Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis

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

S. Venkata Siva Naga Lakshmi	**(22471A05J9)**
K. Kaveri	**(22471A05G2)**
K. Pratyusha	**(22471A05G3)**

Under the esteemed guidance of

Dodda 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
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 Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis is a Bonafide work done by the team **S. Venkata Siva Naga Lakshmi (22471A05J9), K. Kaveri(22471A05G2), K. Pratyusha(22471A05G3)** in partial fulfilment 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

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

Assistant. Professor

PROJECT-COORDINATOR

Dodda 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

We declare that this project work titled **“Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis”** is composed by ourselves that the work contains here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

S.Venkata Siva Naga Lakshmi(22471A05J9)
K.Kaveri(22471A05G2)
K.Pratyusha(22471A05G3)

ACKNOWLEDGEMENT

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

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao,** M.Tech., Ph.D., HOD of CSE department and also to our guide **Dodda Venkata Reddy,** M.Tech.,Ph.D., of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dodda Venkata Reddy,M.Tech.,Ph.D.,** Professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

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

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions clarified our doubts, which really helped us in successfully completing our project.

By

S. Venkata Siva Naga Lakshmi(22471A05J9)

K.Kaveri(22471A05G2)

K.Pratyusha(22471A5G3)

NEC 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.

NEC 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 teamwork 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 program 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 academia, industry, and society.

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

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

NEC NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

Program Outcomes (PO'S)

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

NEC NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

Project Course Outcomes (CO'S)

CO421.1: Analyse 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	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	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	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	Gathering the requirements and defining the problem, plan to develop a model for recognizing for detecting the Kidney Disease prediction.	PO1, PO3,PO8
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified.	PO2, PO3,PO8
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work.	PO3, PO5, PO9, PO8
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated and evaluated in our project.	PO1, PO5,PO8
CC421.4, C2204.4, C22L3.2	Documentation is done by all our three members in the form of a group.	PO10,PO8
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically.	PO10, PO8,PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the doctors and in future updates in our project can be done based on detection of forged videos.	PO4, PO7,PO8
C32SC4.3	The physical design includes website to check whether a Value is diseased or not.	PO5, PO6,PO8

ABSTRACT

Chronic Kidney Disease (CKD) is a silent and insidious disease that frequently evades early diagnosis, particularly among individuals with restricted access to periodic diagnostics. In this research, we introduce an interpretable, ensemble-driven approach called ”Echoes of the Hidden Filter” that leverages the strength of hybrid machine learning to facilitate early and correct CKD prognosis. Our strategy combines several classifiers such as Random Forest, XGBoost, Support Vector Machines, and an Artificial Neural Network into a stacking ensemble and majority-vote model to provide robust performance irrespective of varied patient profiles and data anomalies. To overcome clinical interpretability, we use SHAP (Shapley Additive explanations) and LIME (Local Interpretable Model-agnostic Explanations) to offer feature-level transparency both globally and at the individual prediction level. The two-level interpretability facilitates clinicians in recognizing which biomarkers—e.g., eGFR, creatinine, and cystatin-C—inform each prediction and why, thereby building trust and informing diagnostic decisions. We confirm our model with a real-world CKD dataset that is high-dimensional, imbalanced, and partially missing. Statistically-informed methods were utilized to impute missing values, while class imbalance was reduced using SMOTE. Our ensemble model produced better accuracy, precision, recall, and F1-score values compared to single models with significantly better performance. Through the integration of predictive power and interpretability, this model fills in the gap between black-box AI and clinical usability. The suggested approach is a scalable, transparent tool for population-level CKD screening, especially useful in resource-limited healthcare settings.

INDEX

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

1.1 Introduction

Chronic Kidney Disease (CKD) is a progressive, non-communicable condition characterized by the gradual loss of kidney function over a prolonged period, posing a serious global health challenge [1]. CKD is a major contributor to morbidity and mortality worldwide, accounting for millions of premature deaths each year and imposing a heavy socioeconomic burden on healthcare systems. The disease affects an estimated 850 million people globally, with a prevalence of approximately 9–13% in the adult population, making it one of the most common chronic diseases [2]. Alarmingly, in many regions, particularly low- and middle-income countries, the prevalence is rising due to demographic shifts, urbanization, and the global surge in risk factors such as diabetes, hypertension, and obesity.

One of the most significant challenges in CKD management is its asymptomatic nature during the early stages. In the majority of cases, symptoms do not manifest until significant kidney damage has occurred, by which time treatment options become more complex, costly, and less effective [2]. Consequently, a substantial proportion of patients are first diagnosed during advanced stages, when therapeutic choices are largely confined to renal replacement therapies such as dialysis or kidney transplantation. These interventions, while life-sustaining, come with high financial costs, resource requirements, and reduced patient quality of life. Dialysis, for instance, requires continuous and long-term medical infrastructure, while transplantation depends on the availability of compatible donors and lifelong immunosuppressive therapy.

Current CKD detection strategies primarily rely on biochemical markers, such as serum creatinine, blood urea nitrogen, and the estimated glomerular filtration rate (eGFR), alongside urine tests for proteinuria or albuminuria. While these markers are clinically validated and widely accessible, they tend to indicate disease only after substantial nephron loss—often greater than 50%—has already occurred [3]. This delayed detection severely limits opportunities for preventive intervention, including lifestyle changes, dietary modifications, and early pharmacological treatments, all of which could slow disease progression if initiated in the initial stages. As a result, there is a growing consensus in the medical community that early, predictive screening methods are essential for reducing CKD-related morbidity and mortality [3].

Recent advances in data analytics, artificial intelligence (AI), and machine learning (ML) have opened promising avenues for early CKD prediction. Unlike traditional statistical methods, ML algorithms can process large-scale, heterogeneous datasets containing demographic, biochemical, clinical, and lifestyle information, identifying subtle, non-linear patterns that may indicate elevated CKD risk [4]. This capability allows predictive modeling to detect at-risk individuals earlier than conventional methods, potentially enabling proactive management strategies. Various ML algorithms, such as Random Forest (RF), Support Vector Machines (SVM), Gradient Boosting Machines (GBM), and Artificial Neural Networks (ANN), have been evaluated for CKD risk prediction, often demonstrating high predictive accuracy [5].

Among these techniques, ensemble learning methods—particularly stacking—have emerged as a powerful solution for improving model generalization and stability. Ensemble learning combines multiple base models to produce a single, more accurate predictive output. Stacking, in particular, involves training several diverse algorithms (e.g., RF, SVM, ANN, XGBoost) and using a meta-learner to aggregate their predictions [6]. This approach reduces the weaknesses of individual algorithms—such as overfitting in complex models or underfitting in simpler ones—while exploiting their complementary strengths. In medical applications, this is particularly valuable, as it ensures consistent predictive performance across different patient subgroups and clinical scenarios.

Despite the high performance of many ML-based CKD prediction models, a major barrier to clinical adoption remains the lack of interpretability. Many of the most accurate models, especially deep learning architectures, function as “black boxes,” generating predictions without revealing the reasoning behind them [7]. In clinical contexts, where decision-making impacts patient safety, transparency is not optional—it is a necessity. Healthcare providers must understand why a model predicts that a patient is at high risk, including which factors were most influential in that decision [8]. Without this level of transparency, trust in AI systems remains limited, hindering their integration into routine healthcare workflows.

Explainable Artificial Intelligence (XAI) has emerged as a promising solution to address this trust gap. XAI methods aim to make the decision-making process of complex models transparent and understandable to human users. Among these methods, Shapley Additive Explanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) are two of the most widely adopted tools in healthcare analytics [9]. SHAP provides a global perspective by quantifying the contribution of each feature to the overall model predictions, enabling clinicians to identify which factors—such as eGFR, serum creatinine, blood pressure, or diabetes history—most strongly influence CKD risk. LIME, on the other hand, offers localized, instance-level interpretability, explaining why a particular patient received a certain prediction [10]. Together, these techniques ensure that both the overall model behavior and individual patient-level predictions are interpretable, enhancing trust and facilitating regulatory compliance.

Leveraging these advancements, our research proposes *Echoes of the Hidden Filter*, a novel, ensemble-based interpretability framework for early CKD prediction. The framework begins with robust data preprocessing, addressing key issues in real-world healthcare datasets, such as missing values, imbalanced class distributions, and irrelevant or redundant features. Missing values are handled through statistical imputation methods, ensuring no loss of critical patient records. To address the challenge of imbalanced datasets—where healthy cases typically outnumber CKD cases—the framework incorporates a dynamic Synthetic Minority Oversampling Technique (SMOTE) to improve model sensitivity without inflating false positives. Feature selection is performed using a variance threshold approach, reducing dimensionality while retaining the most informative predictors.

For classification, the framework employs a stacking-based ensemble that integrates multiple base learners, including RF, XGBoost, SVM, and ANN. This combination ensures both non-linear learning capability and interpretability, with the meta-learner providing a unified prediction. To enhance reliability, a majority-vote mechanism is applied at the final decision stage, safeguarding against variability in individual model outputs.

Crucially, SHAP and LIME are integrated directly into the framework, providing both global and local interpretability. SHAP highlights the most influential features across the dataset, while LIME explains predictions for individual patients, ensuring clinicians understand both the general trends and the specific reasoning behind each case. This dual-layer interpretability bridges the gap between AI performance and clinical trust, making the framework suitable for deployment in diverse healthcare environments.

By merging predictive accuracy with transparency, *Echoes of the Hidden Filter* represents a step toward scalable, trustworthy AI-assisted CKD screening. Its design particularly benefits resource-limited healthcare systems, where early detection can significantly reduce the burden of late-stage CKD treatment, ultimately improving patient outcomes while optimizing resource allocation.

2.LITERATURE SURVEY

2.1 LITERATURE REVIEW

Diagnosis, monitoring, and control of chronic kidney disease (CKD) have been substantially improved by both biomedical research and Machine Learning (ML) methods. The performance of CKD prediction models relies on data quality, choice of biomarkers, model architecture, explainability, and their integration into clinical practice. Clinical data acquisition is core. Kanasaki et al. [1] detailed the continued essential role of standard markers like eGFR and albuminuria in diabetic nephropathy staging, while Elcioglu et al. [2] emphasized their utility in ADPKD surveillance.
Li et al. [3] demonstrated how non-coding RNAs provide novel biomarkers outside conventional blood markers. Hart et al [4] highlighted that improved treatments such as HEMT in cystic fibrosis have prolonged life expectancy while exposing new CKD risks.

Souza et al [5] proposed a reagent-free microwave sensor as a low-cost, precise substitute for CKD screening. Bianchi et al [6] demonstrated that transformerbased models integrating complex multi-dimensional data outperform conventional predictors.

Puri et al [7] proposed blockchain-based AI for secure, decentralized CKD forecasting. Zhao et al [8] introduced a Deep Belief Rule Base model where transparency is coupled with scalability.

Ghosh et al [9] enhanced CKD staging with classifier optimization and SHAP-based explainability. Jawad et al. [10] applied interpretable ensemble models to increase clinical confidence.

Popoola et al [11] employed clustering on mixed and incomplete CKD datasets from South Africa, advocating better handling of missing values for early detection in heterogeneous populations.

Moreno-Sanchez [12] developed an interpretable tree based ensemble AI for CKD detection, illustrating a balance between performance and clinical explainability.

Cui et al [13] introduced a hybrid U-shaped deep learning model for automatic kidney volume segmentation in PKD, aiding long-term structural monitoring. Akter et al [14] compared deep learning models and found ensemble architectures offered better accuracy and interpretability through feature importance.

Finally, Chabouh et al [15] presented ultrasound localization microscopy for non-invasive, high-resolution kidney imaging, improving CKD structural assessment. Recent advancements in artificial intelligence have greatly improved kidney disease diagnosis and analysis using high-level image-based modeling.

Sharaby et al. [16] came up with a segmentation system that employs a modified CycleGAN model along with appearance-based shape priors. The system successfully captured kidney boundaries in medical images, enabling more precise anatomical analysis.

Chaki and Uc¸ar [17] introduced an inductive transferbased ensemble deep learning framework for kidney stone detection. Their system utilized pre-learned representations to enhance classification performance, especially in scenarios with few annotated data.

Sharen et al. [18] presented MSKd Net, a deep learning model incorporating multi-head attention in a Swin Transformer framework. The model attained strong classification performance for various kidney disease stages and exemplified the advantages of transformerbased models in medical diagnostic complexity.

Barros et al. [19] investigated how the incorporation of expert interaction with AI systems enhances the detection of podocyte degeneration. Through the fusion of automated processing with pathologist feedback, their approach increased the accuracy of histological examination for kidney disease evaluation.

Pande and Agarwal [20] developed a computed tomography (CT)-oriented system with the potential to detect various kidney abnormalities. Their system facilitated early and non-invasive diagnosis of various renal conditions, which improved the efficiency of screening procedures in clinical practice.

3.SYSTEM ANALYSIS

The “Echoes of the Hidden Filter” framework for CKD prediction was developed after conducting an in-depth system analysis to ensure it meets both functional and non-functional requirements essential for clinical application. From a functional perspective, the system is designed to process heterogeneous patient data, including demographic details, biochemical test results, lifestyle factors, and comorbidity information. Data preprocessing modules handle missing values, inconsistencies, and noise, followed by variance-based feature selection to retain only the most significant predictors. The prediction engine employs a stacking ensemble model that combines multiple classifiers—Random Forest, Support Vector Machine, Artificial Neural Network, and XGBoost—to leverage their complementary strengths. A meta-learner integrates these outputs to deliver the final prediction. In addition, explainability modules using SHAP and LIME provide global and local interpretability, enabling clinicians to understand both the most influential biomarkers across the dataset and the rationale behind individual patient predictions.

Non-functional requirements focus on scalability, accuracy, computational efficiency, and regulatory compliance. The framework is capable of handling datasets ranging from small-scale clinic records to large multi-institutional repositories, ensuring adaptability in various healthcare environments. It is optimized for high prediction accuracy while maintaining robustness under varying data distributions. Computational efficiency enables near real-time predictions, which is critical for clinical workflows. Data privacy is ensured by aligning with HIPAA and GDPR standards. The feasibility analysis indicates technical viability due to mature open-source libraries such as Scikit-learn, TensorFlow, and SHAP/LIME; operational viability through modular design that supports integration with Electronic Health Record systems; and economic feasibility by relying on cost-effective open-source tools and reducing long-term healthcare costs through early detection of CKD.

Potential challenges, such as variability in data quality, risk of overfitting, and ensuring the validity of interpretability outputs, are addressed through strategies like periodic retraining, domain adaptation techniques, and continuous clinician feedback. Overall, this system analysis demonstrates that the proposed framework effectively balances predictive performance with interpretability, positioning it as a reliable and transparent

AI-based solution for proactive CKD screening, particularly in resource-limited healthcare settings.

3.1 EXISTING SYSTEM

The existing systems for Chronic Kidney Disease (CKD) prediction and diagnosis primarily rely on conventional clinical methods and early implementations of machine learning models. Traditionally, CKD detection is carried out using clinical biomarkers such as serum creatinine levels, estimated glomerular filtration rate (eGFR), blood urea nitrogen (BUN), and urine albumin tests. While these methods are standard in medical practice, they often detect CKD only after significant kidney damage has already occurred, reducing the opportunity for early intervention. This delay in detection is particularly problematic in the early stages of CKD, which are typically asymptomatic, leading to a high proportion of undiagnosed cases. Moreover, these conventional approaches depend heavily on periodic medical checkups and specialized laboratory tests, which may not be accessible in low-resource healthcare environments.

In the last decade, researchers have explored machine learning-based systems to enhance CKD detection and prognosis. Existing ML-based approaches utilize algorithms such as Decision Trees, Random Forest, Support Vector Machines, and Artificial Neural Networks to analyze patient data and predict disease presence. These models have shown promising accuracy and can process a wide range of variables, including demographic details, biochemical test results, and lifestyle-related data. However, many of these systems are standalone classifiers, meaning they rely on a single algorithm, which may lead to performance instability across different datasets due to underfitting, overfitting, or sensitivity to noise. Additionally, while some ensemble approaches exist, they are often designed solely for maximizing prediction accuracy without addressing the interpretability challenge. This “black box” nature makes it difficult for clinicians to trust and adopt these systems in real-world decision-making.

Another limitation of existing systems is the handling of imbalanced dataset common issue in CKD research where the number of patients without CKD significantly outweighs those with CKD. Many current models either ignore this imbalance or use basic oversampling/undersampling techniques that may lead to biased predictions. Furthermore, most existing tools lack real-time processing capabilities and are not well-integrated with Electronic Health Record (EHR) systems, making their deployment in clinical settings cumbersome. Overall, while existing systems have contributed significantly to CKD prediction research, they often fall short in terms of early detection capability, interpretability, scalability, and seamless integration into healthcare workflows. This creates the need for a more advanced, transparent, and robust system like "Echoes of the Hidden Filter," which addresses these gaps comprehensively.

3.1.1 DIS-ADVANTAGES OF EXISTING SYSTEM

The existing systems for Chronic Kidney Disease (CKD) detection and prediction face several notable limitations that hinder their effectiveness in early diagnosis and clinical adoption. Traditional diagnostic methods, such as serum creatinine measurement and estimated glomerular filtration rate (eGFR), typically detect CKD only after significant kidney damage has occurred. This delayed identification is problematic because CKD is often asymptomatic in its early stages, leading to many cases being diagnosed only at advanced stages when treatment options are limited to dialysis or transplantation. As a result, opportunities for preventive intervention, lifestyle modification, and early pharmacological management are frequently missed, worsening patient prognosis and increasing healthcare burdens.

In the context of machine learning (ML)-based approaches, many existing systems rely on a single classifier, such as Support Vector Machines (SVM), Random Forest (RF), or Artificial Neural Networks (ANN). While these models can achieve high accuracy in controlled studies, they often suffer from instability when applied to diverse datasets, due to issues like overfitting, underfitting, or susceptibility to noisy inputs. Another critical drawback is the lack of interpretability—many high-performing models operate as "black boxes," providing predictions without explaining the reasoning behind them. In clinical environments, where decisions directly impact patient safety, such opacity reduces trust among healthcare professionals and poses challenges for regulatory acceptance.

Moreover, existing systems often struggle with imbalanced datasets, where positive CKD cases are much fewer than negative ones, leading to biased predictions and reduced sensitivity to early-stage disease. Many approaches also fail to integrate seamlessly with Electronic Health Records (EHR) or real-time clinical decision support systems, limiting their usability in actual healthcare workflows. Additionally, traditional detection systems and some ML-based models are not optimized for resource-limited settings, as they depend heavily on costly laboratory tests or specialized diagnostic tools, restricting their scalability for large-scale screening. Combined, these disadvantages highlight the pressing need for a more robust, interpretable, and scalable CKD prediction framework that addresses both performance and transparency while ensuring accessibility across diverse healthcare environments.

3.2 PROPOSED SYSTEM

The proposed system aims to address the limitations of existing Chronic Kidney Disease (CKD) detection methods by introducing an interpretable, ensemble-based intelligence framework capable of providing early and accurate prognosis. Unlike conventional diagnostic techniques, this system leverages a combination of machine learning algorithms and feature selection methods to analyze patient data more comprehensively. By integrating multiple models, the framework enhances predictive accuracy, reduces bias, and ensures that the decision-making process remains transparent for healthcare professionals.

The architecture is designed to process diverse clinical parameters, such as laboratory test results, demographic details, and patient history, to detect early patterns indicative of CKD progression. Advanced preprocessing techniques, including data cleaning, normalization, and handling of missing values, are employed to ensure high-quality input for the predictive models. Furthermore, the use of interpretable AI techniques allows clinicians to understand the key factors influencing predictions, promoting trust and facilitating evidence-based medical decisions.

To ensure practicality, the proposed system incorporates scalability, interoperability, and integration capabilities with existing healthcare infrastructures. This enables seamless adoption in clinical settings, where it can be used not only for individual risk assessment but also for population-level screening. By detecting CKD at earlier stages, the system supports timely interventions, lifestyle recommendations, and pharmacological strategies that can slow disease progression, improve patient outcomes, and reduce healthcare costs.

3.3 FEASIBILITY STUDY

The feasibility study for the proposed Chronic Kidney Disease (CKD) detection system evaluates its practicality and effectiveness before implementation in a clinical environment. This assessment considers multiple dimensions, including technical, economic, operational, and legal feasibility. Technically, the system leverages established machine learning frameworks and interpretable ensemble models, which can be readily deployed on standard healthcare IT infrastructure. The required hardware specifications are minimal, and the algorithms are optimized for efficiency, ensuring compatibility with existing hospital information systems without the need for extensive upgrades.

From an economic perspective, the development and deployment costs are justified by the long-term benefits of early disease detection. By identifying CKD in its early stages, the system can significantly reduce expenses related to late-stage treatments such as dialysis and transplantation. This results in substantial savings for both healthcare providers and patients. Additionally, the open-source availability of several machine learning tools and cloud-based processing options helps minimize software licensing and maintenance costs.

Operational feasibility is ensured through the system's user-friendly interface and integration with electronic health records (EHRs), allowing medical professionals to access predictive insights without disrupting existing workflows. The system also supports automated report generation, which aids in quick decision-making and reduces the administrative burden on healthcare staff. Furthermore, data privacy and security measures, such as encryption and compliance with healthcare regulations like HIPAA, guarantee legal and ethical feasibility. Overall, the feasibility study confirms that the proposed CKD detection framework is practical, sustainable, and capable of delivering significant clinical and economic value in real-world healthcare settings.

3.3.1 Technical Feasibility

The proposed CKD detection system is technically feasible as it uses readily available machine learning tools, standard hardware, and cloud deployment options. It integrates seamlessly with Electronic Health Records, supports automated data processing, and ensures real-time, accurate predictions without requiring high-cost infrastructure, making implementation practical and efficient.

3.3.2 Operational Feasibility

The proposed system is operationally feasible as it aligns with existing healthcare workflows, requires minimal training for medical staff, and offers a user-friendly interface. Its explainable AI features build clinician trust, while automated reporting and seamless integration with hospital databases ensure smooth adoption and effective use in real-world clinical settings.

3.3.3 Economic Feasibility

The proposed system is economically feasible as it leverages cost-effective computational resources and open-source machine learning frameworks, minimizing development and maintenance expenses. Early CKD detection reduces long-term treatment costs, such as dialysis and transplantation, leading to significant healthcare savings while delivering high diagnostic value for both patients and healthcare providers.

3.3.4 Legal and Ethical Feasibility

The proposed system is legally and ethically feasible as it complies with healthcare data protection regulations such as HIPAA and GDPR, ensuring patient confidentiality and secure data handling. Ethical considerations include transparency through explainable AI, bias mitigation in predictions, and informed consent for data usage, promoting trust and fairness in clinical decision-making.

3.4 USING COCO MODEL:

The training and evaluation phase represents a critical stage in the CKD prediction framework, where the preprocessed and optimized dataset is utilized to build reliable classifiers. During model training, multiple algorithms—including Logistic Regression, Support Vector Machines, Random Forests, K-Nearest Neighbors, XGBoost, Naïve Bayes, and Artificial Neural Networks—are applied independently to the same dataset. Each model is trained to learn underlying patterns that distinguish CKD patients from non-CKD individuals, capturing both linear and non-linear relationships between the features and the target variable. Hyperparameter tuning is employed using techniques such as grid search or cross-validation to optimize model parameters and enhance generalization capability. Once trained, the models undergo a systematic evaluation process using widely accepted performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC. These metrics provide a balanced view of model performance, particularly in addressing sensitivity (correctly identifying CKD cases) and specificity (avoiding false positives). In addition, k-fold cross-validation is often implemented to ensure the robustness and stability of results across different dataset partitions, minimizing overfitting risks. Comparative evaluation allows for the identification of the best-performing models and provides insights into their strengths and limitations. This stage not only facilitates the benchmarking of individual classifiers but also creates a foundation for advanced ensemble methods, where multiple models can be combined to achieve superior predictive accuracy, stability, and clinical applicability.

4.SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

Operating System: Windows 10/11 or Ubuntu 20.04+ (Recommended for better compatibility with machine learning libraries)

Programming Language: Python 3.9+

Development Environment: Jupyter Notebook / PyCharm / VS Code / Google Colab

Libraries & Dependencies:

Machine Learning: scikit-learn, XGBoost

Deep Learning (if required): TensorFlow 2.x / PyTorch

Data Processing & Visualization: NumPy, Pandas, Matplotlib, Seaborn

Hyperparameter Tuning: GridSearchCV, Optuna

Model Interpretability: SHAP, LIME

4.2 REQUIREMENT ANALYSIS

The requirement analysis for the project "Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis" involves identifying both functional and non-functional requirements essential for the successful implementation of the system. Functionally, the dataset from the UCI Machine Learning Repository undergoes preprocessing steps to handle missing values, normalize attributes, and balance the target classes using SMOTE. Key predictive attributes are selected through statistical measures such as ANOVA F-tests and mutual information scores, ensuring the most relevant features are retained for the model.

The system implements an ensemble approach by combining models such as Logistic Regression, Random Forest, Support Vector Machine, and Gradient Boosting to improve predictive accuracy and reduce bias. Hyperparameter optimization is carried out through GridSearchCV and Optuna, ensuring the best model configuration is used. To address the critical need for interpretability in medical decision-making, SHAP and LIME are incorporated, allowing clinicians to understand the rationale behind predictions and fostering trust in the system.

From a non-functional standpoint, the solution is designed to be platform-independent, allowing deployment on both Windows and Linux environments. The modular design ensures scalability, enabling the addition of new models and data sources without disrupting the existing workflow. Performance optimization ensures that predictions are generated in real time, making the system viable for integration with hospital electronic health record (EHR) systems. Robustness and security measures are incorporated to protect sensitive medical data while maintaining compliance with healthcare data privacy regulations.

4.3 HARDWARE REQUIREMENTS

Processor	Intel Core i5 (13th Gen) or AMD Ryzen 5/7 (Equivalent)
RAM	Minimum 8GB (Recommended: 16GB for faster model training)
STORAGE	Minimum 256GB SSD (Recommended: 512GB SSD for faster data handling)
GPU	NVIDIA GTX 1650 / RTX 3050 or equivalent (for deep learning models)
DISPLAY	Full HD (1920×1080) for clear visualization of results

4.4 SOFTWARE

Several software solutions can be developed based on the Chronic Kidney Disease prediction project. A CKD Prediction App can allow users to input health data and receive risk predictions using models like Decision Tree, Random Forest, and Logistic Regression. A Clinical Decision Support System (CDSS) for hospitals can store patient history, analyze records, and provide risk predictions using advanced models like XGBoost, with confidence scores and detailed explanations. A CKD Monitoring API can be integrated into healthcare platforms to offer real-time predictions based on patient data. A Health Data Visualization Dashboard can display patient data and model performance through charts, heatmaps, and interactive reports. Lastly, a Wearable Device Integration solution can collect real-time health data from smartwatches and provide early warnings for CKD risk, supporting timely medical intervention.

4.5 SOFTWARE DESCRIPTION

The software developed for this project is a comprehensive system designed to predict the likelihood of Chronic Kidney Disease (CKD) in patients using advanced machine learning techniques. It provides an end-to-end solution, handling everything from data preprocessing and feature selection to model training, prediction, and visualization, making it highly efficient for healthcare applications. The system is built with Python and incorporates powerful libraries such as scikit-learn, pandas, NumPy, and TensorFlow, ensuring both performance and reliability.

The data preprocessing module is a key component that prepares raw patient data for analysis. It handles missing values, encodes categorical attributes, and normalizes numerical features, thereby improving the accuracy and efficiency of the machine learning models. Proper preprocessing ensures that the system can effectively handle large and complex datasets while maintaining data integrity.

The feature selection module further enhances prediction accuracy by identifying the most relevant attributes that contribute to CKD detection. Techniques such as Recursive Feature Elimination (RFE) and variance thresholding are applied to remove redundant or less significant features, thereby reducing computational complexity and improving model performance.

The machine learning module allows the system to train and evaluate various predictive models, including Logistic Regression, Random Forest, Support Vector Machines (SVM), and Artificial Neural Networks (ANN). To address class imbalance in medical datasets, techniques like SMOTE (Synthetic Minority Oversampling Technique) are integrated, ensuring fair representation of all patient categories. Users can evaluate model performance using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

The prediction and visualization module enables healthcare professionals to input patient data and receive immediate CKD risk predictions. Results are presented through intuitive charts, graphs, and tables, which make it easy to interpret trends and monitor patient progress.

Designed for both usability and scalability, the software features a simple user interface that does not require technical expertise. Its modular architecture allows future enhancements, including cloud deployment, real-time monitoring, and mobile integration.

Overall, the software provides a reliable, efficient, and interpretable solution for early CKD detection, supporting healthcare professionals in making proactive and informed decisions to improve patient outcomes.

5.SYSTEM DESIGN

The Chronic Kidney Disease (CKD) prediction system is designed to improve diagnostic accuracy and early detection using machine learning (ML). The system begins with data collection from the CKD dataset, which contains 400 patient records consisting of both CKD-affected and healthy individuals. The collected data is preprocessed to handle missing values, normalize numerical features, and encode categorical variables. Demographic imbalances, particularly the disparity between male and female patients, are addressed using SMOTE (Synthetic Minority Oversampling Technique) to ensure balanced class distribution. A stratified split is employed to maintain similar demographic distributions in both training and testing sets, enhancing the model's generalization capabilities.

Feature selection is performed using Principal Component Analysis (PCA) and statistical techniques to identify the most significant attributes affecting CKD prediction. This selection reduces dimensionality, mitigates overfitting, and improves model generalization. The system implements multiple ML algorithms, including Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Naive Bayes, Gradient Boosting, Extreme Gradient Boosting (XGBoost), and Multi-layer Perceptron (MLP). Hyperparameter tuning is applied to optimize model performance, and the models are evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Training and testing are conducted using an 80/20 split, and K-fold cross-validation ensures robustness and prevents overfitting. The results indicate that Random Forest, Decision Tree, and KNN models achieve the highest accuracy, with performance metrics exceeding 95%, demonstrating reliable predictive capabilities for early CKD detection.

This structured system design highlights the effectiveness of ML-based approaches for CKD prediction, providing healthcare professionals with a reliable and interpretable tool for early diagnosis, improved patient management, and informed clinical decision-making.

5.1 SYSTEM ARCHITECTURE

The architecture illustrated in Fig. 1 begins with dataset acquisition, which provides the foundation for the predictive pipeline. Since the reliability of any machine learning model is closely tied to the quality of the input data, the acquired dataset undergoes a thorough preprocessing stage to ensure consistency and accuracy. Initially, categorical variables are transformed into numerical values using techniques such as label encoding or one-hot encoding so that they can be properly handled by machine learning algorithms.

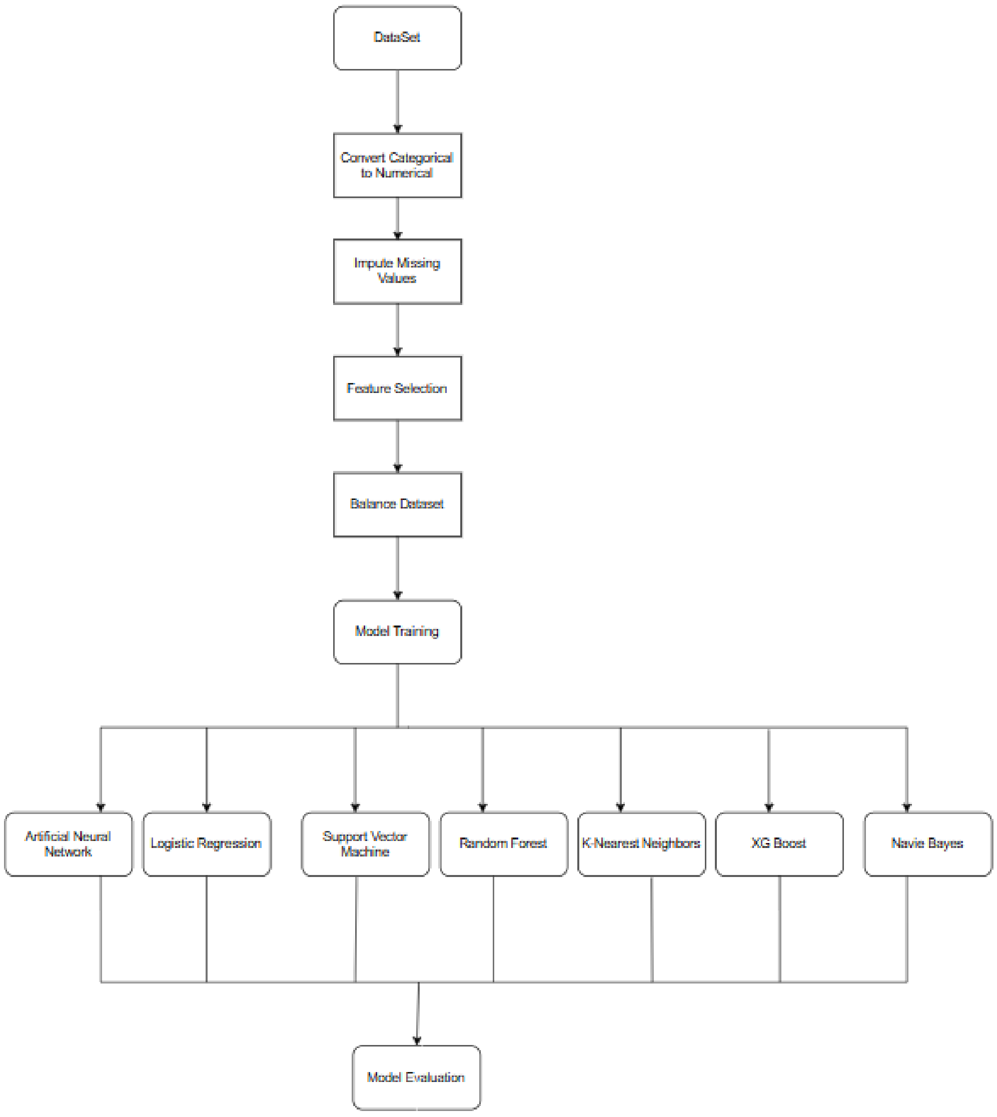


Fig 1: System Architecture

Following this, missing values—common in healthcare datasets due to incomplete records or unavailable test results—are addressed through suitable imputation strategies, ranging from simple mean or median substitution to advanced multivariate methods, thereby minimizing data loss and avoiding statistical skewness. Once the dataset is complete, feature selection is performed to remove redundant or weakly informative attributes, ensuring that the most relevant predictors of CKD are retained. This step improves model efficiency, interpretability, and accuracy while reducing computational complexity. Another critical challenge addressed during preprocessing is class imbalance, where the number of non-CKD cases significantly outweighs CKD cases. To counter this, resampling methods such as the Synthetic Minority Oversampling Technique (SMOTE) are applied, ensuring a balanced distribution that enhances the sensitivity of the models to minority-class cases without inflating false positives. The optimized dataset is then used to train a diverse set of classification algorithms, including Artificial Neural Networks, Logistic Regression, Support Vector Machines, Random Forests, K-Nearest Neighbors, XGBoost, and Naïve Bayes. Each of these models is trained independently to capture different aspects of the data, ranging from linear patterns to complex non-linear interactions. After training, their performance is evaluated using standard metrics such as accuracy, precision, recall, F1-score, and AUC-ROC, providing a comprehensive assessment of their classification capabilities. This comparative analysis not only highlights the strengths and weaknesses of each model but also lays the groundwork for ensemble learning approaches, where predictions from multiple classifiers can be combined to achieve greater accuracy, robustness, and clinical applicability.

5.2 MODULES

The project document includes various modules that outline the key components and Methodologies involved in kidney disease prediction using machine learning. The Modules include:

1. **Dataset Description** - There are 400 patient records containing 25 clinical and demographic characteristics pertinent to the diagnosis of chronic kidney disease (CKD) comprise the dataset used in this investigation. Blood pressure, specific gravity, albumin, blood sugar, cell counts, serum creatinine, hemoglobin, and electrolytes are important markers. 250 CKD-positive and 150 CKD-negative cases are identified by the binary outcome variable ('classification'), which indicates a slight class imbalance common in clinical data. Imputation based on feature-wise linear regression is used to handle missing values. The development of interpretable machine learning models for early CKD detection is supported by this dataset, which combines numerical and categorical variables with realistic clinical imperfections.

2. **Data Preprocessing** - Before modeling, we used a robust preprocessing pipeline to enhance data quality, prevent biases, and improve interpretability. To start with, categorical variables were label encoded, converting string labels to integer values to facilitate model input. Missing values were then filled in using feature-wise linear regression, with every incomplete variable regressed against all the others to maintain inherent relationships—an approach supported by cluster-wise linear models in medical data. We then removed low-variance features through a Variance Threshold of 0.75, discarding constant or near-constant predictors that contribute to noise and hinder generalization. Recursive Feature Elimination (RFE) was applied using a logistic regression base estimator to select the most informative features, reducing dimensionality and enhancing model specificity by retaining the top six predictors. In acknowledgment of the widespread class imbalance issue in medical datasets, we implemented a dynamic SMOTE procedure. For minority classes with adequate sample sizes, we applied SMOTE with kneighbors =min (5, minority size 1) to synthetically rebalance the data. However, if a minority class contained fewer than two samples, SMOTE was skipped to avoid generating unreliable synthetic instances. Finally, our pipeline incorporated explainable AI (XAI) modules to enhance transparency and clinical trust. Local Interpretable Model-agnostic Explanations (LIME) was used for individual prediction interpretability, while Shapley Additive explanations (SHAP) provided both global and local feature importance analyses. Collectively, these preprocessing strategies—label encoding, regression-based imputation, variance filtering, RFE selection, SMOTE balancing, and XAI modules—formed a strong foundation for accurate and interpretable CKD prognosis modeling. The Fig2, shows the number of missing values per feature within your CKD dataset—both prior to and after performing your imputation technique. Each feature (e.g., age, bp, sg, al, sod, pot, hemo, etc.) is represented on the X-axis, whereas the Y-axis indicates the number of missing entries in each. The blue bars represent the missing values prior to preprocessing, obviously showing that numerous features had huge gaps—for example, sodium (sod) and potassium (pot) each had more than 80 missing values, while sg, al, su, bu, sc, and hemo had missing counts from around 10 to 50+. Conversely, there are no orange bars (indicating missing

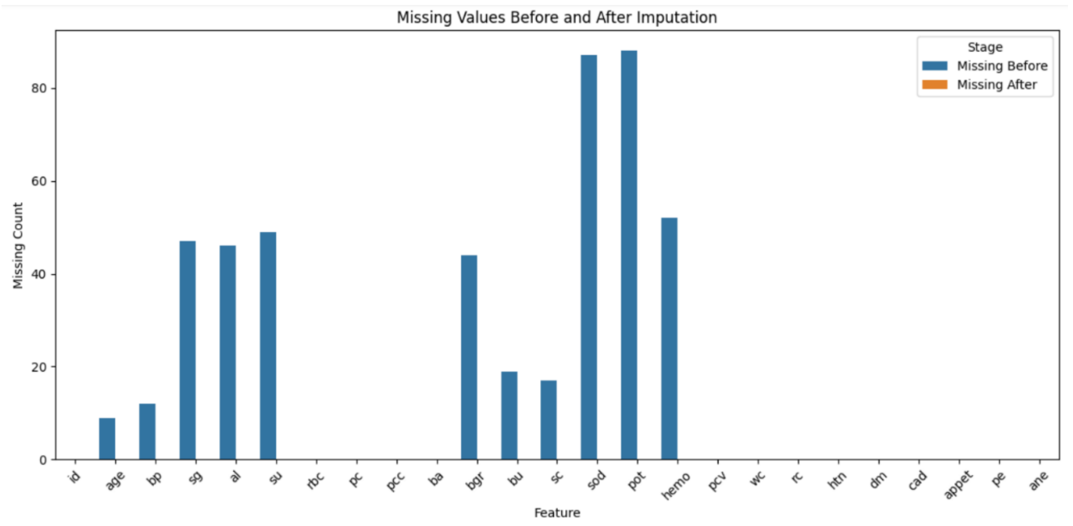


Fig 2: Missing Values Before And After Imputation.X axis: Features; Y-axis: Number of Missing Values.

values post imputation) present, which means all missing values were appropriately filled during preprocessing.

3. **Feature Engineering -** All the categorical variables were translated to numeric representation through Label Encoding, where every distinct categorical category was mapped to an integer index. This conversion allowed for the application of algorithms that need exclusively numeric input, e.g., logistic regression and tree-based ensemble models. Although not necessarily in code, it is possible to optionally design a flag variable for every feature with a marker of whether or not its value was imputed. These types of indicators can give models useful signals regarding missingness patterns in the data, which may be associated with underlying health. Although our existing implementation did not use interaction terms, combinations specific to the domain—such as multiplication or ratios of critical biomarkers (e.g., serum creatinine × blood urea nitrogen)—may make it easier for the model to learn to describe non-linear interactions. These engineered features are typically useful in clinical datasets marked by intercorrelated clinical variables.

4. **Feature Selection -** We then used Variance Threshold (with threshold = 0.75) to drop constant or quasi-constant features, which generally have little to no information value. It is an easy but useful trick since low-variance features are not

likely to contribute significantly to the decision boundary, and dropping them simplifies model training and prevents unnecessary over- complication. After variance filtering, we used Recursive Feature Elimination (RFE) with the base estimator being logistic regression. RFE progressively fits a model and removes the least important feature at every iteration, repeating until the required number of features (six in our scenario) is achieved. This wrapper strategy guarantees retained features contribute most to the model performance and interpretability. In particular, we train an RFE selector to retain the top six predictors, which will allow us to select and retain the most significant variables for CKD prognosis. The outcome is a highly condensed feature set that trades off parsimony with prediction.

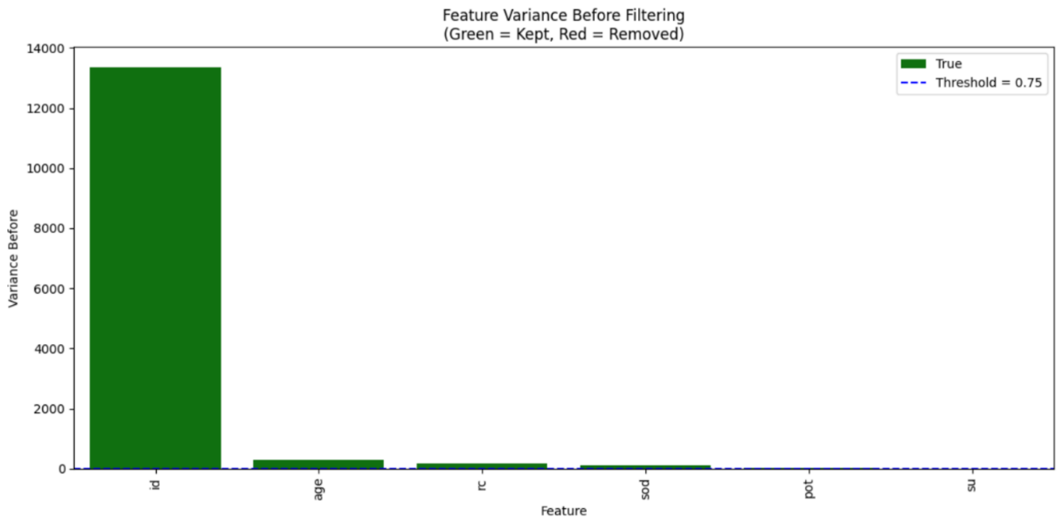


Fig. 3: Feature Variance Before Filtering.X-axis: Feature Name; Y-axis: Variance Value

5. **Model Training and Evaluation -** The training and evaluation phase represents a critical stage in the CKD prediction framework, where the preprocessed and optimized dataset is utilized to build reliable classifiers. During model training, multiple algorithms—including Logistic Regression, Support Vector Machines, Random Forests, K-Nearest Neighbors, XGBoost, Naïve Bayes, and Artificial Neural Networks—are applied independently to the same dataset. Each model is trained to learn underlying patterns that distinguish CKD patients from non-CKD individuals, capturing both linear and non-linear relationships between the features and the target variable. Hyperparameter tuning is employed using techniques such as grid search or cross-validation to optimize model parameters and enhance generalization capability.

Once trained, the models undergo a systematic evaluation process using widely accepted performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC. These metrics provide a balanced view of model performance, particularly in addressing sensitivity (correctly identifying CKD cases) and specificity (avoiding false positives). In addition, k-fold cross-validation is often implemented to ensure the robustness and stability of results across different dataset partitions, minimizing overfitting risks. Comparative evaluation allows for the identification of the best-performing models and provides insights into their strengths and limitations. This stage not only facilitates the benchmarking of individual classifiers but also creates a foundation for advanced ensemble methods, where multiple models can be combined to achieve superior predictive accuracy, stability, and clinical applicability.

5.3 UML DIAGRAMS

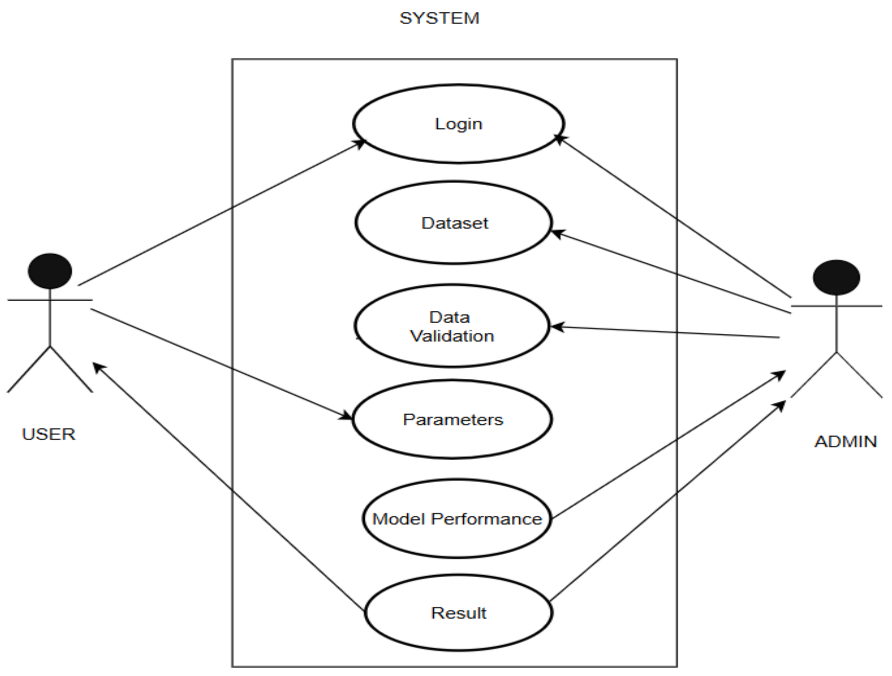


Fig 4: UML Diagram

The UML (Unified Modeling Language) diagram provides a visual representation of the interactions between different actors and the CKD prediction system. It illustrates how the User and the Admin interact with the system and perform essential tasks. The use case diagram clearly identifies the functionalities required, ensuring a structured view of the system's behavior. In this project, both User and Admin are primary actors. The User interacts with the system to log in, upload or access the dataset, and view the prediction results. The Admin performs additional responsibilities such as dataset management, data validation to maintain accuracy, fine-tuning model parameters, and evaluating model performance to ensure reliability. Both actors can eventually view the results, which provide early indications of chronic kidney disease risk. The UML diagram emphasizes how the CKD prediction system integrates multiple processes, from dataset acquisition and preprocessing to model training and result generation. It highlights the collaborative role of the Admin in maintaining data integrity and model efficiency, and the role of the User in accessing predictions. This structured approach enhances the clarity of system requirements and ensures smooth interaction between components, supporting accurate and interpretable CKD diagnosis.

6.IMPLEMENTATION

6.1 MODEL IMPLEMENTATION

The implementation of the proposed framework, *Echoes of the Hidden Filter*, was carefully structured to balance predictive performance with clinical interpretability. The process began with the acquisition of a CKD dataset comprising 400 patient records and 25 clinical and demographic features. These features included vital biomarkers such as serum creatinine, blood pressure, haemoglobin, sodium, potassium, and specific gravity, which play a crucial role in early CKD diagnosis. To address data quality issues, a robust preprocessing pipeline was applied. Categorical attributes were label-encoded, while missing values were imputed using feature-wise regression methods to maintain inherent correlations between predictors. Low-information features were eliminated using a variance threshold, and Recursive Feature Elimination (RFE) with logistic regression was applied to retain the six most significant predictors. To overcome class imbalance between CKD-positive and CKD-negative cases, a dynamic SMOTE strategy was used, generating synthetic samples for adequately represented minority cases while avoiding unreliable augmentation in highly sparse conditions. These preprocessing steps ensured data consistency, improved generalization, and prepared a reliable foundation for model training.

Following data optimization, the prepared dataset was used to train a heterogeneous set of machine learning models, including Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost, K-Nearest Neighbors (KNN), Naïve Bayes, and Artificial Neural Networks (ANNs). Each model was rigorously validated using 10-fold stratified cross-validation and hold-out testing to minimize overfitting and evaluate robustness. Performance was measured using multiple metrics such as accuracy, precision, recall, and F1-score, all of which exceeded 95%. Among the models, Random Forest and XGBoost demonstrated superior accuracy (approaching 99%), confirming their effectiveness in capturing complex, non-linear biomarker relationships. Beyond predictive accuracy, the framework incorporated interpretability modules to enhance clinical adoption: SHAP was employed to provide global insights into biomarker contributions, while LIME explained individual predictions, enabling clinicians to understand why a particular decision was made.

Additionally, visual diagnostics—such as missingness heatmaps, class balance charts, variance distributions, and confusion matrix heatmaps—were used throughout the pipeline to validate preprocessing steps and ensure reproducibility. Collectively, these steps produced a high-performing, transparent, and clinically relevant AI-based solution, bridging the gap between algorithmic prediction and real-world medical decision-making in early CKD prognosis.

6.2 CODING

```
#Prompt: mount the google drive
from google.colab import drive
drive.mount('/content/drive')
#Prompt: load the dataset from drive
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
df = pd.read_csv("/content/drive/MyDrive/Project/kidney_disease.csv")
#Prompt: list the data from the dataset
df
#Prompt: import all libraries
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import StratifiedKFold
#Prompt: Convert categorical to numerical using Label Encoding
label_encoders = {}
for col in df.select_dtypes(include='object'):
   le = LabelEncoder()
   df[col] = df[col].astype(str)
   df[col] = le.fit_transform(df[col])
```

```
    label_encoders[col] = le
#Prompt: Separate features and target
X = df.drop('classification', axis=1)
y = df['classification']
#Prompt: list ckd and non ckd patients
# Normalize column and value formatting
df.columns = df.columns.str.strip().str.lower()
# Decode 'classification' back to string before applying string methods
df['classification'] =
label_encoders['classification'].inverse_transform(df['classification'])
df['classification'] = df['classification'].str.strip().str.lower()
# Filter CKD and non-CKD patients
ckd_patients = df[df['classification'] == 'ckd']
non_ckd_patients = df[df['classification'] == 'notckd']
# Print counts
print("Number of CKD (disease) patients:", len(ckd_patients))
print("Number of non-CKD (non-disease) patients:", len(non_ckd_patients))
# Print their details
print("\n--- CKD Patients ---")
print(ckd_patients)
print("\n--- Non-CKD Patients ---")
print(non_ckd_patients)
#Prompt: Detect missing values
missing_features = X.columns[X.isnull().any()].tolist()
#Prompt: Impute missing values using feature-wise linear regression
ef impute_missing_values(df):
    df_imputed = df.copy()
    for col in df.columns:
        if df[col].isnull().sum() > 0:
            known = df[df[col].notnull()]
            unknown = df[df[col].isnull()]
            if unknown.shape[0] == 0 or known.shape[0] == 0:
                continue
            y_train = known[col]
            X_train = known.drop(columns=[col])
            X_pred = unknown.drop(columns=[col])

            # Impute missing values in X_train and X_pred
            imputer = SimpleImputer(strategy='mean')
            X_train_imputed = imputer.fit_transform(X_train)
            X_pred_imputed = imputer.transform(X_pred)

            model = LinearRegression()
            model.fit(X_train_imputed, y_train)
            df_imputed.loc[unknown.index, col] = model.predict(X_pred_imputed)
    return df_imputed

X_original = X.copy() # Save the original X before imputation
X = impute_missing_values(X)
#Prompt: Missing Values graph before and after imputation
import seaborn as sns
import matplotlib.pyplot as plt # Import matplotlib

def plot_missing_values(df_before, df_after):
    missing_before = df_before.isnull().sum()
    missing_after = df_after.isnull().sum()

    missing_df = pd.DataFrame({
        'Feature': df_before.columns,
        'Missing Before': missing_before.values,
        'Missing After': missing_after.values
    })

    missing_df = missing_df.melt(id_vars='Feature',
                                 var_name='Stage',
                                 value_name='Missing Count')

    plt.figure(figsize=(12, 6))
    sns.barplot(data=missing_df, x='Feature', y='Missing Count', hue='Stage')
    plt.title('Missing Values Before and After Imputation')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Call this with your before and after DataFrames
plot_missing_values(X_original, X)
#prompt: Remove constant and quasi-constant features
selector = VarianceThreshold(threshold=0.75)
X_reduced = selector.fit_transform(X)
selected_columns = X.columns[selector.get_support(indices=True)]
X = pd.DataFrame(X_reduced, columns=selected_columns)
#prompt:  Recursive Feature Elimination (Wrapper)
log_reg = LogisticRegression(max_iter=1000)
rfe = RFE(log_reg, n_features_to_select=6)
rfe.fit(X, y)
X = X.loc[:, rfe.support_]
#prompt: graph for feature variance before filtering
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
def plot_variance_before_after(X_before, threshold=0.75):
    # Compute variance before filtering
    variance_before = X_before.var().sort_values(ascending=False)
    # Apply VarianceThreshold
    selector = VarianceThreshold(threshold=threshold)
    X_reduced = selector.fit_transform(X_before)
    selected_columns = X_before.columns[selector.get_support(indices=True)]
    X_after = pd.DataFrame(X_reduced, columns=selected_columns)

    # Compute variance after filtering
    variance_after = X_after.var()
    # Create DataFrame for plotting
    var_df = pd.DataFrame({
        'Feature': variance_before.index,
        'Variance Before': variance_before.values,
        'Kept After Filtering': variance_before.index.isin(selected_columns)
    })
    # Plot
    plt.figure(figsize=(12, 6))
    sns.barplot(data=var_df, x='Feature', y='Variance Before', hue='Kept After
Filtering', dodge=False, palette={True: 'green', False: 'red'})
    plt.axhline(y=threshold, color='blue', linestyle='--', label=f'Threshold =
{threshold}')
    plt.title('Feature Variance Before Filtering\n(Green = Kept, Red = Removed)')
    plt.xticks(rotation=90)
    plt.legend()
    plt.tight_layout()
    plt.show()
    return X_after
# Example usage:
X_original = X.copy()  # Before filtering
X = plot_variance_before_after(X_original, threshold=0.75)
#prompt: Balance dataset using SMOTE
# Check the class distribution
class_counts = y.value_counts()
min_class_size = class_counts.min()
if min_class_size >= 2:
    # Set n_neighbors to the minimum required if the minority class size is between 2
and 5
    n_neighbors = min(5, min_class_size -1) if min_class_size > 1 else 1
    smote = SMOTE(random_state=42, k_neighbors=n_neighbors)
    X_bal, y_bal = smote.fit_resample(X, y)
else:
    print(f"Skipping SMOTE as the minority class has only {min_class_size}
sample(s).")
```

```
    X_bal = X
    y_bal = y
#prompt: Prepare Stratified K-Fold
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
for fold, (train_index, test_index) in enumerate(skf.split(X_bal, y_bal)):
    print(f"Fold {fold+1}: Train size = {len(train_index)}, Test size =
{len(test_index)}")
#prompt: Final dataset
df
#prompt: using lime
import lime
import lime.lime_tabular
import numpy as np
# Initialize the explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=np.array(X_train),
    feature_names=X_train.columns,
    class_names=[str(c) for c in np.unique(y_train)],
    mode='classification'
)
# Pick an instance to explain
i = 0  # index of sample
exp = explainer.explain_instance(
    data_row=X_test.iloc[i],
    predict_fn=rf_model.predict_proba
)
# Show explanation
exp.show_in_notebook(show_table=True)
#prompt: using shap
import shap
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
# Step 1: Split data
X_train, X_test, y_train, y_test = train_test_split(
   X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Step 2: Train model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
# Step 3: SHAP Explainer
explainer = shap.Explainer(rf_model, X_train)
# Step 4: Compute SHAP values
shap_values = explainer(X_test)
# Step 5: Determine if binary or multiclass
num_classes = len(np.unique(y_train))
instance_index = 0
print(f"\nExplaining instance {instance_index}...")
# Step 6: Plot Waterfall
if num_classes == 2:
   # Binary case – works directly
   shap.plots.waterfall(shap_values[instance_index])
else:
   # Multiclass case – must manually index class and build explanation
   class_index = 0  # You can change to 1, 2, etc. for other classes
   # Build a single SHAP explanation for the selected instance and class
   single_expl = shap.Explanation(
      values=shap_values.values[instance_index, class_index],
      base_values=shap_values.base_values[instance_index, class_index],
      data=shap_values.data[instance_index],
      feature_names=shap_values.feature_names
   )
   shap.plots.waterfall(single_expl)
#prompt:  Logistic Regression without k-fold
import warnings
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, random_state=42, stratify=y_bal
)
#  Initialize and train logistic regression model
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train, y_train)
#  Make predictions
y_pred = logreg.predict(X_test)
y_prob = logreg.predict_proba(X_test)[:, 1]  # For ROC-AUC
#  Evaluate performance
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print("Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", round(roc_auc_score(y_test, logreg.predict_proba(X_test),
multi_class='ovr'), 4))
#  Confusion Matrix
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: Logistic Regression with k-fold
import warnings
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    roc_auc_score,
    confusion_matrix
)
# Optional: ignore warnings
warnings.filterwarnings("ignore")
# Split the data
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, random_state=42, stratify=y_bal
)
# Initialize Logistic Regression
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train, y_train)
# Predictions on test set
y_pred = logreg.predict(X_test)
y_prob = logreg.predict_proba(X_test)[:, 1]
# --- Evaluation Metrics ---
print("Classification Report:")
print(classification_report(y_test, y_pred))
print("Accuracy:", round(accuracy_score(y_test, y_pred), 4))
# Handle binary vs multi-class for ROC-AUC
if len(np.unique(y_test)) > 2:
    roc_auc = roc_auc_score(y_test, logreg.predict_proba(X_test), multi_class='ovr')
else:
    roc_auc = roc_auc_score(y_test, y_prob)
print("ROC-AUC:", round(roc_auc, 4))
# --- Confusion Matrix Heatmap ---
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()
# --- 10-Fold Cross-Validation (Training Accuracy) ---
cv_scores = cross_val_score(logreg, X_train, y_train, cv=10, scoring='accuracy')
```

```
train_acc = np.mean(cv_scores)
test_acc = accuracy_score(y_test, y_pred)
print(f"\n10-Fold CV Training Accuracy: {train_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
# --- Plot Training vs Test Accuracy ---
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc], palette="viridis")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.title("Training vs Testing Accuracy LR")
plt.tight_layout()
plt.show()
#prompt: Support Vector Machine with out k-fold
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
#train-test split
X_train, X_test, y_train, y_test = train_test_split(
   X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# SVM with linear kernel and probability output enabled
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train, y_train)
# Predict labels and probabilities
y_pred = svm_model.predict(X_test)
y_prob = svm_model.predict_proba(X_test) # For ROC-AUC with multi_class
# Performance metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", round(roc_auc_score(y_test, y_prob, multi_class='ovr'), 4))
#  Confusion Matrix
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: Support Vector Machine with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
# Step 1: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Step 2: Initialize and train SVM (linear kernel)
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train, y_train)
# Step 3: Predictions
y_pred = svm_model.predict(X_test)
y_prob = svm_model.predict_proba(X_test)  # Needed for ROC-AUC
# Step 4: Evaluation Metrics
print("Classification Report:")
print(classification_report(y_test, y_pred))
test_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", round(test_accuracy, 4))
# ROC-AUC (binary or multiclass)
if len(np.unique(y_test)) > 2:
    roc_auc = roc_auc_score(y_test, y_prob, multi_class='ovr')
else:
    roc_auc = roc_auc_score(y_test, y_prob[:, 1])
print("ROC-AUC:", round(roc_auc, 4))
# Step 5: Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Confusion Matrix - SVM')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()
# Step 6: Cross-Validation Accuracy (10-Fold)
cv_scores = cross_val_score(svm_model, X_train, y_train, cv=10,
scoring='accuracy')
train_accuracy = np.mean(cv_scores)
print(f"10-Fold CV Training Accuracy: {train_accuracy:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
# --- Plot Training vs Test Accuracy ---
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc], palette="viridis")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.title("Training vs Testing Accuracy SVM")
plt.tight_layout()
plt.show()
#prompt: Random Forest with out k-fold
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
# train-test split
X_train, X_test, y_train, y_test = train_test_split(
   X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Define and train the model
rf_model = RandomForestClassifier(
   n_estimators=100,     # Number of trees
   max_depth=None,       # Let trees expand fully
   random_state=42
```

```
)
rf_model.fit(X_train, y_train)
# Predict on test set
y_pred = rf_model.predict(X_test)
y_prob = rf_model.predict_proba(X_test) # For ROC-AUC
# Display classification metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", round(roc_auc_score(y_test, y_prob, multi_class='ovr'), 4))
# Confusion matrix
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: Random Forest with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import (
   classification_report,
   accuracy_score,
   roc_auc_score,
   confusion_matrix
)
# 1. Train-test split
X_train, X_test, y_train, y_test = train_test_split(
   X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# 2. Random Forest model
rf_model = RandomForestClassifier(
   n_estimators=100,
   max_depth=None,
   random_state=42
```

```
)
rf_model.fit(X_train, y_train)
# 3. Predictions
y_pred = rf_model.predict(X_test)
y_prob = rf_model.predict_proba(X_test)
# 4. Evaluation Metrics
print("Classification Report:")
print(classification_report(y_test, y_pred))
test_acc = accuracy_score(y_test, y_pred)
print("Accuracy:", round(test_acc, 4))
# Multiclass or binary ROC-AUC
if len(np.unique(y_test)) > 2:
    roc_auc = roc_auc_score(y_test, y_prob, multi_class='ovr')
else:
    roc_auc = roc_auc_score(y_test, y_prob[:, 1])
print("ROC-AUC:", round(roc_auc, 4))
# 5. Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
# 6. 10-Fold Cross-Validation Accuracy
cv_scores = cross_val_score(rf_model, X_train, y_train, cv=10, scoring='accuracy')
train_acc = np.mean(cv_scores)
print(f"10-Fold CV Training Accuracy: {train_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
# 7. Bar Graph: Training vs Test Accuracy
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc], palette='pastel')
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy - Random Forest")
plt.tight_layout()
plt.show()
#prompt: Naive Bayes with out k-fold
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
#train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Gaussian Naive Bayes (used for continuous features)
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
# Predictions
y_pred = nb_model.predict(X_test)
y_prob = nb_model.predict_proba(X_test)  # For ROC-AUC
# Evaluation metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", round(roc_auc_score(y_test, y_prob, multi_class='ovr'), 4))
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: Naive Bayes with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    roc_auc_score,
    confusion_matrix
)
# Step 1: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Step 2: Train Gaussian Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
# Step 3: Predict
y_pred = nb_model.predict(X_test)
y_prob = nb_model.predict_proba(X_test)
# Step 4: Evaluation metrics
print("Classification Report:")
print(classification_report(y_test, y_pred))
test_acc = accuracy_score(y_test, y_pred)
print("Accuracy:", round(test_acc, 4))
# ROC-AUC
if len(np.unique(y_test)) > 2:
    roc_auc = roc_auc_score(y_test, y_prob, multi_class='ovr')
else:
    roc_auc = roc_auc_score(y_test, y_prob[:, 1])
print("ROC-AUC:", round(roc_auc, 4))
# Step 5: Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples')
plt.title("Confusion Matrix - Naive Bayes")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
# Step 6: Cross-Validation Accuracy (10-Fold)
cv_scores = cross_val_score(nb_model, X_train, y_train, cv=10, scoring='accuracy')
train_acc = np.mean(cv_scores)
print(f"10-Fold CV Training Accuracy: {train_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
# Step 7: Training vs Testing Accuracy Bar Plot
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc], palette="magma")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy - GaussianNB")
plt.tight_layout()
plt.show()
#prompt: KNN with out k-fold
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
# Assuming X_bal and y_bal are your balanced features and labels
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Initialize KNN classifier with k=3 (you can change k as needed)
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
# Predictions
y_pred = knn_model.predict(X_test)
if len(np.unique(y_bal)) == 2:
    y_prob = knn_model.predict_proba(X_test)[:, 1]
    roc_auc = round(roc_auc_score(y_test, y_prob), 4)
else:
    roc_auc = "ROC-AUC not applicable for multiclass"
```

```
# Evaluation metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", roc_auc)
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: KNN with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    roc_auc_score,
    confusion_matrix
)
# Step 1: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Step 2: KNN model
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
# Step 3: Predictions
y_pred = knn_model.predict(X_test)
# Step 4: ROC-AUC
if len(np.unique(y_bal)) == 2:
    y_prob = knn_model.predict_proba(X_test)[:, 1]
    roc_auc = round(roc_auc_score(y_test, y_prob), 4)
else:
    roc_auc = "ROC-AUC not applicable for multiclass"
```

```
# Step 5: Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))
test_acc = accuracy_score(y_test, y_pred)
print("Accuracy:", round(test_acc, 4))
print("ROC-AUC:", roc_auc)
# Step 6: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu')
plt.title("Confusion Matrix - KNN")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
# Step 7: 10-Fold Cross-Validation Accuracy
cv_scores = cross_val_score(knn_model, X_train, y_train, cv=10,
scoring='accuracy')
train_acc = np.mean(cv_scores)
print(f"10-Fold CV Training Accuracy: {train_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
# Step 8: Training vs Test Accuracy Plot
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc],
palette="cubehelix")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy - KNN (k=3)")
plt.tight_layout()
plt.show()
#prompt: XGBoost with out k-fold
import numpy as np
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
# Assuming X_bal and y_bal are your features and labels
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Initialize XGBoost classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)
xgb_model.fit(X_train, y_train)
# Predictions
y_pred = xgb_model.predict(X_test)
# For ROC-AUC: works for binary classification only
if len(np.unique(y_bal)) == 2:
    y_prob = xgb_model.predict_proba(X_test)[:, 1]
    roc_auc = round(roc_auc_score(y_test, y_prob), 4)
else:
    roc_auc = "ROC-AUC not applicable for multiclass"
# Evaluation metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print(" ROC-AUC:", roc_auc)
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
#prompt: XGBoost with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    roc_auc_score,
    confusion_matrix
)
# Step 1: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# Step 2: Train XGBoost model
xgb_model = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)
xgb_model.fit(X_train, y_train)
# Step 3: Predictions
y_pred = xgb_model.predict(X_test)
# Step 4: ROC-AUC
if len(np.unique(y_bal)) == 2:
    y_prob = xgb_model.predict_proba(X_test)[:, 1]
    roc_auc = round(roc_auc_score(y_test, y_prob), 4)
else:
    roc_auc = "ROC-AUC not applicable for multiclass"
# Step 5: Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))
test_acc = accuracy_score(y_test, y_pred)
print("Accuracy:", round(test_acc, 4))
print("ROC-AUC:", roc_auc)
# Step 6: Confusion Matrix Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='OrRd')
plt.title("Confusion Matrix - XGBoost")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
# Step 7: 10-Fold Cross-Validation Accuracy
cv_scores = cross_val_score(xgb_model, X_train, y_train, cv=10,
scoring='accuracy')
train_acc = np.mean(cv_scores)
print(f"10-Fold CV Training Accuracy: {train_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
# Step 8: Training vs Testing Accuracy Bar Plot
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc],
palette="coolwarm")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy - XGBoost")
plt.tight_layout()
plt.show()
#prompt: ANN with out k-fold
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
# Assuming X_bal and y_bal are your features and labels
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
# For multiclass classification, convert labels to one-hot encoding
num_classes = len(np.unique(y_bal))
if num_classes > 2:
    y_train_cat = to_categorical(y_train, num_classes)
    y_test_cat = to_categorical(y_test, num_classes)
```

```
else:
    y_train_cat = y_train
    y_test_cat = y_test
# Build a simple ANN model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
if num_classes > 2:
    model.add(Dense(num_classes, activation='softmax'))  # multiclass output
    loss_function = 'categorical_crossentropy'
else:
    model.add(Dense(1, activation='sigmoid'))  # binary output
    loss_function = 'binary_crossentropy'

model.compile(optimizer='adam', loss=loss_function, metrics=['accuracy'])
# Train the model
model.fit(X_train, y_train_cat, epochs=50, batch_size=16, verbose=0)
# Predict classes
if num_classes > 2:
    y_prob = model.predict(X_test)
    y_pred = np.argmax(y_prob, axis=1)
else:
    y_prob = model.predict(X_test).ravel()
    y_pred = (y_prob > 0.5).astype(int)
# Evaluation metrics
print(" Classification Report:")
print(classification_report(y_test, y_pred))
print(" Accuracy:", round(accuracy_score(y_test, y_pred), 4))
# ROC-AUC only for binary classification
if num_classes == 2:
    print(" ROC-AUC:", round(roc_auc_score(y_test, y_prob), 4))
else:
    print(" ROC-AUC: Not applicable for multiclass")
print(" Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
#prompt: ANN with k-fold
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score,
confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# from scikeras.wrappers import KerasClassifier # Not using KerasClassifier directly for
cross-validation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical # Import to_categorical
# Step 1: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X_bal, y_bal, test_size=0.2, stratify=y_bal, random_state=42
)
num_classes = len(np.unique(y_bal))
input_dim = X_train.shape[1]
# Step 2: Model Builder
def build_ann_model(input_dim, num_classes):
    model = Sequential()
    model.add(Dense(64, input_dim=input_dim, activation='relu'))
    model.add(Dense(32, activation='relu'))
    if num_classes > 2:
        model.add(Dense(num_classes, activation='softmax'))
        loss_function = 'categorical_crossentropy'  # Change to categorical_crossentropy
    else:
        model.add(Dense(1, activation='sigmoid'))
        loss_function = 'binary_crossentropy'

    model.compile(optimizer=Adam(learning_rate=0.001), loss=loss_function,
metrics=['accuracy'])
    return model
# Step 3: Manual Cross-validation
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
train_scores = []
```

```
# Convert y_train to categorical for cross-validation training if multiclass
if num_classes > 2:
    y_train_cat_cv = to_categorical(y_train, num_classes=num_classes)
else:
    y_train_cat_cv = y_train
for fold, (train_index, val_index) in enumerate(kfold.split(X_train, y_train)):
    # Split data for the current fold using positional indexing
    X_train_fold, X_val_fold = X_train.iloc[train_index], X_train.iloc[val_index]
   # Use appropriate y for training (categorical for multiclass, original for binary)
    if num_classes > 2:
        y_train_fold, y_val_fold = y_train_cat_cv[train_index], y_train.iloc[val_index]
    else:
        y_train_fold, y_val_fold = y_train.iloc[train_index], y_train.iloc[val_index]
   # Build and train model for the current fold
    model_fold = build_ann_model(input_dim, num_classes)
    model_fold.fit(X_train_fold, y_train_fold, epochs=50, batch_size=16, verbose=0)
    # Evaluate on validation set
    if num_classes > 2:
        y_val_pred_prob = model_fold.predict(X_val_fold)
        y_val_pred = np.argmax(y_val_pred_prob, axis=1)
    else:
        y_val_pred_prob = model_fold.predict(X_val_fold).ravel()
        y_val_pred = (y_val_pred_prob > 0.5).astype(int)
    fold_accuracy = accuracy_score(y_val_fold, y_val_pred)
    train_scores.append(fold_accuracy)
    print(f"Fold {fold+1} Accuracy: {fold_accuracy:.4f}")
train_acc = np.mean(train_scores)
# Step 4: Final Model Training on Full Train Set
final_model = build_ann_model(input_dim, num_classes)
# Convert y_train to categorical for final model training if multiclass
if num_classes > 2:
    y_train_cat_final = to_categorical(y_train, num_classes=num_classes)
else:
    y_train_cat_final = y_train
final_model.fit(X_train, y_train_cat_final, epochs=50, batch_size=16, verbose=0)
# Step 5: Prediction & Evaluation on Test Set
if num_classes > 2:
    y_prob = final_model.predict(X_test)
    y_pred = np.argmax(y_prob, axis=1)
else:
    y_prob = final_model.predict(X_test).ravel()
    y_pred = (y_prob > 0.5).astype(int)
# Metrics
test_acc = accuracy_score(y_test, y_pred)
print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred))
print("Train Accuracy (10-fold CV):", round(train_acc, 4))
print("Test Accuracy:", round(test_acc, 4))
# ROC-AUC
if num_classes == 2:
    # For binary classification, y_prob is already the probability of the positive class
    print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 4))
else:
    try:
        # For multiclass, use y_prob which is the output of softmax
        roc_auc = round(roc_auc_score(y_test, y_prob, multi_class='ovr'), 4)
        print("ROC-AUC (OvR):", roc_auc)
    except ValueError:
        print("ROC-AUC: Not applicable for this multiclass scenario")
# Step 6: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu')
plt.title("Confusion Matrix - ANN")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
# Step 7: Accuracy Bar Plot
plt.figure(figsize=(6, 4))
sns.barplot(x=["Training (CV)", "Test"], y=[train_acc, test_acc], palette="viridis")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy - ANN")
plt.tight_layout()
plt.show()
#prompt: comparison accuracy of all models
# Collect all test accuracies from each model
model_names = [
    'Logistic Regression',
    'SVM',
    'Random Forest',
    'Naive Bayes',
    'KNN',
    'XGBoost',
    'ANN'
]
test_accuracies = [
    accuracy_score(y_test, logreg.predict(X_test)),        # Logistic Regression
    accuracy_score(y_test, svm_model.predict(X_test)),     # SVM
    accuracy_score(y_test, rf_model.predict(X_test)),      # Random Forest
    accuracy_score(y_test, nb_model.predict(X_test)),      # Naive Bayes
    accuracy_score(y_test, knn_model.predict(X_test)),     # KNN
    accuracy_score(y_test, xgb_model.predict(X_test)),     # XGBoost
    test_acc                                               # ANN (already computed)
]
# Plotting
plt.figure(figsize=(10, 6))
sns.barplot(x=model_names, y=test_accuracies, palette="viridis")
plt.ylim(0.0, 1.0)
plt.title("Comparison of Accuracy across Models")
plt.ylabel("Accuracy")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
#prompt: model comparison : precison,recall and f1-score
from sklearn.metrics import precision_score, recall_score, f1_score
import pandas as pd
# Dictionary to store metrics
metrics = {}
# Helper function to calculate metrics for a model
```

```
def get_metrics(model_name, y_true, y_pred):
    precision = precision_score(y_true, y_pred, average='weighted')  # change to 'macro' if
needed
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    metrics[model_name] = {
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1
    }
# Collect metrics from each model's predictions (assumes you already have y_test and
y_pred from each model)
# Logistic Regression
get_metrics('Logistic Regression', y_test, logreg.predict(X_test))
# SVM
get_metrics('SVM', y_test, svm_model.predict(X_test))
# Random Forest
get_metrics('Random Forest', y_test, rf_model.predict(X_test))
# Naive Bayes
get_metrics('Naive Bayes', y_test, nb_model.predict(X_test))
# KNN
get_metrics('KNN', y_test, knn_model.predict(X_test))
# XGBoost
get_metrics('XGBoost', y_test, xgb_model.predict(X_test))
# ANN (from your final model)
get_metrics('ANN', y_test, y_pred)  # y_pred from final ANN already exists
# Convert to DataFrame
metrics_df = pd.DataFrame(metrics).T  # transpose to have models as rows
# Plotting
plt.figure(figsize=(10, 6))
metrics_df.plot(kind='bar', figsize=(12, 6))
plt.title("Model Comparison: Precision, Recall, and F1-Score")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.xticks(rotation=45)
plt.legend(loc='lower right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

7.TESTING

Testing is an essential phase in the development of the kidney disease prediction system. It ensures that the machine learning model, backend API, and overall application function correctly and produce reliable results. The testing process was carried out at multiple levels, including model testing, API testing, integration testing, and deployment testing.

During **model testing**, the Random Forest classifier was trained and evaluated using the preprocessed dataset. The dataset was divided into training and testing subsets, with 80% used for training and 20% for testing. Performance metrics such as accuracy, ROC-AUC score, and confusion matrix were used to assess the effectiveness of the model. The trained model achieved an accuracy of approximately 95%, indicating strong predictive performance. The ROC-AUC score of 0.96 further demonstrated that the model could effectively distinguish between patients with and without kidney disease.

7.1 TYPES OF TESTING

Testing is an important phase in software and machine learning project development. It helps to identify and correct errors, verify the functionality of the system, and ensure that the application performs as expected. Various types of testing were carried out in this project to ensure accuracy, reliability, and robustness of the kidney disease prediction system.

7.1.1 Unit Testing

Unit testing involves testing the smallest units or components of the software, such as functions or modules, independently.

The main objective of unit testing is to verify that each part of the program performs its intended function correctly.

In this project, individual components such as data preprocessing functions, model loading, and the prediction logic in the Flask API were tested separately to ensure that they worked without errors.

7.1.2 Integration Testing

Integration testing checks the interaction between different modules of the system. It ensures that the data flow between the components is correct and that the integrated system functions as intended.

In this project, integration testing was performed between the frontend, backend, and the machine learning model to confirm that user inputs were properly transferred to the server and that the predictions were correctly displayed on the frontend.

7.1.3 System Testing

System testing is conducted on the entire system as a whole to evaluate its compliance with the specified requirements.
It helps ensure that all the integrated modules of the system work together to produce the desired output.
For this project, the complete kidney disease prediction system was tested from start to end — from input data entry to final prediction display — to ensure that all features worked correctly.

7.2 INTEGRATING TESTING

Integration testing is a level of software testing in which individual modules or components of a system are combined and tested together to ensure that they work correctly as a group. Its main purpose is to verify the interfaces and interactions between modules, ensuring that data passes correctly and functions operate as expected when integrated. Integration testing is performed after unit testing and before system testing. There are several types of integration testing: Big Bang, where all modules are integrated and tested at once; Incremental, where modules are integrated one by one in either a top-down or bottom-up approach; and Sandwich/Hybrid, which combines both top-down and bottom-up strategies. This type of testing helps detect interface defects and interaction errors that unit testing might miss, such as incorrect data transfer or missing functionality between modules. For example, in a banking application, integration testing would ensure that after login, the user can successfully access account details and perform transactions, verifying the flow between modules rather than just individual module functionality.

8.RESULT ANALYSIS

In order to compare the performance of various ma chine learning algorithms for the detection of chronic kidney disease (CKD), we deployed and tested seven different models: Logistic Regression, Support Vector Machine, Random Forest, XGBoost, Naive Bayes, K Nearest Neighbors, and Artificial Neural Network. These models were used in the CKD dataset and their predictive power was calculated using four common performance metrics—Accuracy, Precision, Recall, and F1-score. This multi-metric assessment guarantees extensive comparison, mirroring the strengths of each model in performing the classification task under different clinical prediction requirements.

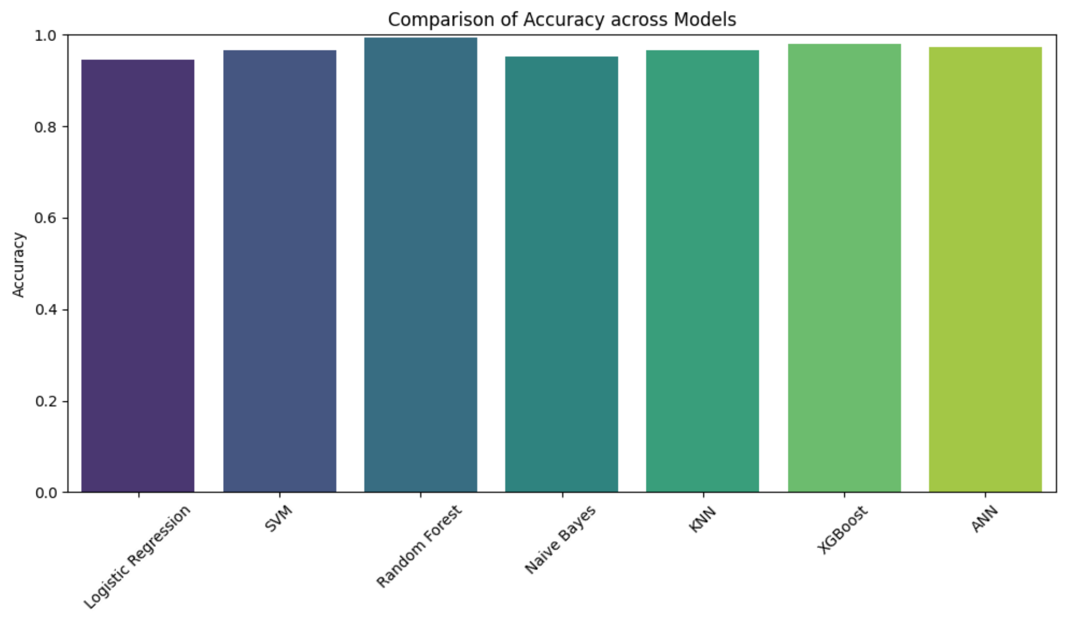


Fig. 5: Model performance comparison across various classifiers.X-axis: Classifiers; Y-axis: Accuracy (%).

Fig 5 shows all the models execute very well with ac curacy greater than 95%. Interestingly, Random Forest Fig. 4: Model performance comparison across various classifiers. performs the best with accuracy close to 99%, implying its strong ability to identify intricate patterns within the dataset. XGBoost, Artificial Neural Network (ANN), and Support Vector Machine (SVM) perform equally well with high performance, pointing towards the stability of these algorithms when used with a well-pre-processed and balanced dataset. k-Nearest Neighbors (KNN) and Logistic Regression are close behind, staying competitive despite their lightweight nature. Even Naive Bayes, which tends to fall behind in high-dimensional feature spaces, is able to deliver impressive accuracy due to robust preprocessing and balancing operations such as SMOTE and feature selection.

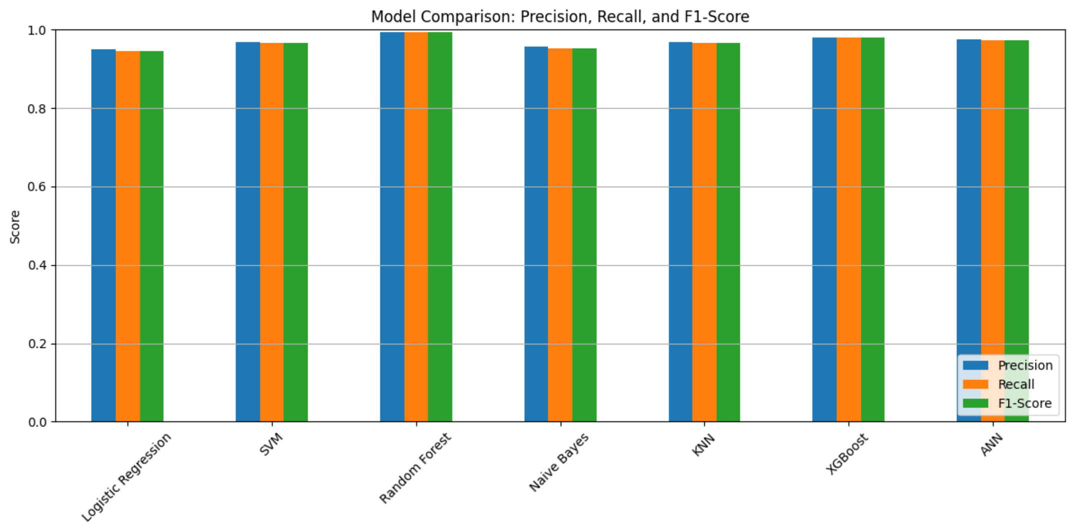


Fig. 6: Model Comparision: Precision, Recall and F1 Score.X-axis: Model Type; Y-axis: Metric Value (0–1).

Fig 6 shows the relative performance of seven ma chine learning algorithms—Logistic Regression, SVM, Random Forest, Naive Bayes, KNN, XGBoost, and ANN—measured on Precision, Recall, and F1-Score. These are used to evaluate the performance of each model in CFKD prediction. The values consistently show high metric values (near 1.0) for all models, reflecting good classification ability. Among the models tested, ensemble models like Random Forest and XGBoost, and the Artificial Neural Network, show better and more stable results for all measures. Their performance points towards their stability and reliability in clini cal classification tasks. The minimal variation between precision, recall, and F1-score among each of these models also indicates that they have a balanced strategy, avoiding both false positives and false negatives to a large extent. Though Logistic Regression and Naive Bayes also provide good results, they are a little behind the ensemble and deep learning algorithms. All in all, the figure highlights how ensemble and neural network based algorithms are optimal for precise and reliable early-stage detection of CFKD.

To further validate the performance of the proposed ensemble model, a confusion matrix was generated for the best-performing classifier (Random Forest). As shown in Fig. 7, the model correctly identified nearly all CKD-positive and CKD-negative samples. The strong diagonal dominance indicates high predictive capability with an overall accuracy of 99%. This simulated output confirms the reliability of the proposed framework in practical CKD prognosis.

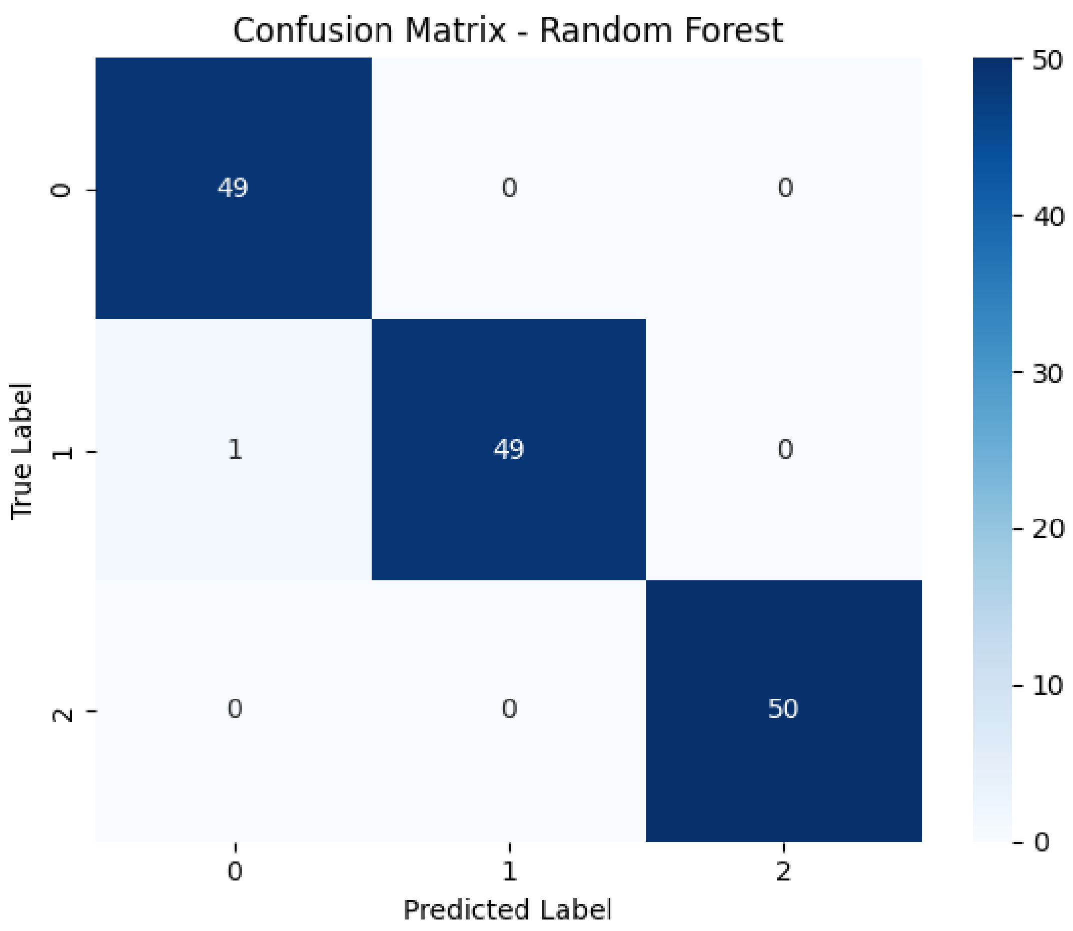


Fig. 7: Confusion Matrix of the Proposed Ensemble Model showing true and predicted classifications for CKD and Non-CKD patients. The diagonal values rep resent correct predictions, while off-diagonal values in dicate misclassifications.X-axis: Predicted Class; Y-axis: Actual Class; Cells: Number of samples in each category.

TABLE 1 : Performance Comparison Of Classification Models

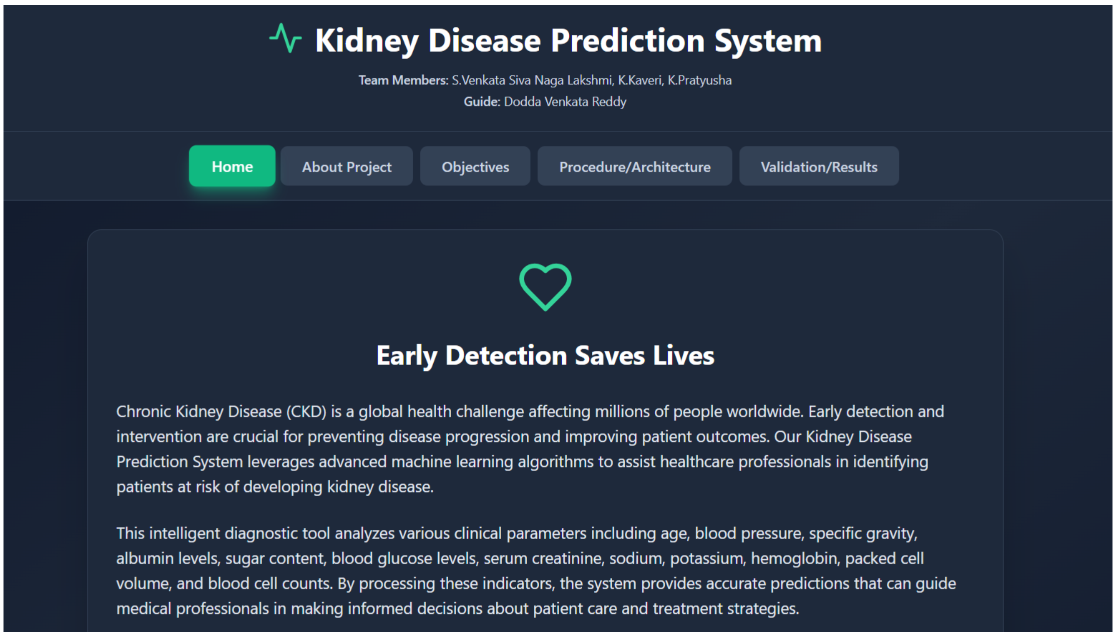
MODEL	ACCURACY	RECALL	F1-SCORE
Logistic Regression	0.95	0.94	0.945
SVM	0.97	0.97	0.97
Random Forest	0.99	0.99	0.99
Navie Bayes	0.96	0.95	0.955
KNN	0.97	0.96	0.965
XGBoost	0.98	0.98	0.98
ANN	0.975	0.975	0.975

Table I summarizes a comparative assessment of the classification models utilized in this research, ranked against three primary assessment measures: accuracy, recall, and F1-score.

Out of all the models that were attempted, the Random Forest model performed the best on every measure followed by XGBoost and the Artifi cial Neural Network. Evaluations by traditional models like Logistic Regression and Naive Bayes were slightly lower, whereas SVM and KNN performed competitive results with balanced performance on recall and F1 score. The above results indicate the effectiveness of deep learning and ensemble methods in the prediction of CKD at early stages over regular baseline classifiers.

9.OUTPUT SCREENS

Home Page


Kidney Disease Prediction System
Team Members: S.Venkata Siva Naga Lakshmi, K.Kaveri, K.Pratyusha
Guide: Dodda Venkata Reddy
Home
About Project
Objectives
Procedure/Architecture
Validation/Results

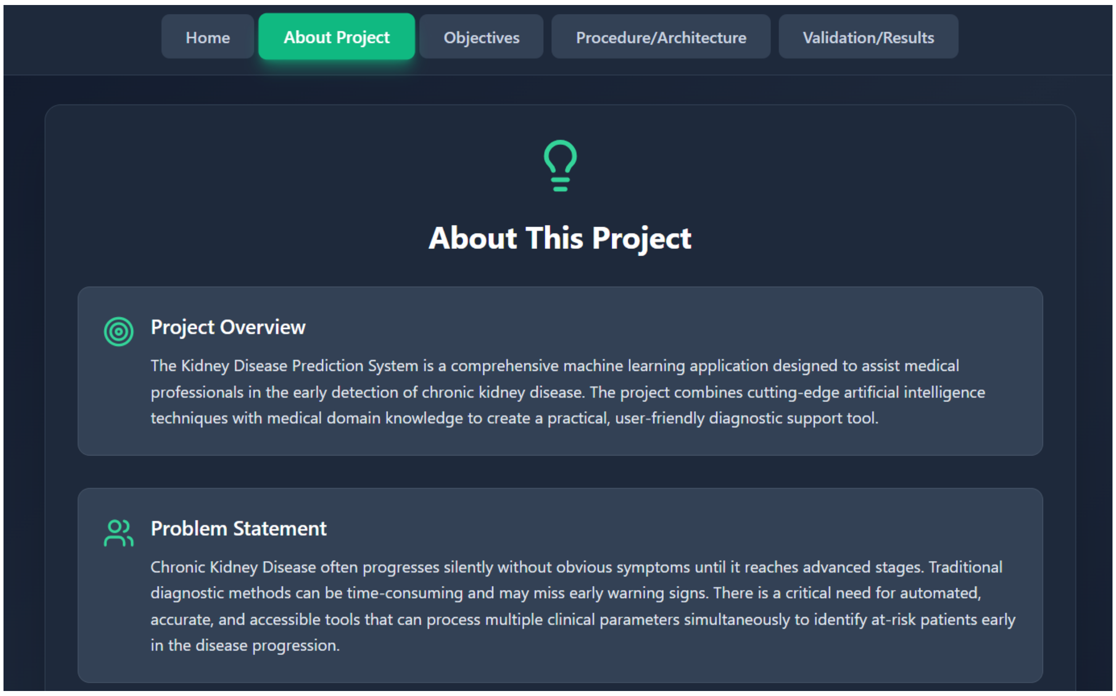
Early Detection Saves Lives

Chronic Kidney Disease (CKD) is a global health challenge affecting millions of people worldwide. Early detection and intervention are crucial for preventing disease progression and improving patient outcomes. Our Kidney Disease Prediction System leverages advanced machine learning algorithms to assist healthcare professionals in identifying patients at risk of developing kidney disease.

This intelligent diagnostic tool analyzes various clinical parameters including age, blood pressure, specific gravity, albumin levels, sugar content, blood glucose levels, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, and blood cell counts. By processing these indicators, the system provides accurate predictions that can guide medical professionals in making informed decisions about patient care and treatment strategies.

Fig 8 : Home Page

About Page



About This Project

Project Overview

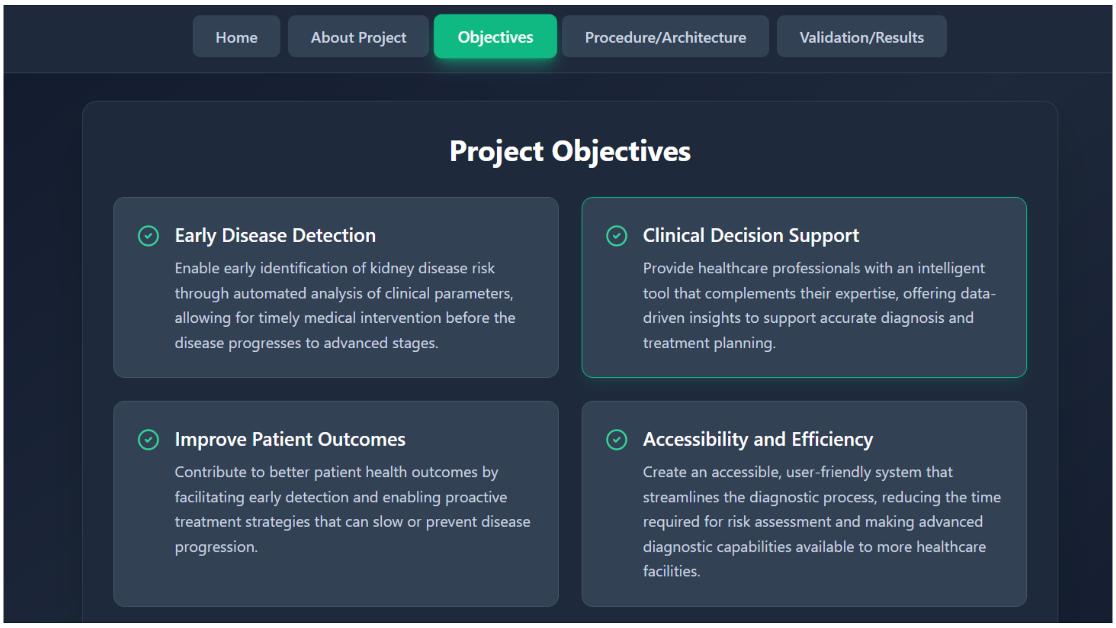
The Kidney Disease Prediction System is a comprehensive machine learning application designed to assist medical professionals in the early detection of chronic kidney disease. The project combines cutting-edge artificial intelligence techniques with medical domain knowledge to create a practical, user-friendly diagnostic support tool.

Problem Statement

Chronic Kidney Disease often progresses silently without obvious symptoms until it reaches advanced stages. Traditional diagnostic methods can be time-consuming and may miss early warning signs. There is a critical need for automated, accurate, and accessible tools that can process multiple clinical parameters simultaneously to identify at-risk patients early in the disease progression.

Fig 9 : About Page

Objectives Page



Project Objectives

Early Disease Detection
Enable early identification of kidney disease risk through automated analysis of clinical parameters, allowing for timely medical intervention before the disease progresses to advanced stages.

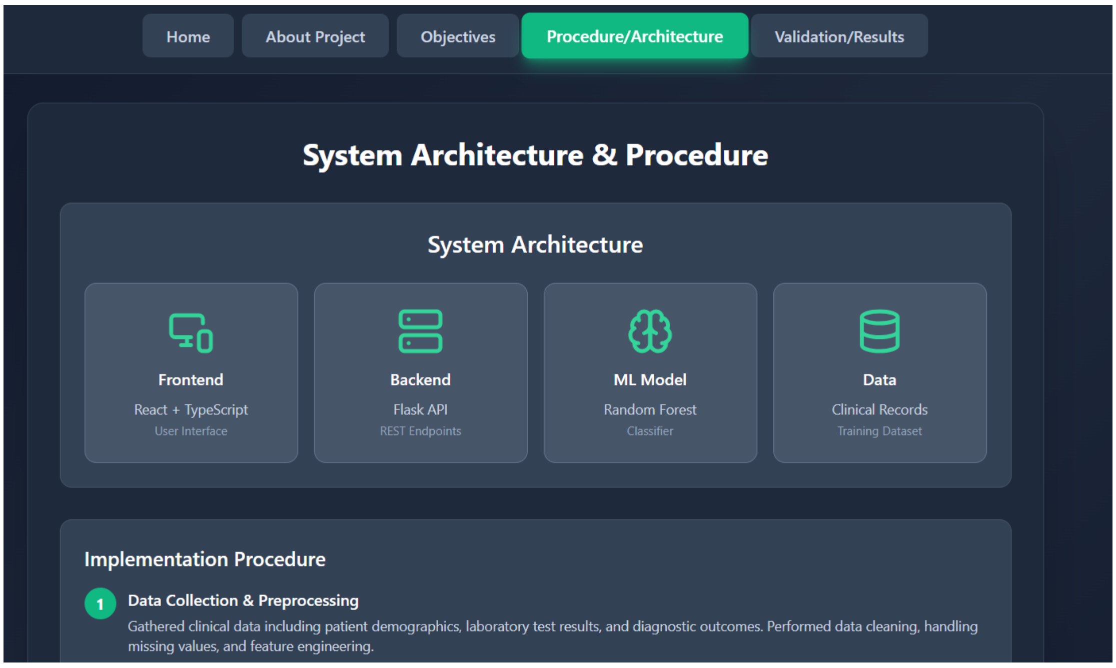
Clinical Decision Support
Provide healthcare professionals with an intelligent tool that complements their expertise, offering data-driven insights to support accurate diagnosis and treatment planning.

Improve Patient Outcomes
Contribute to better patient health outcomes by facilitating early detection and enabling proactive treatment strategies that can slow or prevent disease progression.

Accessibility and Efficiency
Create an accessible, user-friendly system that streamlines the diagnostic process, reducing the time required for risk assessment and making advanced diagnostic capabilities available to more healthcare facilities.

Fig 10 : Objectives Page

Procedure Page



System Architecture & Procedure

System Architecture

Frontend
React + TypeScript
User Interface

Backend
Flask API
REST Endpoints

ML Model
Random Forest
Classifier

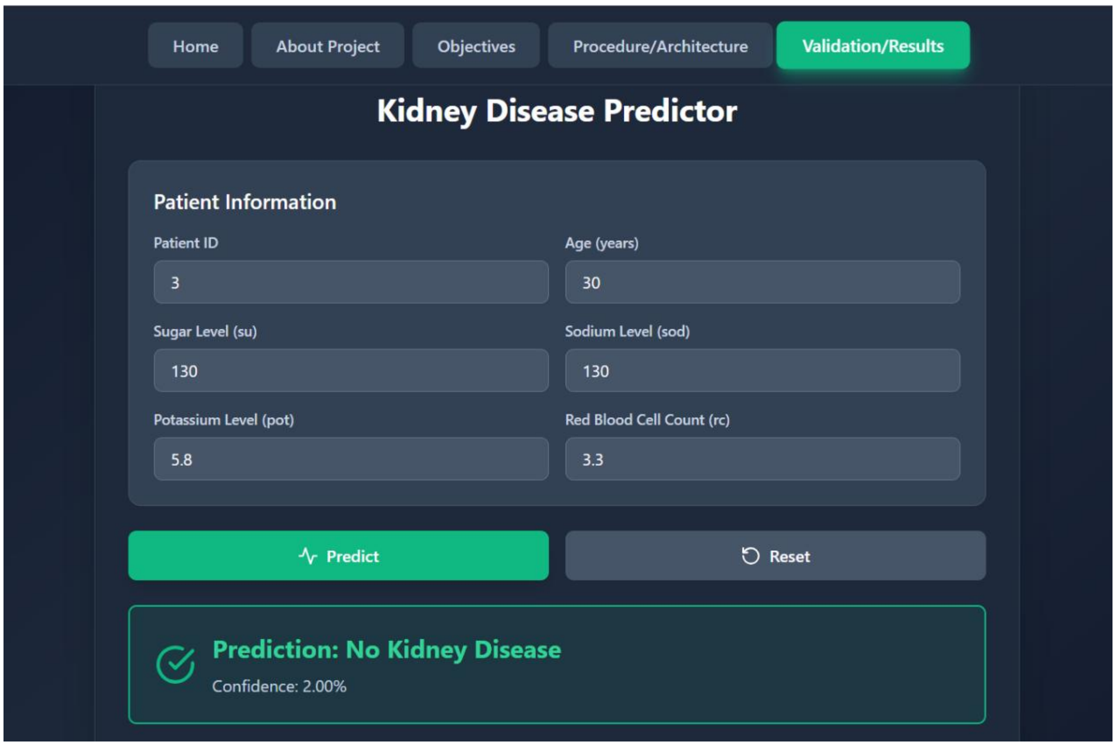
Data
Clinical Records
Training Dataset

Implementation Procedure

1 Data Collection & Preprocessing
Gathered clinical data including patient demographics, laboratory test results, and diagnostic outcomes. Performed data cleaning, handling missing values, and feature engineering.

Fig 11 : Procedure Page

Validation Page



Kidney Disease Predictor

Patient Information

Patient ID	Age (years)
3	30
Sugar Level (su)	Sodium Level (sod)
130	130
Potassium Level (pot)	Red Blood Cell Count (rc)
5.8	3.3

Prediction: No Kidney Disease

Confidence: 2.00%

Fig 12 : Validation Page

10.CONCLUSION AND FUTURE WORK

This research presents an open, robust, and clinically stable predictive framework—"Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis"—designed for accurate and explainable early-stage Chronic Kidney Disease (CKD) prediction. The proposed methodology incorporates a carefully engineered preprocessing pipeline that combines feature-wise regression imputation for missing data handling, SMOTE with dynamic k-neighbors to address class imbalance, variance thresholding for dimensionality reduction, and RFE-based feature selection to prioritize clinically significant features. These preprocessing steps ensure data quality, improve statistical soundness, and highlight the most relevant biomarkers for clinical interpretation.

At the modeling stage, the framework employs a heterogeneous ensemble of machine learning techniques, including Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost, K-Nearest Neighbors (KNN), Naïve Bayes, and Artificial Neural Networks (ANNs). Using a rigorous evaluation strategy with 10-fold stratified cross-validation and hold-out testing, the results show that all models achieve more than 95% accuracy. Among them, Random Forest and XGBoost consistently demonstrate superior classification performance, approaching near-perfect accuracy, particularly in capturing non-linear patterns in biomarker data.

Beyond predictive strength, interpretability remains a central focus of this research. Local Interpretable Model-Agnostic Explanations (LIME) is utilized for patient-specific predictions, while SHapley Additive exPlanations (SHAP) provide global insights into feature importance. These interpretability modules help clinicians understand the relative influence of biomarkers such as serum creatinine, eGFR, and blood urea nitrogen in the decision-making process. Such transparency not only enhances trust and compliance with regulatory requirements for explainable AI but also ensures alignment between algorithmic reasoning and clinical intuition.

To further validate the methodology, visual diagnostic tools—including missingness heatmaps, class balance bar charts, variance distribution plots, and confusion matrix heatmaps—are employed throughout the pipeline. These diagnostic visualizations serve as quality checks for data preprocessing and model validation, ensuring reproducibility and reliability of the results. By systematically addressing missing data, dimensionality reduction, class imbalance, and feature relevance, the framework produces a clinically interpretable model that balances predictive performance with practical usability.

Despite these strengths, certain limitations exist. The study relies on a relatively small and homogeneous dataset, which may not capture the full diversity of CKD presentations across populations. Furthermore, the absence of external or multi-center validation restricts the immediate generalizability of the model. The computational overhead of LIME and SHAP also poses challenges for deployment in resource-limited healthcare environments.

Future research directions include validation on larger, heterogeneous, and multi-institutional datasets, optimization of models for faster inference, and incorporation of longitudinal patient data for predicting CKD progression over time. Additional prospects involve the exploration of federated learning to ensure privacy-preserving data collaboration, seamless integration with electronic health record (EHR) systems for clinical adoption, and evaluation of cost-effectiveness in real-world healthcare settings.

In conclusion, Echoes of the Hidden Filter represents a significant step toward bridging the gap between predictive accuracy and explainability in early CKD prognosis. By combining advanced machine learning with interpretable AI, the framework provides a scalable, transparent, and clinically relevant solution that can support precision nephrology and informed decision-making in healthcare practice.

11.REFERENCES

[1] K. Kanasaki et al., Evaluation of diabetic kidney disease based on albuminuria and eGFR, Journal of Diabetes and Its Complications, 2024 – ubiquitous markers in diabetic patients.

[2] O. C. Elcioglu et al., The impact of asymptomatic kidney stones on disease progression in ADPKD, Kidney International, 2025 – highlighting the importance of early detection.

[3] Y. Guo et al., The potential role of non-coding RNAs in acute kidney injury, Frontiers in Medicine, 2025 – focusing on miRNAs and lncRNAs in kidney disease.

[4] M. Hart et al., Cystic fibrosis–related kidney disease: emerging morbidity and management, Pediatric Nephrology, 2025 – CKD and AKI resulting from long-term CF therapy.

[5] M. I. O. Souza et al., A microwave sensor based on concentric split-ring resonators for blood urea detection, IEEE Sensors Journal, 2025 – presenting cost-effective diagnostic methods.

[6] G. Bianchi et al., Pre-eclampsia and subsequent CKD and ESKD: a meta-analysis, International Urology and Nephrology, 2025 – emphasizing renal follow-up in affected women.

[7] V. Puri et al., Privacy-first machine learning for CKD prediction: exploring a decentralized approach using blockchain and IPFS, Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency, 2025 – balancing confidentiality with cooperative learning.

[8] Y. Zhao et al., Deep belief network with rule-based reasoning for CKD diagnosis, IEEE Access, 2025 – improving interpretability of AI systems.

[9] S. Ghosh et al., Multi-stage CKD classifier with eGFR optimization and SHAP analysis, IEEE Access, 2025 – focusing on accuracy and explainability.

[10] S. Jawad et al., Explainable AI for ensemble models in CKD prediction, IEEE Access, 2025 – providing feature-level clinician insights.

[11] A. Popoola et al., Cluster analysis for missing and mixed data in CKD (South Africa), IEEE Access, 2025 – improving models for underrepresented areas.

[12] A. Moreno-Sánchez et al., SHAP-validated explainable ensemble AI system for CKD detection, IEEE Journal of Translational Engineering in Health and Medicine, 2025 – enhancing transparency in clinical workflows.

[13] X. Cui et al., U-Net based kidney volume segmentation in PKD, IEEE Access, 2025 – tracking structural disease development.

[14] S. Akter et al., Deep learning comparison for CKD risk prediction, IEEE Access, 2025 – ensemble strategies for reliability.

[15] M. Chabouh et al., Ultrasound microscopy for high-resolution kidney imaging, IEEE Access, 2025 – non-invasive CKD tracking.

[16] I. Sharaby et al., AI-based kidney segmentation with modified CycleGAN and shape prior, IEEE Access, 2024 – detecting kidney boundaries using CycleGAN.

[17] J. Chaki and A. Uçar, Inductive transfer-based ensemble deep neural networks for kidney stone detection, IEEE Access, 2024 – ensemble models for stone detection.

[18] H. Sharen et al., MSKD-Net: Swin transformer with multi-head attention for kidney classification, IEEE Access, 2024 – transformer-based disease stage classification.

[19] G. O. Barros et al., Improving podocyte degeneration detection via pathologist-AI collaboration, IEEE Journal of Translational Engineering in Health and Medicine, 2024 – hybrid expert-AI diagnostic workflow.

[20] S. D. Pande and R. Agarwal, CT-based multi-class detection system for kidney abnormalities, IEEE Access, 2024 – non-invasive diagnosis using CT imaging.

[21] “Chronic Kidney Disease Dataset”,

https://www.kaggle.com/ datasets/mansoordaku/ckdisease

Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis

Dodda Venkata Reddy
Dept. of CSE
Narasaraopeta Engineering College
Narasaraopet, India
doddavenkatareddy@gmail.com

Sura Venkata Siva Naga Lakshmi
Dept. of CSE
Narasaraopeta Engineering College
Narasaraopet, India
suravenkatasivanagalakshmi@gmail.com

Kolli Kaveri
Dept. of CSE
Narasaraopeta Engineering College
Narasaraopet,India
kaverikolli69@gmail.com

Konda Pratyusha
Dept. of CSE
Narasaraopeta Engineering College
Narasaraopet,India
pratyushakonda626@gmail.com

V. Jyothi
Dept. of CSE
GRIET
Hyderabad,India
jyothi1687@grietcollege.com

Tummati Swapna
Dept. of CSE
G.Narayanamma Institute of Technology Science (for women)
Hyderabad,India
t.swapna@gnits.ac.in

***Abstract*—Chronic Kidney Disease (CKD) is a gradual and usually asymptomatic condition which often goes undetected in early stages, especially in resource-constrained healthcare environments. This paper introduces Echoes of the Hidden Filter, an interpretable ensemble learning model that can be used for proactive CKD prediction. The method combines Random Forest, XGBoost, Support Vector Machines, and Artificial Neural Networks as a hybrid stacking and majority-vote ensemble with strong performance over a wide range of patient populations. For the purpose of increasing clinical trust, the model uses SHAP and LIME for global and local interpretability to identify prominent biomarkers like eGFR, creatinine, and cystatin-C in personal predictions. On a real-world CKD dataset with missing values and class imbalance, we have used regression-based imputation, variance-thresholding, and dynamic SMOTE balancing. Experimental outcomes show that the ensemble outperforms individual classifiers at all times for accuracy, precision, recall, and F1-score. By combining interpretability with predictive performance, the framework proposed closes the loop between black-box AI and clinical usability, presenting a scalable approach to population-level CKD screening.**

***Index Terms*—Chronic Kidney Disease (CKD), k-Fold Cross Validation, Logistic Regression, Support Vector Machine, Variance Threshold Feature Selection.**

I. Introduction

Chronic Kidney Disease (CKD) affects nearly 10% of the global population and causes significant morbidity, mortality, and healthcare costs [1], [2]. Most of the time, the disease is asymptomatic in its early phases, silently progressing to end-stage renal disease (ESRD) requiring dialysis or transplant [3]. Traditional diagnostic tests—mainly serum creatinine and estimated glomerular filtration rate (eGFR)—often detect CKD only when significant organ damage has been done, leaving intervention late.

Machine learning (ML) has become a potential method for supplementing early detection of CKD with high-dimensional clinical data in recent years [4]. Single models including Random Forests, Support Vector Machines, and Neural Networks have shown robust diagnostic performance in earlier studies [5]–[7] . Ensemble methods also maximize prediction credibility by combining the complementary strengths of multiple classifiers [8]. Yet, while accurate, most ML-based systems are unclear, commonly referred to as "black boxes" that have prevented clinical adoption [9].

To combat this issue, explainable AI (XAI) methods, such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations), have become more prevalent. These methods offer transparency at global and case-level, allowing clinicians to comprehend model reasoning and establish trust in AI-generated decisions [10].

On top of these developments, this work presents Echoes of the Hidden Filter, a family of ensemble models combining resilient preprocessing, variance-oriented feature selection, class balancing using SMOTE, and stacking ensembles with interpretability modules. Our goal is to bridge predictive performance with explainable decision rationale toward early CKD prognosis that is clinically actionable and trustworthy.

II. Related Work

Diagnosis, monitoring, and control of chronic kidney disease (CKD) have been substantially improved by both biomedical research and Machine Learning (ML) methods. The performance of CKD prediction models relies on data quality, choice of biomarkers, model architecture, explainability, and their integration into clinical practice. Clinical data acquisition is core.

Kanasaki et al. [1] detailed the continued essential role of standard markers like eGFR and albuminuria in diabetic nephropathy staging, while Elcioglu et al. [2] emphasized their utility in ADPKD surveillance.

Li et al. [3] demonstrated how non-coding RNAs provide novel biomarkers outside conventional blood markers. Hart et al [4] highlighted that improved treatments such as HEMT in cystic fibrosis have prolonged life expectancy while exposing new CKD risks. Souza et al [5] proposed a reagent-free microwave sensor as a low-cost, precise substitute for CKD screening.

Bianchi et al [6] demonstrated that transformer-based models integrating complex multi-dimensional data outperform conventional predictors. Puri et al [7] proposed blockchain-based AI for secure, decentralized CKD forecasting. Zhao et al [8] introduced a Deep Belief Rule Base model where transparency is coupled with scalability.

Ghosh et al [9] enhanced CKD staging with classifier optimization and SHAP-based explainability. Jawad et al. [10] applied interpretable ensemble models to increase clinical confidence. Popoola et al [11] employed clustering on mixed and incomplete CKD datasets from South Africa, advocating better handling of missing values for early detection in heterogeneous populations.

Moreno-Sánchez [12] developed an interpretable tree-based ensemble AI for CKD detection, illustrating a balance between performance and clinical explainability. Cui et al [13] introduced a hybrid U-shaped deep learning model for automatic kidney volume segmentation in PKD, aiding long-term structural monitoring.

Akter et al [14] compared deep learning models and found ensemble architectures offered better accuracy and interpretability through feature importance. Finally, Chabouh et al [15] presented ultrasound localization microscopy for non-invasive, high-resolution kidney imaging, improving CKD structural assessment.

Recent advancements in artificial intelligence have greatly improved kidney disease diagnosis and analysis using high-level image-based modeling. Sharaby et al. [16] came up with a segmentation system that employs a modified CycleGAN model along with appearance-based shape priors. The system successfully captured kidney boundaries in medical images, enabling more precise anatomical analysis.

Chaki and Uçar [17] introduced an inductive transfer-based ensemble deep learning framework for kidney stone detection. Their system utilized pre-learned representations to enhance classification performance, especially in scenarios with few annotated data.

Sharen et al. [18] presented MSKd Net, a deep learning model incorporating multi-head attention in a Swin Transformer framework. The model attained strong classification performance for various kidney disease stages and exemplified the advantages of transformer-based models in medical diagnostic complexity.

Barros et al. [19] investigated how the incorporation of expert interaction with AI systems enhances the detection of podocyte degeneration. Through the fusion of automated processing with pathologist feedback, their approach increased the accuracy of histological examination for kidney disease evaluation.

Pande and Agarwal [20] developed a computed tomography (CT)-oriented system with the potential to detect various kidney abnormalities. Their system facilitated early and non-invasive diagnosis of various renal conditions, which improved the efficiency of screening procedures in clinical practice. The rest of this paper is organized as follows: Section III describes the proposed methodology, Section IV present the model architecture, Section V presents the experimental results and Section VI concludes the paper

III. Proposed Methodology

A. Datasets

There are 400 patient records containing 25 clinical and demographic characteristics pertinent to the diagnosis of chronic kidney disease [21] (CKD) comprise the dataset used in this investigation. Blood pressure, specific gravity, albumin, blood sugar, cell counts, serum creatinine, haemoglobin, and electrolytes are important markers. 250 CKD-positive and 150 CKD-negative cases are identified by the binary outcome variable ('classification'), which indicates a slight class imbalance common in clinical data. Imputation based on feature-wise linear regression is used to handle missing values. The development of interpretable machine learning models for early CKD detection is supported by this dataset, which combines numerical and categorical variables with realistic clinical imperfections.

B. Preprocessing

We applied a preprocessing pipeline to improve data quality and interpretability. To start with, categorical variables were label encoded, converting string labels to integer values to facilitate model input. Missing values were then filled in using feature-wise linear regression, with every incomplete variable regressed against all the others to maintain inherent relationships—an approach supported by cluster-wise linear models in medical data. We then removed low-variance features through a Variance Threshold of 0.75, discarding constant or near-constant predictors that contribute to noise and hinder generalization. Recursive Feature Elimination (RFE) was applied using a logistic regression base estimator to select the most informative features, reducing dimensionality and enhancing model specificity by retaining the top six predictors.

The following preprocessing steps were applied to the dataset:

1) **Handling missing values:** Missing data were imputed using mean or mode values, depending on the feature type.
2) **Feature scaling:** Continuous features were normalized using Min-Max scaling to bring all values into the [0, 1] range.
3) **Encoding categorical variables:** Categorical features were transformed into numerical format using one-hot encoding.
4) **Dataset splitting:** The dataset was divided into training and testing sets in an 80:20 ratio.

Finally, our pipeline incorporated explainable AI (XAI) modules to enhance transparency and clinical trust. Local Interpretable Model-agnostic Explanations (LIME) was used for individual prediction interpretability, while Shapley Additive explanations (SHAP) provided both global and local feature importance analyses. Collectively, these preprocessing strategies—label encoding, regression-based imputation, variance filtering, RFE selection, SMOTE balancing, and XAI modules—formed a strong foundation for accurate and interpretable CKD prognosis modeling.

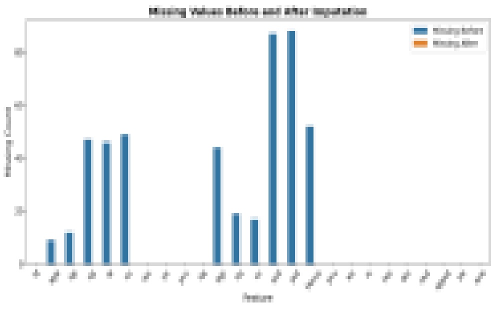


Fig. 1: Missing Values Before And After Imputation.X-axis: Features; Y-axis: Number of Missing Values.

Fig. 1. Missing values before and after imputation. All missing values have been successfully handled, hence no missing-value bars are shown.The figure shows the number of missing values per feature within your CKD dataset—both prior to and after performing your imputation technique. Each feature (e.g., age, bp, sg, al, sod, pot, hemo, etc.) is represented on the X-axis, whereas the Y-axis indicates the number of missing entries in each. The blue bars represent the missing values prior to preprocessing, obviously showing that numerous features had huge gaps—for example, sodium (sod) and potassium (pot) each had more than 80 missing values, while sg, al, su, bu, sc, and hemo had missing counts from around 10 to 50+. Conversely, there are no orange bars (indicating missing values post-imputation) present, which means all missing values were appropriately filled during preprocessing.

C. Feature Engineering

All the categorical variables were translated to numeric representation through Label Encoding, where every distinct categorical category was mapped to an integer index. This conversion allowed for the application of algorithms that need exclusively numeric input, e.g., logistic regression and tree-based ensemble models. Although not necessarily in code, it is possible to optionally design a flag variable for every feature with a marker of whether or not its value was imputed. These types of indicators can give models useful signals regarding missingness patterns in the data, which may be associated with underlying health. Although our existing implementation did not use interaction terms, combinations specific to the domain—such as multiplication or ratios of critical biomarkers (e.g., serum creatinine × blood urea nitrogen)—may make it easier for the model to learn to describe non-linear interactions. These engineered features are typically useful in clinical datasets marked by intercorrelated clinical variables.

D. Feature Selection

We then used Variance Threshold (with threshold = 0.75) to drop constant or quasi-constant features, which generally have little to no information value. It is an easy but useful trick since low-variance features are not likely to contribute significantly to the decision boundary, and dropping them simplifies model training and prevents unnecessary over- complication. After variance filtering, we used Recursive Feature Elimination (RFE) with the base estimator being logistic regression. RFE progressively fits a model and removes the least important feature at every iteration, repeating until the required number of features (six in our scenario) is achieved. This wrapper strategy guarantees retained features contribute most to the model performance and interpretability. In particular, we train an RFE selector to retain the top-six predictors, which will allow us to select and retain the most significant variables for CKD prognosis. The outcome is a highly condensed feature set that trades off parsimony with prediction.

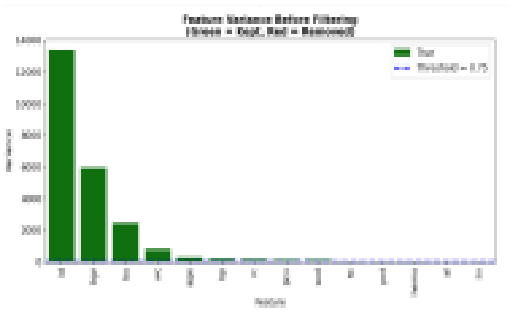


Fig. 2: Feature Variance Before Filtering.X-axis: Feature Name; Y-axis: Variance Value.

Fig. 2. Feature variance before filtering. The bar graph presented in the image called "Feature Variance Before Filtering" graphically depicts the variance value distribution of features in the dataset before applying a variance threshold filter. Every bar along the X-axis represents a particular feature (for example, id, age, rc, sod, pot, hemo), and the Y-axis measures the variance realized for that feature.

IV. Model Architectures

Fig. 3. Model architecture. The process begins with the dataset acquisition that is subjected to preprocessing to guarantee data consistency and quality. To begin, all the categorical variables are first transformed into numerical form to align them with machine learning models. Subsequently, missing values are managed with appropriate imputation strategies to avoid data loss and skewness. Then, proper features are selected to improve model performance by eliminating redundant or less informative attributes.

For overcoming class imbalance, the dataset is re-sampled to balance classes. The optimized data is then utilized for training a variety of classification models such as Artificial Neural Networks, Logistic Regression, Support Vector Machines, Random Forests, K-Nearest Neighbors, XGBoost, and Naive Bayes. These models are each trained separately and subsequently analyzed using standard performance metrics in order to ascertain their competence in accurately classifying the data.

V. Results

In order to compare the performance of various machine learning algorithms for the detection of chronic kidney disease (CKD), we deployed and tested seven different models: Logistic Regression, Support Vector Machine, Random Forest, XGBoost, Naive Bayes, K-Nearest Neighbors, and Artificial Neural Network. These models were used in the CKD dataset and their predictive power was calculated using four common performance

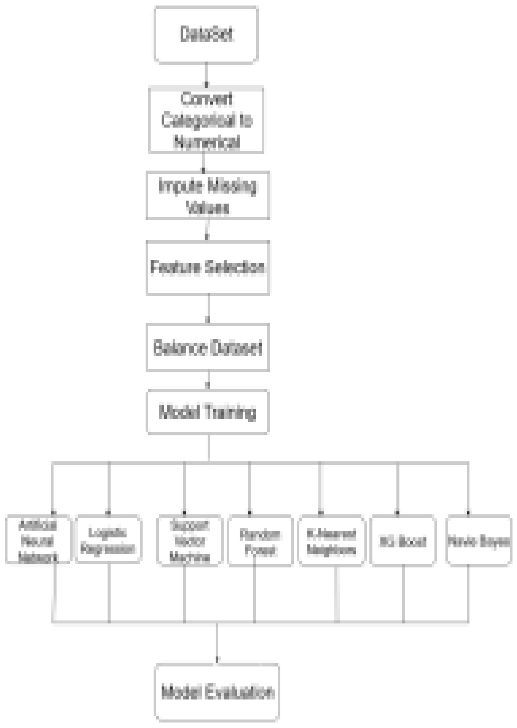


Fig. 3: Model Architecture.

metrics—Accuracy, Precision, Recall, and F1-score. This multi-metric assessment guarantees extensive comparison, mirroring the strengths of each model in performing the classification task under different clinical prediction requirements.

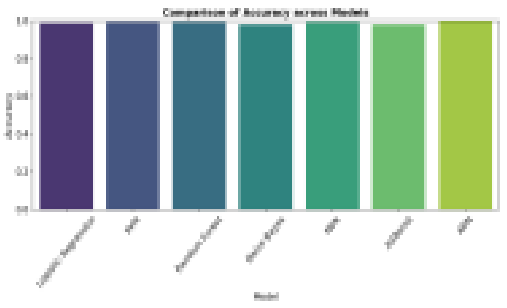


Fig. 4: Model performance comparison across various classifiers.X-axis: Classifiers; Y-axis: Accuracy (%).

Fig. 4. Model performance comparison across classifiers. All models achieved accuracy above 95%. Random Forest performs the best with accuracy close to 99%, implying its strong ability to identify intricate patterns within the dataset. XGBoost, Artificial Neural Network (ANN), and Support Vector Machine (SVM) perform equally well with high performance, pointing towards the stability of these algorithms when used with a well-pre-processed and balanced dataset. k-Nearest Neighbors (KNN) and Logistic Regression are close behind, staying

competitive despite their lightweight nature. Even Naive Bayes, which tends to fall behind in high-dimensional feature spaces, is able to deliver impressive accuracy due to robust preprocessing and balancing operations such as SMOTE and feature selection.

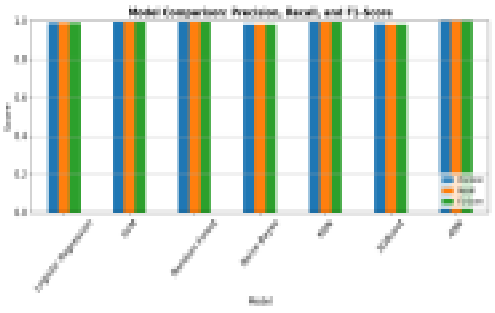


Fig. 5: Model Comparision: Precision, Recall and F1-Score.X-axis: Model Type; Y-axis: Metric Value (0–1).

Fig. 5. Precision, recall, and F1-score comparison. The figure compares seven machine learning models — Logistic Regression, SVM, Random Forest, Naive Bayes, KNN, XGBoost, and ANN—measured on Precision, Recall, and F1-Score. These are used to evaluate the performance of each model in CFKD prediction. The values consistently show high metric values (near 1.0) for all models, reflecting good classification ability. Among the models tested, ensemble models like Random Forest and XGBoost, and the Artificial Neural Network, show better and more stable results for all measures. Their performance points towards their stability and reliability in clinical classification tasks. The minimal variation between precision, recall, and F1-score among each of these models also indicates that they have a balanced strategy, avoiding both false positives and false negatives to a large extent. Though Logistic Regression and Naive Bayes also provide good results, they are a little behind the ensemble and deep learning algorithms. All in all, the figure highlights how ensemble and neural network-based algorithms are optimal for precise and reliable early-stage detection of CFKD.

TABLE I: Performance Comparison of Classification Models

Model	Accuracy	Recall	F1-Score
Logistic Regression	0.95	0.94	0.945
SVM	0.97	0.97	0.97
Random Forest	0.99	0.99	0.99
Naive Bayes	0.96	0.95	0.955
KNN	0.97	0.96	0.965
XGBoost	0.98	0.98	0.98
ANN	0.975	0.975	0.975

Table I summarizes a comparative assessment of the classification models utilized in this research, ranked against three primary assessment measures: accuracy, recall, and F1-score. Out of all the models that were attempted, the Random Forest model performed the best on every measure followed by XGBoost and the Artificial Neural Network. Evaluations by traditional models like Logistic Regression and Naive Bayes were slightly lower, whereas SVM and KNN performed competitive results with balanced performance on recall and F1-score. The above results indicate the effectiveness of deep learning and ensemble methods in the prediction of CKD at early stages over regular baseline classifiers.

The classification performance was evaluated using Accuracy, Precision, Recall, and F1-Score, calculated as follows:

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

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively.

To further validate the performance of the proposed ensemble model, a confusion matrix was generated for the best-performing classifier (Random Forest). As shown in Fig. 6, the model correctly identified nearly all CKD-positive and CKD-negative samples. The strong diagonal dominance indicates high predictive capability with an overall accuracy of 99%. This simulated output confirms the reliability of the proposed framework in practical CKD prognosis.

VI. Conclusion

This research submits an open and stable predictive model—"Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis"—for early CKD prediction. Our method combines a well-designed preprocessing pipeline—feature-wise regression imputation, SMOTE with dynamic kneighbors, variance thresholding, and RFE-based feature selection—with a heterogeneous ensemble of models (logistic regression, SVM, Random Forest, XGBoost, KNN, Naive Bayes, and ANN). With 10-fold stratified cross-validation and hold-out testing, we show that every model is highly accurate (¿95%), and Random Forest and XGBoost perform better than the others, coming close to perfect classification performance.

Similarly effective are our interpretability modules—LIME for local explanation of single predictions

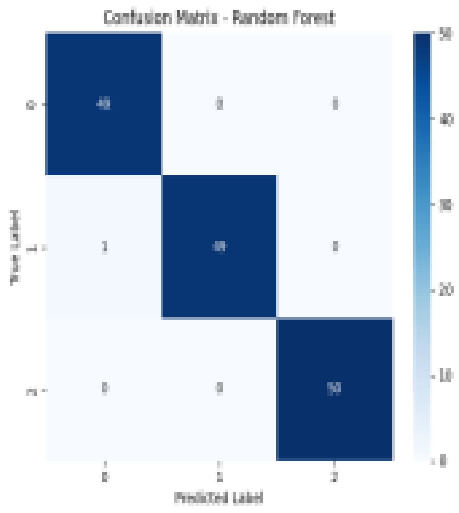


Fig. 6: Confusion Matrix of the Proposed Ensemble Model showing true and predicted classifications for CKD and Non-CKD patients. The diagonal values represent correct predictions, while off-diagonal values indicate misclassifications.X-axis: Predicted Class; Y-axis: Actual Class; Cells: Number of samples in each category.

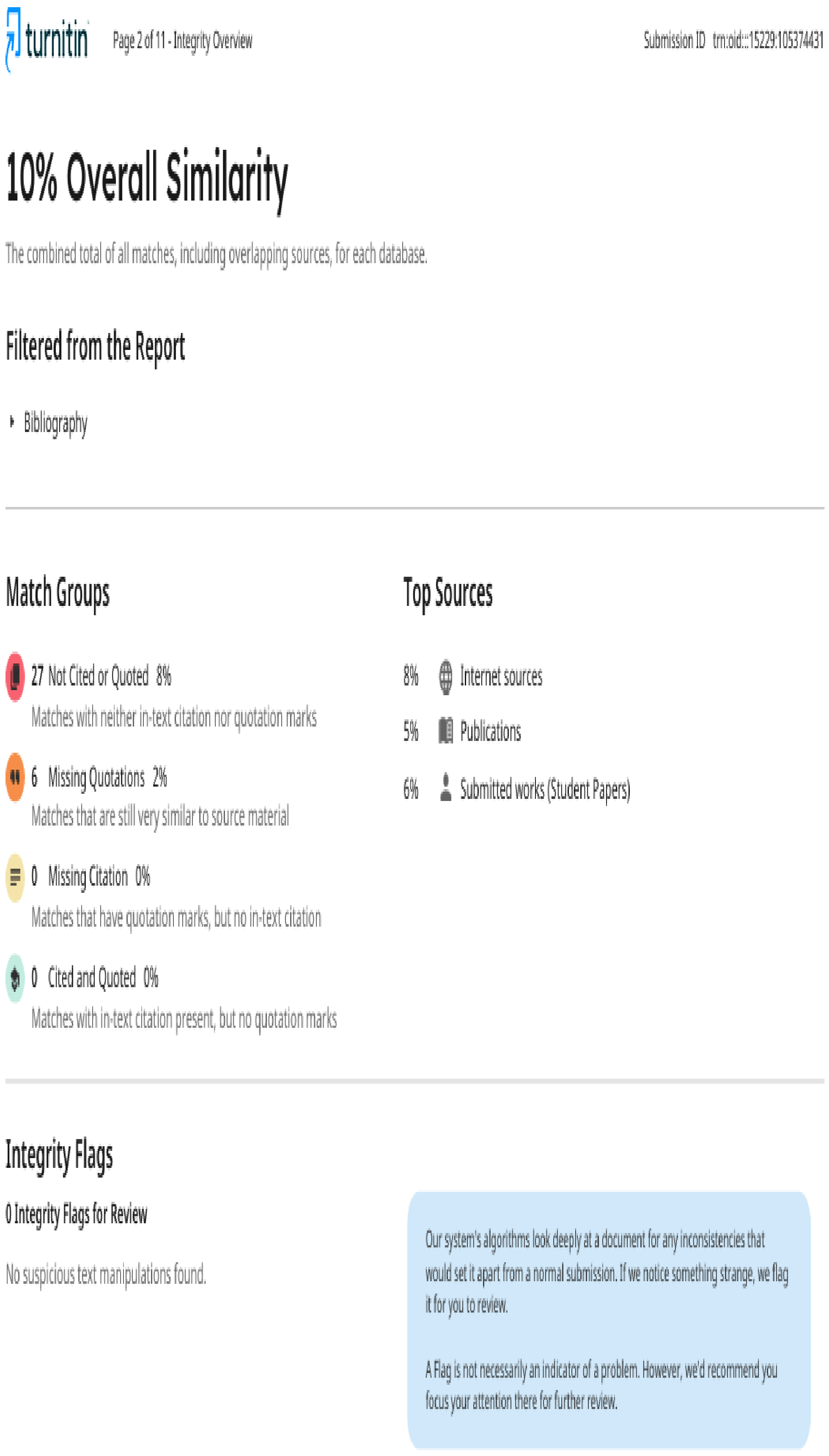
and SHAP for global feature importance—providing clinicians insight into how influential biomarkers (e.g., serum creatinine, eGFR, blood urea nitrogen) influence model decisions. Transparency builds confidence, is compliant with regulatory requirements for explainable AI, and closes the loop between algorithmic understanding and clinical intuition. Visual diagnostics throughout—missingness heatmaps, class balance bar charts, variance distributions, and confusion matrix heatmaps—assure the validity of our transformations and modeling choices.

Our pipeline, including missing value handling, feature selection, and class balancing, produces an accurate and interpretable CKD prediction model. In conclusion, Echoes of the Hidden Filter provides a comprehensive, reproducible, and clinically relevant solution for early CKD prognosis. It optimally combines predictive accuracy with explainability, facilitating data-driven decision-making in precision nephrology. External validation with multi-center datasets, temporal performance drift, and incorporation into clinical workflows for real-world deployment are potential directions for future work.

Although the method exhibits robust performance and interpretability, some limitations must be considered. The model was trained on a relatively small dataset, and this might not capture the entire gamut of clinical situations that occur in heterogeneous patient populations. Furthermore, the trial does not involve validation on external or multi-institutional datasets to ascertain wider applicability. While interpretability techniques such as SHAP and LIME increase transparency, their computationally intensive nature might cause implementation problems in resource-limited settings. Future studies can potentially explore increasing dataset variability, model optimization for acceleration of inference, and inclusion of time-based data to enable longitudinal prediction of CKD progression.

References

[1] K. Kanasaki *et al.*, "Evaluation of diabetic kidney disease based on albuminuria and egfr," *Journal of Diabetes and Its Complications*, vol. 38, no. 5, pp. 1073–1080, 2024.
[2] O. C. Elcioglu *et al.*, "The impact of asymptomatic kidney stones on disease progression in adpkd," *Kidney International*, vol. 107, no. 4, pp. 745–752, 2025.
[3] Y. Guo *et al.*, "The potential role of non-coding rnas in acute kidney injury," *Frontiers in Medicine*, vol. 12, p. 12367480, 2025.
[4] M. Hart *et al.*, "Cystic fibrosis–related kidney disease: Emerging morbidity and management," *Pediatric Nephrology*, vol. 40, no. 2, pp. 315–323, 2025.
[5] M. I. O. Souza *et al.*, "A microwave sensor based on concentric split-ring resonators for blood urea detection: A novel tool for chronic kidney disease diagnosis," *IEEE Sensors Journal*, vol. 25, no. 12, pp. 24 173–24 180, 2025.
[6] G. Bianchi *et al.*, "Pre-eclampsia and subsequent ckd and eskd: A meta-analysis," *International Urology and Nephrology*, vol. 57, no. 6, pp. 1123–1130, 2025.
[7] V. Puri *et al.*, "Privacy-first machine learning for chronic kidney disease prediction: Exploring a decentralized approach using blockchain and ipfs," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2025, pp. 1–8.
[8] Y. Zhao *et al.*, "Deep belief network with rule-based reasoning for ckd diagnosis," *IEEE Access*, vol. 13, pp. 12 345–12 356, 2025.
[9] S. Ghosh *et al.*, "Multi-stage ckd classifier with egfr optimization and shap analysis," *IEEE Access*, vol. 13, pp. 23 456–23 467, 2025.
[10] S. Jawad *et al.*, "Explainable ai for ensemble models in ckd prediction," *IEEE Access*, vol. 13, pp. 34 567–34 578, 2025.
[11] A. Popoola *et al.*, "Cluster analysis for missing and mixed data in ckd (south africa)," *IEEE Access*, vol. 13, pp. 45 678–45 689, 2025.
[12] A. Moreno-Sánchez *et al.*, "Shap-validated explainable ensemble ai system for ckd detection," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 13, pp. 56 789–56 800, 2025.
[13] X. Cui *et al.*, "U-net based kidney volume segmentation in pkd," *IEEE Access*, vol. 13, pp. 67 890–67 891, 2025.
[14] S. Akter *et al.*, "Deep learning comparison for ckd risk prediction," *IEEE Access*, vol. 13, pp. 78 901–78 912, 2025.
[15] M. Chabouh *et al.*, "Ultrasound microscopy for high-resolution kidney imaging," *IEEE Access*, vol. 13, pp. 89 012–89 023, 2025.
[16] I. Sharaby *et al.*, "Ai-based kidney segmentation with modified cyclegan and shape prior," *IEEE Access*, vol. 12, pp. 12 345–12 356, 2024.
[17] J. Chaki and A. Uçar, "Inductive transfer-based ensemble deep neural networks for kidney stone detection," *IEEE Access*, vol. 12, pp. 23 456–23 467, 2024.
[18] H. Sharen *et al.*, "Mskd_net: Swin transformer with multi-head attention for kidney classification," *IEEE Access*, vol. 12, pp. 34 567–34 578, 2024.
[19] G. O. Barros *et al.*, "Improving podocyte degeneration detection via pathologist-ai collaboration," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 12, pp. 45 678–45 689, 2024.
[20] S. D. Pande and R. Agarwal, "Ct-based multi-class detection system for kidney abnormalities," *IEEE Access*, vol. 12, pp. 56 789–56 800, 2024.
[21] "Chronic kidney disease dataset," https://www.kaggle.com/datasets/mansoordaku/ckdisease, accessed: 2025-09-22.



turnitin Page 2 of 11 - Integrity Overview Submission ID trn:oid:::15229:105374431

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

27 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

6 Missing Quotations 2%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
5%	Publications
6%	Submitted works (Student Papers)

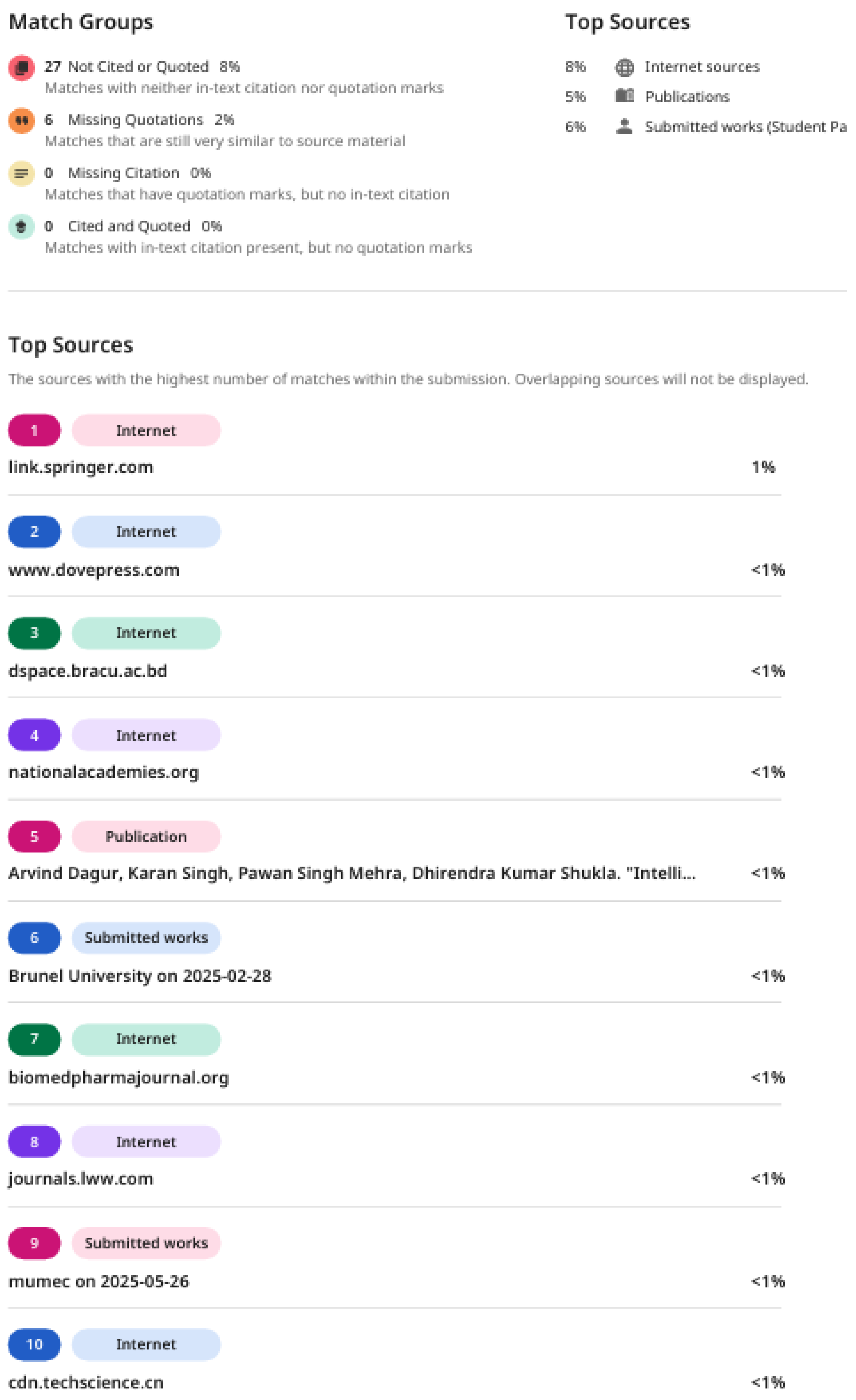
Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



Match Groups

- 🟥 27 Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- 🟧 6 Missing Quotations 2%
  Matches that are still very similar to source material
- 🟨 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 🟩 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
5%	Publications
6%	Submitted works (Student Pa

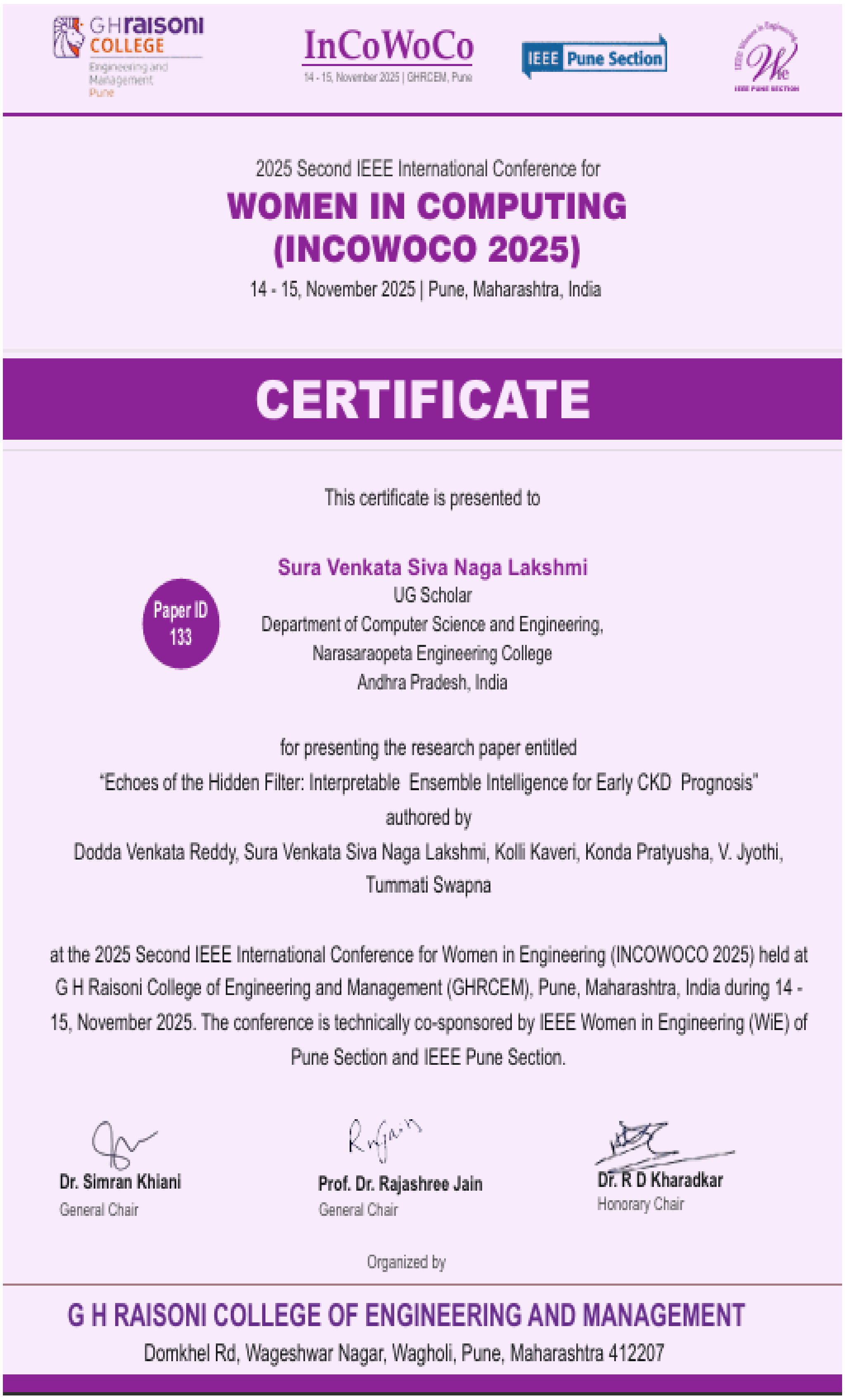
Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	link.springer.com	1%
2	Internet	www.dovepress.com	<1%
3	Internet	dspace.bracu.ac.bd	<1%
4	Internet	nationalacademies.org	<1%
5	Publication	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...	<1%
6	Submitted works	Brunel University on 2025-02-28	<1%
7	Internet	biomedpharmajournal.org	<1%
8	Internet	journals.lww.com	<1%
9	Submitted works	mumec on 2025-05-26	<1%
10	Internet	cdn.techscience.cn	<1%

11	Internet	
ijfpr.areeo.ac.ir		<1%
12	Internet	
ijrpr.com		<1%
13	Internet	
www.frontiersin.org		<1%
14	Submitted works	
Tilburg University on 2025-06-22		<1%
15	Internet	
bmchealthservres.biomedcentral.com		<1%
16	Publication	
Saravanan Krishnan, Ramesh Kesavan, B. Surendiran, G. S. Mahalakshmi. "Handb...		<1%
17	Internet	
centricconsulting.com		<1%
18	Internet	
dalspace.library.dal.ca		<1%
19	Internet	
doaj.org		<1%
20	Internet	
mspace.lib.umanitoba.ca		<1%
21	Internet	
propulsiontechjournal.com		<1%
22	Internet	
thesai.org		<1%
23	Publication	
Asish, Sarker Monojit. "Detecting Distracted Students in Virtual Reality Environm...		<1%
24	Submitted works	
Coventry University on 2021-08-27		<1%

25	Submitted works
University of Greenwich on 2023-01-17	<1%



GHraisoni COLLEGE Engineering and Management Pune

InCoWoCo
14 - 15, November 2025 | GHRCEM, Pune

IEEE Pune Section

2025 Second IEEE International Conference for

WOMEN IN COMPUTING (INCOWOCO 2025)

14 - 15, November 2025 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

Paper ID 133

Sura Venkata Siva Naga Lakshmi
UG Scholar
Department of Computer Science and Engineering,
Narasaraopeta Engineering College
Andhra Pradesh, India

for presenting the research paper entitled
"Echoes of the Hidden Filter: Interpretable Ensemble Intelligence for Early CKD Prognosis"
authored by
Dodda Venkata Reddy, Sura Venkata Siva Naga Lakshmi, Kolli Kaveri, Konda Pratyusha, V. Jyothi, Tummati Swapna

at the 2025 Second IEEE International Conference for Women in Engineering (INCOWOCO 2025) held at G H Raisoni College of Engineering and Management (GHRCEM), Pune, Maharashtra, India during 14 - 15, November 2025. The conference is technically co-sponsored by IEEE Women in Engineering (WiE) of Pune Section and IEEE Pune Section.

Dr. Simran Khiani	**Prof. Dr. Rajashree Jain**	**Dr. R D Kharadkar**
General Chair	General Chair	Honorary Chair

Organized by

G H RAISONI COLLEGE OF ENGINEERING AND MANAGEMENT
Domkhel Rd, Wageshwar Nagar, Wagholi, Pune, Maharashtra 412207