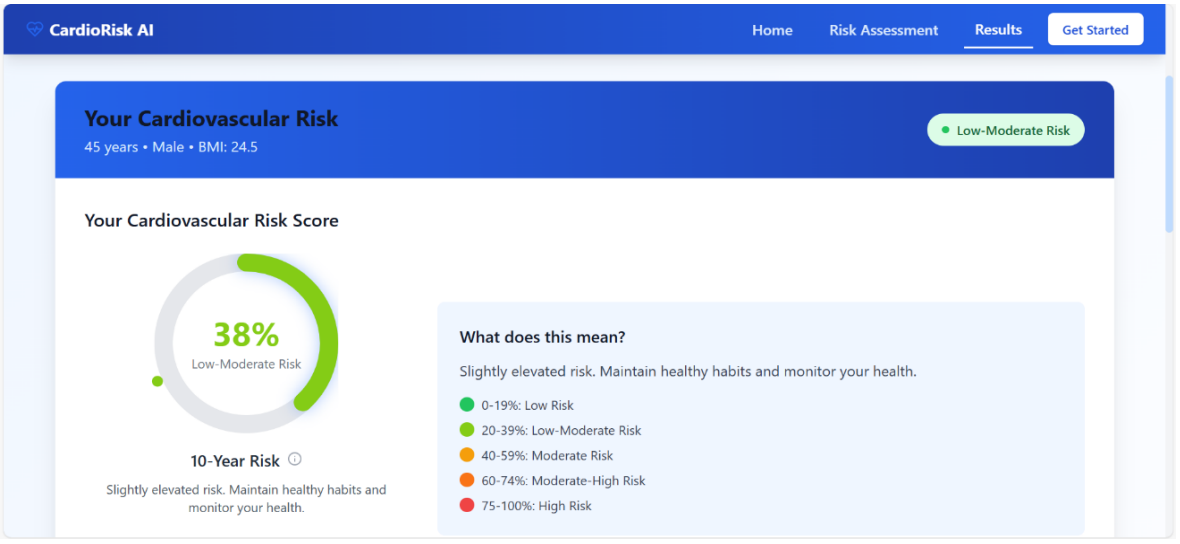


**Fig. 9.4. Results**

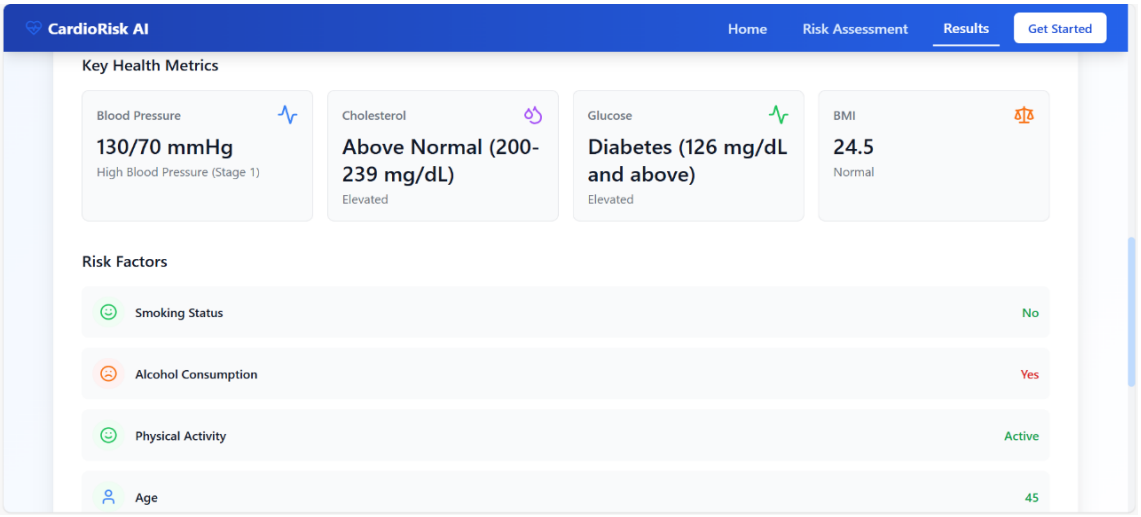


**Fig. 9.5. Key health metrics**

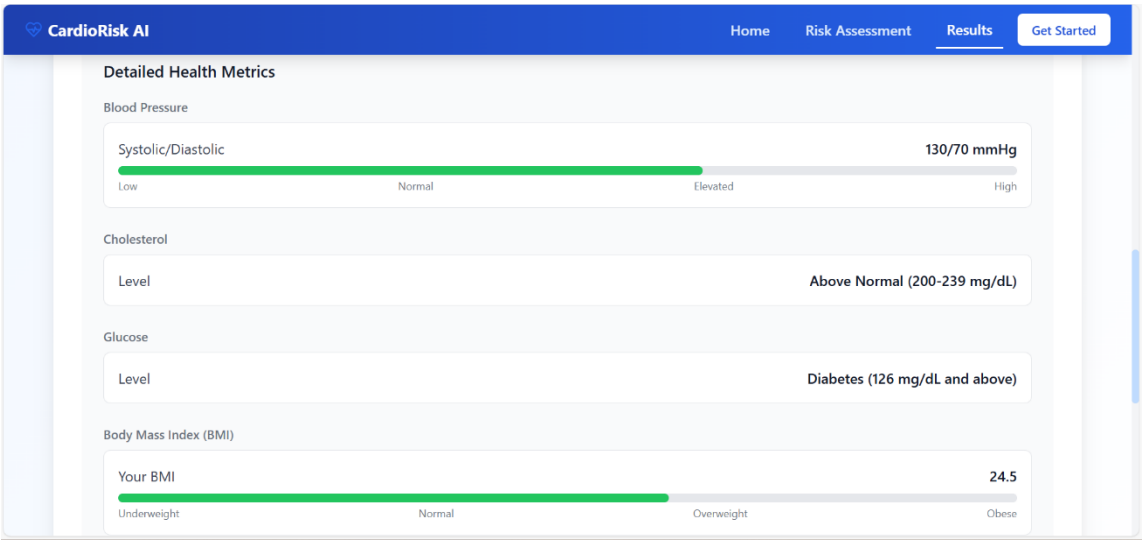


**Fig. 9.6. Detailed health metrics**

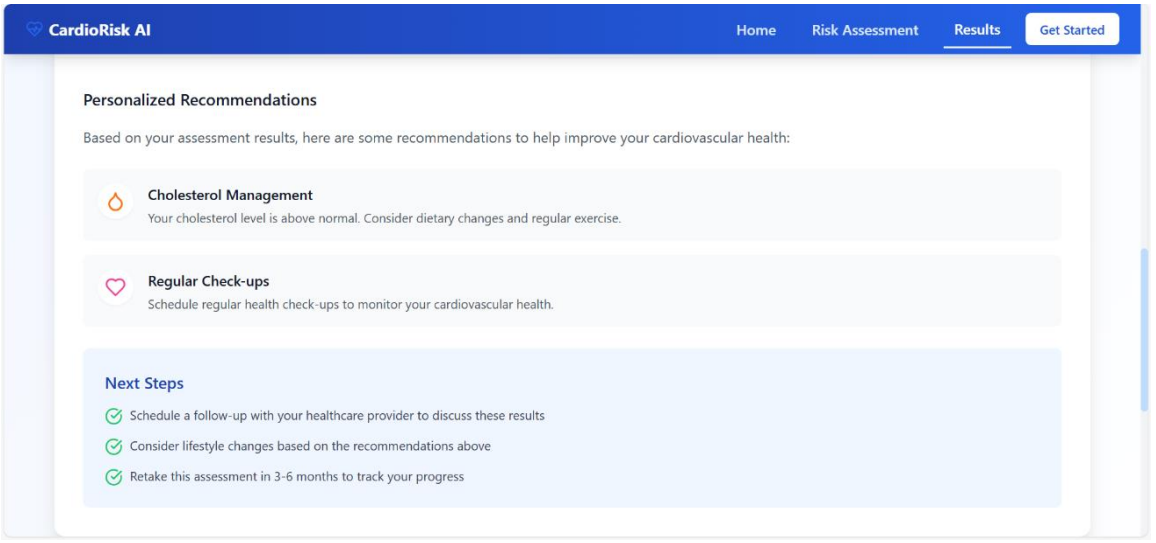


**Fig. 9.7. Personalized recommendations**

# 10. CONCLUSION

This research successfully developed a SHAP-guided deep learning framework for cardiovascular disease (CVD) risk prediction using the Sulianova dataset. The proposed system integrated advanced preprocessing techniques, feature selection, and hybrid deep models such as TabNet, CNN–BiLSTM, MLP, Attention-LSTM, and 1D-CNN, achieving high classification accuracy and interpretability. Each model was rigorously evaluated using metrics like accuracy, F1-score, and AUC to ensure reliable performance. The inclusion of SHAP explainability enhanced model transparency, providing both global and local insights into how clinical parameters influence CVD risk.

Among all models, the TabNet and CNN–BiLSTM architectures demonstrated superior accuracy, robustness, and generalization capability across folds. The comparative analysis confirmed that feature-rich deep networks, combined with proper normalization and resampling strategies, significantly reduce bias and improve detection reliability. The application of Z-score normalization, label encoding, and hybrid resampling helped balance the dataset, ensuring consistent results across training and testing phases. The use of SHAP visualization further strengthened clinical trust by showing the most influential features in the prediction process, aligning with real-world medical understanding.

To bridge the gap between research and real-world usability, a web-based application was implemented using Flask (backend) and React.js (frontend) for end-to-end deployment. The application allows users to input personal and clinical parameters to receive an instant cardiovascular risk assessment. The Flask backend handles model inference and data preprocessing, while the React interface provides an interactive and responsive user experience.

In conclusion, the proposed SHAP-guided deep learning framework demonstrates a highly effective and reliable approach for cardiovascular disease prediction. The integration of multiple deep learning models and explainable AI enhances both performance and interpretability. The results confirm that the system delivers strong predictive accuracy and clinical relevance, validating its potential as a dependable tool for cardiovascular risk stratification.

# 11. FUTURE SCOPE

Future research can focus on expanding the dataset by incorporating more diverse patient populations and real-time clinical data from multiple hospitals and regions. This would enhance the model's generalizability, allowing it to adapt to varying genetic, lifestyle, and environmental factors that influence cardiovascular risk. By including larger and more balanced datasets, the system could address potential data bias and improve prediction reliability. Additionally, integrating longitudinal data would enable the model to analyze disease progression over time. This approach would ultimately lead to more accurate and personalized cardiovascular disease risk assessments for global applicability.

The integration of wearable device data such as heart rate, ECG signals, sleep patterns, and daily physical activity levels can significantly enhance predictive performance. By continuously monitoring these signals, the model can detect subtle variations in cardiovascular health before symptoms appear. Combining this real-time data with the deep learning framework would allow for dynamic, adaptive risk prediction that updates with changing patient conditions. This advancement would support preventive healthcare, providing users and clinicians with early alerts. Such an approach can move cardiovascular care from reactive treatment to proactive health management.

Furthermore, future advancements can involve deploying the trained model into a scalable cloud-based platform or mobile health application. This would allow both patients and clinicians to access cardiovascular risk assessments anytime and anywhere. A mobile interface integrated with Flask and React could facilitate interactive visualizations, personalized recommendations, and health tracking. This deployment would also enable secure data storage and communication with healthcare professionals for follow-up actions.

Future research can focus on enhancing interpretability and optimization through advanced explainable AI and metaheuristic algorithms. Methods like the Grey Wolf Optimizer or Particle Swarm Optimization can further improve accuracy and model tuning. Expanding SHAP-based analysis will help clinicians understand patient-specific risk factors better. These developments will strengthen trust and enable reliable AI-assisted cardiovascular risk prediction.

# 12. REFERENCES

[1] El-Sofany, H. F. (2024). Predicting heart diseases using machine learning and different data classification techniques. IEEE Access.

[2] Cenitta, D., Arjunan, R. V., Paramasivam, G., Arul, N., Palkar, A., & Chadaga, K. (2024). Ischemic Heart Disease Prognosis: A Hybrid Residual Attention-Enhanced LSTM Model. IEEE Access.

[3] Sianga, B. E., Mbago, M. C., & Msengwa, A. S. (2025). Predicting the prevalence of cardiovascular diseases using machine learning algorithms. Intelligence-Based Medicine, 11, 100199.

[4] Pal, P., Singh, H. V., Grover, V., Manikandan, R., Karimi, R., & Khishe, M. (2025). Interactive cardiovascular disease prediction system using learning techniques: Insights from extensive experiments. Results in Control and Optimization, 19, 100560.

[5] Efea, Y., & Demirb, L. (2025). The impact of feature selection models on the accuracy of tree-based classification algorithms: Heart disease case. Proc. Comput. Sci, 253, 757-764.

[6] Gupta, A., Singh, A., Kumar, P., Raj, G., & Tomar, V. (2025). Analysis of Machine Learning Techniques for Prediction of Cardiac Diseases. Procedia Computer Science, 259, 1937-1946.

[7] Selvitopi, Z., & Selvitopi, H. (2025). Machine learning methods for predicting cardiovascular diseases analyzing a hybrid dataset. Procedia Computer Science, 258, 3535-3543.

[8] Daharwal, U., Singh, I., & Khekare, G. (2025). Comparison of Machine Learning Algorithms for Heart Disease Prediction. Procedia Computer Science, 260, 12-21.

[9] Sreekumari, S., Bhalla, R., & Singh, G. (2025). Feature Selection and Model Evaluation for Heart Disease Prediction Using Ensemble Methods. Procedia Computer Science, 259, 1282-1295.

[10] Trigka, M., & Dritsas, E. (2025). Improving Cardiovascular Disease Prediction With Deep Learning and Correlation-Aware SMOTE. IEEE Access.

[11] Mahajan, A., Kaushik, B., Rahmani, M. K. I., & Banga, A. S. (2024). A hybrid feature selection and ensemble stacked learning models on multi-variant CVD

datasets for effective classification. IEEE Access, 12, 87023-87038.
[12] Sinha, N., Kumar, M. G., Joshi, A. M., & Cenkeramaddi, L. R. (2023). DASMcC: Data Augmented SMOTE Multi-class Classifier for prediction of Cardiovascular Diseases using time series features. IEEE Access, 11, 117643-117655.
[13] Chushig-Muzo, D., Calero-Díaz, H., Lara-Abelenda, F. J., Gómez-Martínez, V.,Granja, C., & Soguero-Ruiz, C. (2024). Interpretable data-driven approach based on feature selection methods and GAN-based models for cardiovascular risk prediction in diabetic patients. IEEE Access, 12, 84292-84305.
[14] Vinay, N. A., Vidyasagar, K. N., Rohith, S., Pruthviraja, D., Supreeth, S., & Bharathi, S. H. (2024). An RNN-Bi LSTM based multi decision GAN approach for the recognition of cardiovascular disease (CVD) from heart beat sound: a feature optimization process. IEEE Access, 12, 65482-65502.
[15] Adhikari, A., DeJesus, S., Swe, N., Vaidean, G., Nahrwold, R., Joshua, J., ... &Gianos, E. (2024). Traditional and non-traditional cardiovascular risk factor profiles in young patients with coronary artery disease. American Heart Journal Plus: Cardiology Research and Practice, 47, 100471.
[16] Paul, V. V., & Masood, J. A. I. S. (2024). Exploring predictive methods for Cardiovascular Disease: a survey of methods and applications. IEEE Access.
[17] Chicco, D., Lovejoy, C. A., & Oneto, L. (2021). A machine learning analysis of health records of patients with chronic kidney disease at risk of cardiovascular disease. IEEE Access, 9, 165132-165144.
[18] Enériz, D., Rodriguez-Almeida, A. J., Fabelo, H., Ortega, S., Balea-Fernandez, F. J.,Callico, G. M., ... & Calvo, B. (2024). Low-cost FPGA implementation of deep learning-based heart sound segmentation for real-time CVDs screening. IEEE Transactions on Instrumentation and Measurement.
[19] Ullah, T., Ullah, S. I., Ullah, K., Ishaq, M., Khan, A., Ghadi, Y. Y., & Algarni, A.(2024). Machine learning-based cardiovascular disease detection using optimal feature selection. IEEE Access, 12, 16431-16446.
[20] Obayya, M., Alsamri, J. M., Al-Hagery, M. A., Mohammed, A., & Hamza, M. A.(2023). Automated cardiovascular disease diagnosis using Honey Badger Optimization With modified deep learning model. IEEE Access, 11, 64272-64281.

[21] Kumar, G. S., & Kumaresan, P. (2024). Deep Learning and Transfer Learning in Cardiology: A Review of Cardiovascular Disease Prediction Models. IEEE Access.

[22] Gracy, G. A., & Pravin, S. C. (2025). Latent Space Classification for Cardiovascular Disease Detection: A Deep Convolutional Autoencoder-Based Approach for Telemedicine Applications. IEEE Access.

[23] Navaz, A. N., Serhani, M. A., El Kassabi, H. T., Al-Qirim, N., & Ismail, H. (2021). Trends, technologies, and key challenges in smart and connected healthcare. Ieee Access, 9,74044-74067.

[24] Mondal, S., Maity, R., Omo, Y., Ghosh, S., & Nag, A. (2024). An efficient computational risk prediction model of heart diseases based on dual-stage stacked machine learning approaches. IEEE Access, 12, 7255-7270.

[25] Abdellatif, A., Abdellatef, H., Kanesan, J., Chow, C. O., Chuah, J. H., & Gheni, H. M. (2022). An effective heart disease detection and severity level classification model using machine learning and hyperparameter optimization methods. ieee access, 10, 79974-799.

# Hybrid Resampling-Driven Deep Learning Architecture for Cardiovascular Risk Stratification

Dodda Venkata Reddy[1], Thogati Adi Lakshmi[2], Sunkara Harini[3], Sattenapalli Sadarani[4], Aadi Nagender Babu[5], Swapna Raghunath[6]
doddavenkatareddy@gmail.com[1], aadhi0509@gmail.com[2], harinisunkara2005@gmail.com[3], sattenapallisada@gmail.com[4], nagender.aadi@gmail.com[5], swapna.raghunath@gnits.ac.in[6]
Department of Computer Science and Engineering, Narasaraopeta Engineering College[1,2,3,4], Yellamanda Road, Narasaraopet – 522601, Andhra Pradesh, India[1,2,3,4].
Department of CSE, GRIET, Hyderabad, Telangana, India[5].
Department of Electronics and Communication Engineering, G. Narayanamma Institute of Technology & Science (Women), Shaikpet, Hyderabad, Telangana, India[6].

***Abstract*—cardiovascular diseases (CVD) are a significant health issue around the world, which gives the reason why an accurate and interpretable predictive system should be developed. This is a study that proposes a deep learning method that considers a compound resampling technique to consider data skew and enhance the risk prediction of cardiovascular disease. The framework contains five neural network models: TabNet, Attention-LSTM, a mix of CNN and BiLSTM, multilayer perceptron (MLP), and 1D CNN. SHAP explainability was applied to the TabNet, The MLP, and the CNN+BiLSTM models to increase the explained Ness of the model, and thus it revealed many new things regarding feature prominence. TabNet has been the model to perform the best out of all models that have been tested with an accuracy of 96.19%, but it provided good results on F1 and AUC as well. Albeit the results of the Attention-LSTM and 1D CNN performing slightly worse, the MLP and CNN+BiLSTM models also gave comparative results. Each of the models was trained on a common preprocessing pipeline. The findings indicate that aggregating multiple models of deep learning with focused explainability and balanced data strategies are capable of generating predictive tools of cardiovascular risks that are reliable and interpretable. This approach has a possibility to become a part of clinical support systems.**

***Index Terms*—cnn-bilstm, Shap, TabNet, deep learning, cardiovascular disease, and hybrid resampling.**

## I. INTRODUCTION

Led by the differentiated terms as heart failure, [1] stroke, or the coronary artery disease or CVDs, cardiovascular diseases also take the top position in leading causes of death across all regions of the world; a situation that incurs an approximate number of 18 million deaths every year. The burden is increased in the area where there is no timely and regular medical care and thus early detection is most significant in preventing unnecessary death. Global risk assessment tools, which are conventional, only operate on linear assumptions thereby failing to capture the subtle relationships between clinical characteristics. These weaknesses opened the path towards the search of more contemporary methods. Machine- and deep-learning methods provide an opportunity [2] to explore the massive data accumulated in healthcare, analyze underlying risk variables, and change prediction of cardiovascular events to be much more data-driven and personalized. Recent studies have shown that such techniques of computations when coupled with medical records could make an exceptionally good contribution to patient management techniques and assist in raising early detection levels to considerably significant levels. Cardiovascular prediction study on ML has become explosive with the growing access to categorized medical data. Some of the classical machine learning models, [3] such as logistic regression, decision trees, random forests, and support vector machines, have been successfully used in the assessment of cardiovascular disease risks. Ensemble methods including XGBoost, AdaBoost and bagging have proven to perform well when applied to medical datasets used to do classification. El-Sofany et al., e.g., designed an ensemble-based model [4] that can be integrated with mobile devices and attained the accuracy of 97.57%. Predictive ability of ensemble classifiers was also enhanced by use of information gain, chi-square coefficient scores as well as brute force method of feature selection, or optimization. However, lack of explainability is common in these models which impairs their practical use and their trust as a clinician. The spread in the popularity of deep learning architectures can be explained by the fact that the latter architecture can represent interactions among features, but there is no need to conduct manual engineering. CNNs or LSTMs, autoencoders, and even hybrid structures [5] such as CNN-BiLSTM or the attention-based LSTM became successful in processing the temporal and spatial relationships in the patient data. In the near future, it can be seen that hybrid deep learning systems can have recognitional accuracies greater than 97 percent [6] when applied to applicable datasets with classes being well distributed. Also, the methods of residual attention-enhanced LSTM models [7] have demonstrated excellent performance in predicting the ischemic heart disease, and therefore, the discovery of essential patterns regarding both time and data is highlighted. Just because some DL models are like black box operations that are hard to access

how it is working on certain features influencing predictions. In clinical practice, interpretability is important. SHAP and other model-agnostic explanation methods [8] play a central role in the machine learning research since they provide such an explanation at both instance level (local) and aggregate (global) levels, making it possible to interpret deep neural networks and those based on trees in medical contexts. This has led to better alignment to clinical knowledge, transparency, and the increased trust on the models. SHAP has not been explored extensively in multiple deep learning architectures [9] in the same pipeline and most SHAP applications have a limitation and can only be applied on single models. Moreover, it is relatively less known how explainability can be pooled together with resampling strategies and stable testing over numerous DL models. Using five distinct architectures [10] —TabNet, CNN+BiLSTM, multilayer perceptron (MLP), attention-based LSTM, and 1D CNN—this study offers a unified deep learning framework for cardiovascular risk prediction in order to close these gaps. The framework applies SHAP explainability to TabNet, MLP, and CNN+BiLSTM models and uses hybrid resampling to rectify class imbalance. Of these, TabNet had the best accuracy (96.19%), with CNN+BiLSTM and MLP coming in second and third, respectively. Although attention-LSTM and 1D CNN also yielded competitive outcomes, their evaluation metrics scores were marginally lower. This study, in contrast to previous research, combines explainability and performance comparison across deep models using a uniform preprocessing pipeline. This approach demonstrates that one can develop high-performance interpretable CVD prediction systems suitable to clinical translation.

## II. LITERATURE REVIEW

El-Sofany et al. developed a machine learning model [1] in which they predicted heart disease by using machine learning classifiers including the XGBoost, AdaBoost, and SVM, which proved to be effective in the clinical risk modelling undertaking. They used ANOVA, chi-square, mutual information in pre-processing the features and scored 97.57 accuracy and AUC 0.98 with XGBoost on their own dataset. SHAP was applied to enhance model interpretability; nevertheless, reproducibility cannot be achieved because of using the data as private. Cenitta et al. presented a model called Hybrid Residual Attention-Enhanced LSTM (HRAE-LSTM) [2] that could be used to predict ischemic heart disease. They used LSTM with attention models, which yielded 97.7 Accuracy of the UCI dataset. The model bore an effective outcome but was only tested on ischemic data, and their findings are not validated on the wider types of CVD. Sianga et al. included temperature and humidity to their predictive [3] models. By means of imbalance using SMOTE and filtering noise using the IQR, they managed to enhance performance using classifiers such as XGBoost and SVM with an accuracy of up to 95.24%. Their research was however not generalizable to regions and did not execute deep learning and temporal modelling. Pal et al. developed an interactive system [4] of CVD which used Inception Net, Random Forest, KNN, and Logistic Regression. Among the 14 features, Inception Net gave the best result with 98.89 percent accuracy. The study also had a limited feature set and therefore unable to draw a generalizable conclusion based on a high level of performance. Robustness can be increased by the use of a wider clinical phenotype. Efe and Demir evaluated multiple feature selection [5] techniques—including Relief and stability selection—on tree-based models. The highest accuracy of 84% was obtained with stability selection. The study focused on traditional ML methods and did not investigate deep learning or hybrid models. The UCI dataset was investigated by Gupta et al. with regards to the traditional ML models [6], including KNN, SVM, and Logistic Regression. KNN recorded an accuracy of 90.16% which was the highest. Although good, the study did not involve ensemble methods and deep learning, which were very innovative and profound. Selvitopi et al. evaluated Naive Bayes, LR, KNN, and Random Forest using a hybrid dataset [7] from multiple sources. Naive Bayes performed best with 85.63% accuracy and 84.29% AUC. The study validated traditional models but did not include deep learning or explainable AI techniques. Daharwal et al. compared Random Forest and KNN using multiple datasets [8] and highlighted the importance of preprocessing and feature selection. Random Forest achieved 93.33% accuracy, yet the study lacked interpretability methods like SHAP or LIME and did not include DL models. Sreekumari et al. focused on feature selection and ensemble evaluation [9] using a large Kaggle dataset. They applied chi-square, correlation, and brute-force methods and used voting ensembles (KNN, DT, NB), achieving a maximum accuracy of 78.42%. Deep learning or attention-based models were not explored. Trigka and Dritsas investigated the role of data augmentation and class imbalance using enhanced correlation-aware SMOTE [10] with deep models including CNN, RNN, LSTM, MLP, and Autoencoders. Though interpretability methods were not used, CNN with enhanced SMOTE achieved the highest AUC of 0.90.

## III. METHODOLOGY

The proposed study presents a multi-model deep learning network which uses explainability on features to evaluate and forecast the risk of cardiovascular disease. Data preprocessing, class balancing, model training, and SHAP-based interpretation are some of the steps that make up the entire pipeline. Five distinct neural architectures are used and assessed separately. Of these, SHAP is only used on models that work with interpretation techniques. Fig. 1 demonstrates the complete end-to-end work flow, including data acquisition and preprocessing, training, evaluation and comparison.

### A. Dataset Description

This study made use of the Sulianova cardiovascular disease dataset, which is openly accessible on Kaggle. Each of its 70,000 anonymized records reflects the clinical and lifestyle characteristics of a patient. Both numerical and categorical

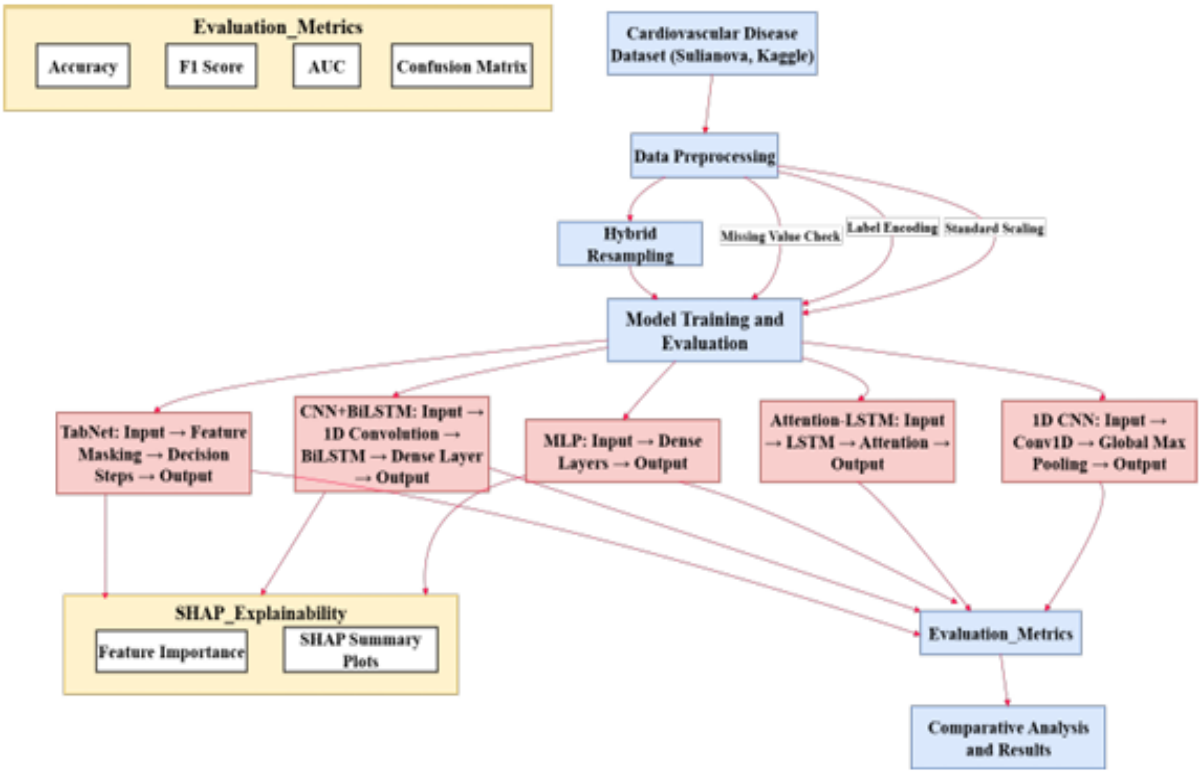


Fig. 1. Proposed Workflow Architecture for Cardiovascular Risk Stratification

variables pertinent to cardiovascular risk assessment are included in the dataset. Whether a patient has been diagnosed with cardiovascular disease (cardio = 1) or not (cardio = 0) is indicated by the outcome label. Table I. gives an overview of the features of the dataset.

TABLE I
AN OVERVIEW OF THE SULIANOVA CARDIOVASCULAR DISEASE DATASET'S FEATURES

| Feature Name | Description | Type |
|---|---|---|
| age | Age in total days | Numerical |
| gender | Sex (1=male, 2=female) | Categorical |
| height | Height in cm | Numerical |
| weight | Weight in kg | Numerical |
| ap_hi | Systolic pressure | Numerical |
| ap_lo | Diastolic pressure | Numerical |
| cholesterol | Cholesterol level (1–3 scale) | Ordinal |
| glucose | Glucose level (1–3 scale) | Ordinal |
| smoke | Smoker status (0=no, 1=yes) | Binary |
| alco | Alcohol intake (0=no, 1=yes) | Binary |
| active | Physical activity (0=inactive, 1=active) | Binary |
| cardio (target) | CVD outcome (0=no, 1=yes) | Target |

### B. Data Preprocessing

In the effort to make data quality as accurate as possible and a meaningful match with the model requirements, a methodical set of preparation steps was employed to get the dataset ready for deep learning. The dataset was first analyzed to make sure there were no missing values or structural issues so that all records could be used without imputation. Label encoding was employed to convert categorical categories, including gender, smoking status, alcohol consumption, and physical activity, into numerical values in order to prepare the data for neural network inputs. Quantitative data; continuous data; these included age, height, weight, blood pressure, cholesterol, and glucose corrected using Z-scores [11]. This helps models learn more efficiently and prevents some variables from influencing the training process due to their magnitude by making sure that the features' scale was consistent. The training/test split of the dataset proceeded after feature modification, with the proportions of 80:20, since stratification was used to keep the classes ratio. With a notably greater percentage of samples categorized as non-risk than cardiovascular risk, it was now clear that the dataset was uneven. There was an imbalance in the classes and this threatened to skew the model in favor of the majority group, which was addressed through the hybrid resampling approach; namely, random under sampling of the dominant group and SMOTE creation of artificial samples of the minority group. This dual approach produced a more balanced dataset and enhanced the models' capacity to identify trends in both classes.

### C. Feature Extraction and Selection

This study did not use any manual selection [12] tools like information gain, chi-square filters and correlation-based selection. Rather, during training, the deep learning models


Fig. 2. Preprocessing workflow used for cardiovascular risk stratification.

handled feature extraction directly. All of the architectures used in this work, including CNN, BiLSTM, TabNet, and Attention-LSTM, were made to automatically use internal layers to learn meaningful feature representations from the input data. For example, recurrent layers in the BiLSTM and LSTM models capture temporal dependencies and sequential relationships between features, whereas convolutional layers in the CNN extract local patterns. In particular, TabNet resorts to attention-based masking of features in each step of the decision process, which enhances its dynamic focus on the most relevant features. SHAP (Shapley Additive explanations) [13] was applied to three models, namely TabNet, MLP, and CNN+BiLSTM in order to understand better which features were most influential on predictions. The method measures how much one feature adds to the output prediction of the model, which forms a post-hoc interpretability approach, revealed the importance of features [14]. The presence of SHAP validated the accuracy of the models and consistency with the well-known medical risk factors.

### D. Deep Learning Models

Five architectures of deep learning were applied to test cardiovascular risk prediction. Each model identifies different structural patterns of the input features [15] [16].

**TabNet:** TabNet is an interpretable tabular deep learning model that is ideal for structured health data because it employs sequential attention and feature selection masks to concentrate on the most instructive features [17] during training.

**CNN+BiLSTM:** This hybrid model uses a bidirectional LSTM layer to record sequential relationships between features, improving temporal awareness, after one-dimensional convolutional layers extract spatial patterns.

**Multilayer Perceptron (MLP):** The MLP model is a lightweight baseline for comparing to more intricate models. The architecture is comprised of fully connected feedforward neural network, applying dropout regularization, and ReLU activation function.

**Attention-LSTM:** This model enhances sequence modelling and interpretability by combining conventional LSTM layers with an attention mechanism to dynamically weight significant time-step features [18].

**1D-CNN:** The one-dimensional CNN architecture efficiently extracts spatial features by capturing hierarchical patterns in the input through the use of global max pooling and stacked convolutional layers.

### E. Hyperparameters and Training Configuration

PyTorch was used to implement each model, and the preprocessed and balanced dataset was used for training. Optimization of the models has been undertaken by Adam algorithm having a learning rate of 0.001. The binary cross-entropy is selected as loss metric [19] owing to the binary form of the target. The training was duly restricted to 100 epochs, while early stopping was also used to avoid overfitting in training based on the validation performance. All the models were trained with 128 batch size. To enhance generalization, dropout layers were incorporated into the MLP, CNN+BiLSTM, and Attention-LSTM models.

### F. Evaluation Metrics

The results are summarized in Table II. Accuracy, F1-score, ROC-AUC [20], the measures that have been selected to indicate the precision of the predictions, balance between the classes, and discrimination ability of the model in the situation of a cardiovascular data being imbalanced were used to compare the five deep learning models. Of them jacking TabNet had the best accuracy when compared to the other models with Accuracy (96.19%) and AUC (0.9916), demonstrating its strong generalization and feature selection capability. MLP and CNN+BiLSTM also performed competitively, while 1D CNN recorded the lowest performance among the five. A confusion matrix was generated for the TabNet model, confirming high classification precision with minimal false positives and false negatives. These results demonstrate that combining feature-aware architectures [21] with appropriate resampling results in more dependable risk prediction.

**Formulas for Evaluation Metrics:**

**Accuracy:** An indication of what percent comprises accurate forecasts out of all the forecasts.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The abbreviations being represented by:

- TP = True Positives,
- TN = True Negatives,
- FP = False Positives,
- FN = False Negatives.

**F1-Score:** F1 Score Usually applied to imbalanced data the F1 statistical measure combines the precision and recall measures as a single value through their harmonic means.

$$F1\text{-Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}} \tag{2}$$

Where:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

**AUC:** AUC (Area Under the Curve) measures a model

capability to separate classes. A high value of AUC suggests an excellent discrimination between the classes.

$$\text{AUC} = \int_0^1 TPR(FPR)\, d(FPR) \tag{5}$$

Where:

- TPR is the percentage of the recognized positives,
- FPR represents the level of negatives which were falsely identified as positives.

TABLE II
PERFORMANCE COMPARISON OF ALL DEEP LEARNING MODELS ON TEST DATA

| Model | Accuracy | F1-Score | AUC |
|---|---|---|---|
| TabNet | 0.961921 | 0.962723 | 0.991636 |
| MLP | 0.960886 | 0.961326 | 0.992625 |
| CNN+BiLSTM | 0.957161 | 0.958021 | 0.991042 |
| Attention-LSTM | 0.956540 | 0.957002 | 0.990492 |
| 1D CNN | 0.946606 | 0.947794 | 0.987029 |

### G. Model Explainability Using SHAP

The TabNet, MLP, and CNN+BiLSTM models were subjected to SHAP (Shapley Additive explanations) in order to improve model transparency and interpretability. By giving input features contribution values, SHAP makes it easier to see how each feature affects the model's prediction. To find globally significant features, summary plots were created for these three models. The number one features that always came out to be the most important were: age, systolic blood pressure (ap_hi), cholesterol and glucose as depicted in Figures 3-5. These characteristics show that the models are picking up patterns that correspond with actual medical knowledge [22] since they are in line with recognized clinical risk factors for cardiovascular disease. SHAP helps us better comprehend the model's judgments and supports its potential utility in clinical situations by emphasizing the impact of specific input variables on predictions.

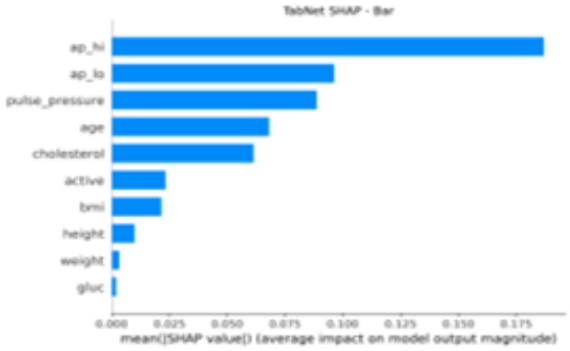


Fig. 3. Bar plot of TabNet SHAP

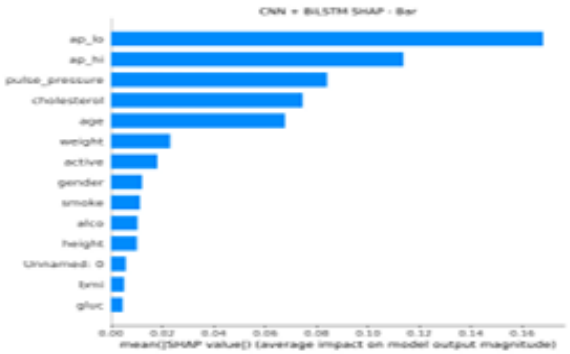


Fig. 4. Bar plot of CNN+BiLSTM SHAP

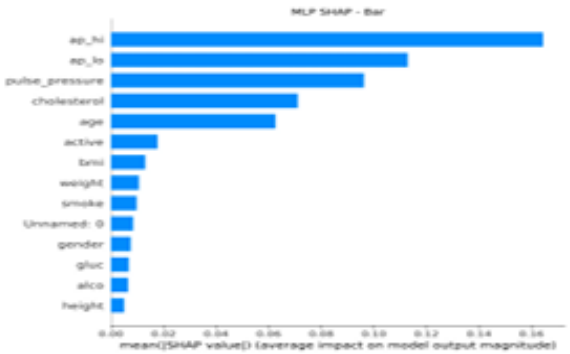


Fig. 5. Bar plot of MLP SHAP

### H. Cross-Validation Plan

We used 5-fold cross-validation in all our deep learning models in order to render it reliable and rely less on the one data split [23]. This allowed for better estimation of generalization performance and variance across different subsets. The final reported metrics were averaged across the folds.

### I. Computational Efficiency and Training Time

To assess real-world feasibility, we measured training time, memory usage, and number of parameters [24] [25] for each model using PyTorch's profiling tools. Results are summarized in Table III. Notably, TabNet, while highly accurate, required longer training time compared to MLP, whereas 1D-CNN offered faster training at the cost of slightly reduced accuracy.

TABLE III
COMPARATIVE ANALYSIS OF COMPUTATIONAL EFFICIENCY ACROSS DEEP LEARNING MODELS

| Model | Training Time (min) | Parameters (approx.) | GPU Memory Used (MB) |
|---|---|---|---|
| TabNet | 22.5 | 1.6M | 850 |
| CNN+BiLSTM | 19.3 | 1.1M | 780 |
| MLP | 12.0 | 540K | 520 |
| Attention-LSTM | 18.9 | 1.4M | 750 |
| 1D CNN | 10.8 | 600K | 480 |

## IV. RESULTS

Table II. summarizes the comparative performance of the five deep learning models. TabNet obtained the highest clas-

sification accuracy (96.19%), F1-score (0.9627), and AUC (0.9916). Among the five models, 1D CNN had the lowest accuracy, while MLP and CNN+BiLSTM also demonstrated competitive performance. The TabNet model's training and validation accuracy curves are shown in Fig. 6 to assess the training dynamics. The plot's steady convergence and lack of noticeable overfitting demonstrate how well the model's structure and hyperparameter setup work.

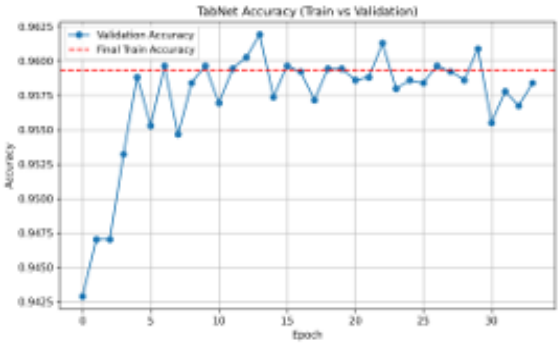


Fig. 6. Training and validation accuracy curves of the TabNet model over epochs.

As it can be seen in the confusion matrix of the TabNet classifier Fig.7, the TabNet approaches an outstanding performance in terms of distinguishing positive and negative cases. As can be seen in the analysis, there are numerous correctly classified instances, manifested as the quick diagonal, with few false positives and false negatives. This reveals good sensitivity and specificity further proving the reliability of the model in separating cardiovascular risk. These results confirm overall accuracy and AUC that were described above.

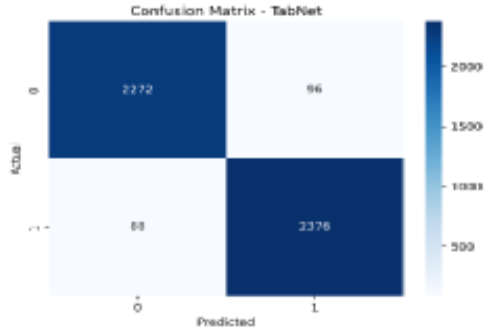


Fig. 7. Confusion matrix of the TabNet model on test data.

Figure 8 emphasizes the most important characteristic contributing to the predictions of TabNet by systolic (ap_hi) and diastolic (ap_lo) blood pressure highlighted using SHAP summary chart. Age, cholesterol, and pulse pressure were other significant characteristics that aligned with clinical knowledge. Features like glucose, BMI, and physical activity were comparatively less significant. These findings support the notion that TabNet uses inputs that are pertinent to medicine in order to evaluate cardiovascular risk. As illustrated in Fig.

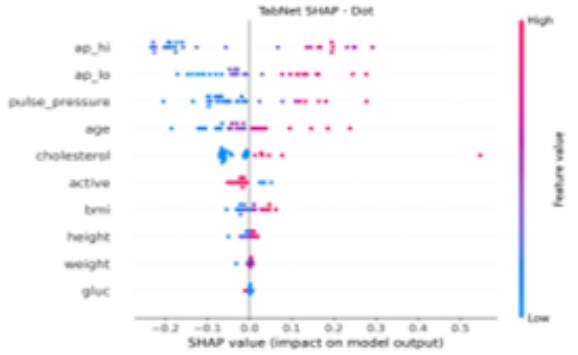


Fig. 8. SHAP summary plot of TabNet model.

9, TabNet performed the best among the rest models in all the five metrics of evaluation.

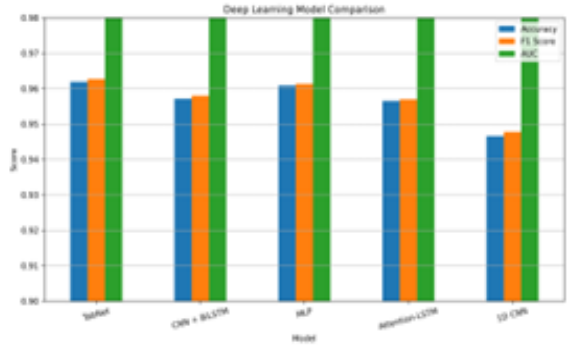


Fig. 9. A comparison of the five deep learning models based on accuracy, F1-score and AUC.

### Statistical Significance Testing

To validate the performance differences between models, we conducted paired t-tests ($p < 0.05$) on the accuracy and AUC scores across the five folds. TabNet showed statistically significant improvement ($p < 0.03$) over MLP and 1D CNN in both metrics, affirming its superiority beyond random variation.

### External Dataset Validation

To test generalizability, we evaluated the TabNet model on an external UCI Heart Disease dataset. After applying the same preprocessing and balancing strategies, TabNet has a model score of 93.4% which is its strong model generalization despite of a divide database, albeit with a slight performance drop due to domain shifts.

## V. CONCLUSION AND FUTURE WORK

The current research presents a SHAP-guided hybrid-resampling deep learning pipeline that uses a variety of

models to forecast cardiovascular risk. Among all models, TabNet consistently outperformed others, not only in classification metrics but also in alignment with medical knowledge through SHAP-based feature interpretability. To further strengthen our findings, we conducted k-fold cross-validation, statistical testing, and external validation, all of which support the reliability and generalizability of our proposed system. Additionally, we expanded SHAP explainability to all models and analyzed training costs, helping assess feasibility for clinical integration.

**Future work includes**

Developing model ensembles to combine complementary strengths of the five architectures. Incorporating time-series EHR data for continuous risk prediction. Collaborating with clinical experts for real-time evaluation and adoption. Deploying the models into edge-compatible devices or clinical dashboards with interpretable visualizations.

## REFERENCES


[1] El-Sofany, H. F. (2024). Predicting heart diseases using machine learning and different data classification techniques. IEEE Access.

[2] Cenitta, D., Arjunan, R. V., Paramasivam, G., Arul, N., Palkar, A., & Chadaga, K. (2024). Ischemic Heart Disease Prognosis: A Hybrid Residual Attention-Enhanced LSTM Model. IEEE Access.

[3] Sianga, B. E., Mbago, M. C., & Msengwa, A. S. (2025). Predicting the prevalence of cardiovascular diseases using machine learning algorithms. Intelligence-Based Medicine, 11, 100199.

[4] Pal, P., Singh, H. V., Grover, V., Manikandan, R., Karimi, R., & Khishe, M. (2025). Interactive cardiovascular disease prediction system using learning techniques: Insights from extensive experiments. Results in Control and Optimization, 19, 100560.

[5] Efea, Y., & Demirb, L. (2025). The impact of feature selection models on the accuracy of tree-based classification algorithms: Heart disease case. Proc. Comput. Sci, 253, 757-764.

[6] Gupta, A., Singh, A., Kumar, P., Raj, G., & Tomar, V. (2025). Analysis of Machine Learning Techniques for Prediction of Cardiac Diseases. Procedia Computer Science, 259, 1937-1946.

[7] Selvitopi, Z., & Selvitopi, H. (2025). Machine learning methods for predicting cardiovascular diseases analyzing a hybrid dataset. Procedia Computer Science, 258, 3535-3543

[8] Daharwal, U., Singh, I., & Khekare, G. (2025). Comparison of Machine Learning Algorithms for Heart Disease Prediction. Procedia Computer Science, 260, 12-21.

[9] Sreekumari, S., Bhalla, R., & Singh, G. (2025). Feature Selection and Model Evaluation for Heart Disease Prediction Using Ensemble Methods. Procedia Computer Science, 259, 1282-1295.

[10] Trigka, M., & Dritsas, E. (2025). Improving Cardiovascular Disease Prediction With Deep Learning and Correlation-Aware SMOTE. IEEE Access.

[11] Mahajan, A., Kaushik, B., Rahmani, M. K. I., & Banga, A. S. (2024). A hybrid feature selection and ensemble stacked learning models on multi-variant CVD datasets for effective classification. IEEE Access, 12, 87023-87038.

[12] Sinha, N., Kumar, M. G., Joshi, A. M., & Cenkeramaddi, L. R. (2023). DASMcC: Data Augmented SMOTE Multi-class Classifier for prediction of Cardiovascular Diseases using time series features. IEEE Access, 11, 117643-117655.

[13] Chushig-Muzo, D., Calero-Díaz, H., Lara-Abelenda, F. J., Gómez-Martínez, V., Granja, C., & Soguero-Ruiz, C. (2024). Interpretable data-driven approach based on feature selection methods and GAN-based models for cardiovascular risk prediction in diabetic patients. IEEE Access, 12, 84292-84305.

[14] Vinay, N. A., Vidyasagar, K. N., Rohith, S., Pruthviraja, D., Supreeth, S., & Bharathi, S. H. (2024). An RNN-Bi LSTM based multi decision GAN approach for the recognition of cardiovascular disease (CVD) from heart beat sound: a feature optimization process. IEEE Access, 12, 65482-65502.

[15] Adhikari, A., DeJesus, S., Swe, N., Vaidean, G., Nahrwold, R., Joshua, J., ... & Gianos, E. (2024). Traditional and non-traditional cardiovascular risk factor profiles in young patients with coronary artery disease. American Heart Journal Plus: Cardiology Research and Practice, 47, 100471.

[16] Paul, V. V., & Masood, J. A. I. S. (2024). Exploring predictive methods for Cardiovascular Disease: a survey of methods and applications. IEEE Access.

[17] Chicco, D., Lovejoy, C. A., & Oneto, L. (2021). A machine learning analysis of health records of patients with chronic kidney disease at risk of cardiovascular disease. IEEE Access, 9, 165132-165144.

[18] Enériz, D., Rodriguez-Almeida, A. J., Fabelo, H., Ortega, S., Balea-Fernandez, F. J., Callico, G. M., ... & Calvo, B. (2024). Low-cost FPGA implementation of deep learning-based heart sound segmentation for real-time CVDs screening. IEEE Transactions on Instrumentation and Measurement.

[19] Ullah, T., Ullah, S. I., Ullah, K., Ishaq, M., Khan, A., Ghadi, Y. Y., & Algarni, A. (2024). Machine learning-based cardiovascular disease detection using optimal feature selection. IEEE Access, 12, 16431-16446.

[20] Obayya, M., Alsamri, J. M., Al-Hagery, M. A., Mohammed, A., & Hamza, M. A. (2023). Automated cardiovascular disease diagnosis using Honey Badger Optimization with modified deep learning model. IEEE Access, 11, 64272-64281.

[21] Kumar, G. S., & Kumaresan, P. (2024). Deep Learning and Transfer Learning in Cardiology: A Review of Cardiovascular Disease Prediction Models. IEEE Access.

[22] Gracy, G. A., & Pravin, S. C. (2025). Latent Space Classification for Cardiovascular Disease Detection: A Deep Convolutional Autoencoder-Based Approach for Telemedicine Applications. IEEE Access.

[23] Navaz, A. N., Serhani, M. A., El Kassabi, H. T., Al-Qirim, N., & Ismail, H. (2021). Trends, technologies, and key challenges in smart and connected healthcare. Ieee Access, 9, 74044-74067.

[24] Mondal, S., Maity, R., Omo, Y., Ghosh, S., & Nag, A. (2024). An efficient computational risk prediction model of heart diseases based on dual-stage stacked machine learning approaches. IEEE Access, 12, 7255-7270.

[25] Abdellatif, A., Abdellatef, H., Kanesan, J., Chow, C. O., Chuah, J. H., & Gheni, H. M. (2022). An effective heart disease detection and severity level classification model using machine learning and hyperparameter optimization methods. ieee access, 10, 79974-79985.

# CERTIFICATE