CLASSIFICATION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING CLASSIFIERS

(PROJECT PHASE- II)

submitted in partial fulfillment of the requirements
for the award of the degree in

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

BY

M.RAMESH (211061101253)
M.RANJITH KUMAR (211061101287)
M.GIRISH (211061101262)

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

APRIL 2025

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

DECLARATION FORMAT

We **M. RAMESH(211061101253), M.RANJITH KUMAR (211061101287) , M.GIRISH (21106110262)** hereby declare that the Project Report (Project Phase-II) entitled **"CLASSIFICATION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING CLASSIFIERS"** is done by us under the guidance of Dr./Prof./Mr./Ms. is submitted in partial fulfillment of the requirements for the award of the degree in **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

.

DATE :

PLACE : Chennai

1.

2.

3.

SIGNATURE OF THE CANDIDATE(S)

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report (Project Phase-II) is the Bonafide work of Mr. **M. RAMESH** Reg. No **211061101253**, Mr. **M. RANJITH KUMAR** Reg. No **211061101287**, Mr. **M. GIRISH** Reg. No **21106110262**, who carried out the project entitled **"CLASSIFICATION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING CLASSIFIERS"** under our supervision from December 2024 to April 2025.

Internal Guide	Project Coordinator	Department Head
Mr. S. Praveen Kumar Assistant Professor Dept of CSE Dr M.G.R Educational and Research Institute	**Dr. M. ANAND** Professor **Ms. G. Priyanka** Assistant Professor (Dept of CSE) Dr M.G.R Educational and Research Institute	**Dr. S. Geetha** HOD of CSE Dr M.G.R Educational and Research Institute

Submitted for Viva Voce Examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We would first like to thank our beloved Founder Chancellor **Thiru.Dr. A.C.SHANMUGAM, B.A., B.L.,** President **Er. A.C.S.Arunkumar, B.Tech., M.B.A.,** and Secretary **Thiru A.RAVIKUMAR** for all the encouragement and support extended to us during the tenure of this project and also our years of studies in his wonderful University.

We express my heartfelt thanks to our Vice Chancellor **Prof. Dr. S. GEETHALAKSHMI** in providing all the support of my Project (Project Phase-II).

We express my heartfelt thanks to our Head of the Department, **Prof. Dr. S. Geetha**, who has been actively involved and very influential from the start till the completion of our project.

Our sincere thanks to our Project Coordinators **Dr. M. ANAND , Ms. G. Priyanka** and Project guide **Mr. S. Praveen Kumar** for their continuous guidance and encouragement throughout this work, which has made the project a success.

We would also like to thank all the teaching and non-teaching staffs of Computer Science and Engineering department, for their constant support and the encouragement given to us while we went about to achieving my project goals.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF FIGURES

ABSTRACT:

Chronic kidney disease (CKD) is a common and serious medical condition that affects millions of people worldwide. Accurate classification of CKD patients into different stages is critical for effective treatment and management. Machine learning (ML) has emerged as a promising tool for CKD classification, with the potential to improve the accuracy and efficiency of diagnosis. In this study, we evaluate the performance of several popular ML classifiers, including KNN, decision tree, Random Forest, and support vector machine, in classifying CKD patients into different stages based on their clinical and laboratory data. Our results demonstrate that ML classifiers can achieve high accuracy in CKD classification, with support vector machine outperforming other classifiers with an accuracy of 86.5%. The findings of this study have important implications for the development of automated CKD diagnosis and decision support systems.

In addition to evaluating the performance of different ML classifiers in CKD classification, this study also investigates the important features that contribute to accurate classification. Our results indicate that age, hemoglobin levels, and White Blood Cell Count are the most important features for CKD classification. Furthermore, we demonstrate that feature selection can significantly improve the performance of ML classifiers in CKD classification, with an accurate improvement of up to 4.7% using a genetic algorithm-based feature selection method. These findings highlight the potential of ML in identifying important features for CKD diagnosis and management and provide insights for the development of more effective CKD classification models. Overall, our study contributes to the growing body of research on the application of ML in medical diagnosis and management and has important implications for improving the accuracy and efficiency of CKD diagnosis and treatment.

MAJOR DESIGN CONSTRAINTS AND DESIGN STANDARDS TABLE

Student Group	M.RAMESH (211061101253)	M.RANJITH (211061101287)	M.GIRISH (211061101262)
Project Title	Classification Of Chronic Kidney Disease Using Machine Learning Classifiers		
Program Concentration Area	Data Collection, Data Preprocessing, Machine Learning Classifiers, Model Training and Validation		
Constraints Example	Power Constraints		
Economic	No		
Environmental	No		
Sustainability	No		
Implementable	No		
Ethical	Yes		
Health and Safety	No		
Social	No		
Political	No		
Other	Data – Related, Technical, Project Management, Legal & Regulatory		
Standards			
1	HL7, FHIR, DICOM		
2	ISO/IEC 270001, NIST		
3	GDPR, HIPAA, CDISC		
Prerequisite Courses for the Major Design Experiences	1. Data Science and Analytics 2. Medical Imaging 3. Data Collection and Preprocessing		

CHAPTER – I

INTRODUCTION:

Chronic Kidney Disease (CKD) is considered an important threat to society with respect to health in the present era. Chronic kidney disease can be detected with regular laboratory tests, and some treatments are present which can prevent development, slow disease progression, reduce complications of decreased Glomerular Filtration Rate (GFR) and risk of cardiovascular disease, and improve survival and quality of life. CKD can be caused due to lack of water consumption, smoking, improper diet, loss of sleep, and many other factors. This disease affected 753 million people globally in 2016 of which 417 million are females and 336 million are males. Majority of the time the disease is detected in its final stage, and which sometimes leads to kidney failure.

To predict positive CKD status and the stages of CKD machine learning can be used. Machine Learning grabs a major part of artificial intelligence when it comes to doing predictions from previous data using classification and regression methods. Application of machine learning methods to predict CKD has been explored based on multiple data sets.

Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then excreted in your urine. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes, and wastes can build up in your body. In the early stages of chronic kidney disease, you may have few signs or symptoms. Chronic kidney disease may not become apparent until your kidney function is significantly impaired. Treatment for chronic kidney disease focuses on slowing the progression of kidney damage, usually by controlling the underlying cause. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant.

Every year, there are approximately 10 lakh cases of chronic kidney disease in India. Chronic Kidney Disease can be detected by regular laboratory tests. There are some treatments to stop the development. This disease can cause permanent kidney failure. If CKD is cured in early-stage then the person can show symptoms like Blood Pressure, Anaemia, poor health, and weak bones, and since the kidney starts to function improperly, the throw-out of waste in the person's body will be minimal. Hence it is essential to detect CKD at its early stage, but some people have no

symptoms. So, machine learning can be helpful to predict whether a person has CKD or not. Glomerular Filtration Rate (GFR) is the best test to measure the level of kidney functionality and can determine the stage of chronic kidney disease. There are five stages of damage severity based on GFR.

Only after stage 2 of CKD, the patient will get to know about the reduction of kidney functionality. The early detection of CKD can reduce the chance of CKD for the patient. With the advancement in machine learning and artificial intelligence, several classifiers and clustering algorithms are being used to achieve this. This research presents the use of machine learning algorithms for the prediction of chronic kidney disease. The dataset used for building the predictive models in this research are available and can be downloaded from the UCI machine learning library. The data is imported in CSV format and cleaned for use. After the dataset is pre-processed and the best attributes are selected, machine learning algorithms including Random Forest, Supper vector machine, KNN and Decision Tree, are used for the prediction of chronic kidney disease.

1.1 Parameters:

The Parameters used in chronic kidney disease are as follows:

- **Specific Gravity:** This parameter measures the concentration of solutes in the urine and can indicate dehydration or kidney damage.
- **Red Blood Cells:** This parameter measures the number of red blood cells in the urine and can indicate bleeding in the urinary tract or kidney damage.
- **Plus Cell Clumps:** This parameter measures the presence of cellular debris in the urine and can indicate inflammation or infection in the kidneys.
- **Bacteria:** This parameter measures the presence of bacteria in the urine and can indicate a urinary tract infection.
- **Blood Glucose Random:** This parameter measures the level of glucose in the blood and can indicate the presence of diabetes.
- **Haemoglobin:** This parameter measures the amount of haemoglobin in the blood and can indicate Anaemia or kidney damage.
- **Packed Cell Volume:** This parameter measures the proportion of red blood cells in the blood and can indicate dehydration or kidney damage.
- **White Blood Cell Count:** This parameter measures the number of white blood cells in the blood and can indicate inflammation or infection in the body.

- **Red Blood Cell Count:** This parameter measures the number of red blood cells in the blood and can indicate Anaemia or kidney damage.
- **Hypertension:** This parameter measures blood pressure and can indicate the presence of hypertension, which is a risk factor for CKD.
- **Diabetes Mellitus:** This parameter measures the presence of diabetes, which is a risk factor for CKD.
- **Coronary Artery Disease:** This parameter measures the presence of coronary artery disease, which is a risk factor for CKD.
- **Appetite:** This parameter measures the patient's appetite and can indicate malnutrition or kidney damage.
- **Pedal Edema:** This parameter measures the swelling of the feet and can indicate fluid retention or kidney damage.

CHAPTER – II

LITERATURE SURVEY:

TITLE: Survey on Chronic Kidney Disease Prediction System with Feature Selection and Feature Extraction Using Machine Learning Technique

Author: Ajay Kumar, Karthik Raja, Jebaz Sherwin, Revathi

To predict An intelligent system named Chronic Kidney Disease Prediction System (CKDPS) has been developed to predict Chronic Kidney Disease (CKD) early, helping doctors diagnose it before kidney failure. CKDPS uses minimal and relevant features for prediction, employing three feature selection algorithms and two feature extraction algorithms. Various Machine Learning algorithms were tested, and the Random Forest algorithm was chosen for its superior performance, achieving 95% accuracy, precision, and recall. This system aims to simplify and enhance the prediction of CKD for medical professionals.

TITLE: Chronic Kidney Disease Prediction Using Data Mining and Machine Learning

Author: Adeeba Azmi, Amiksha Hingu, Ruchi Dholaria, Ms. Alvina Alphonso

To predict Chronic Kidney Disease (CKD) by entering symptoms using Data Mining and Machine Learning techniques. It employs KNN and SVM Ensemble methods. The SVM with RBF kernel is used for Data Mining, achieving 87% accuracy, while KNN with hyperparameters is used for Machine Learning, achieving over 92% accuracy. The dataset used has 400 columns and 24 attributes. The ensembling technique enhances the accuracy of the predictions.

TITLE: Chronic Kidney Disease Diagnosis Using Machine Learning

Author: Dr. Vijayaprabakaran, Pratheek Reddy, Puthin Kumar Reddy, Munnaf, Reddi Prasad

To predict Chronic Kidney Disease (CKD) is a global health issue with many unaware they have it until it's too late. Early prediction is crucial. This research collects patient blood pressure and diabetes data, using machine learning techniques like Random Forest, XGradient Boost, and Support Vector Machines to detect CKD early. The CKD dataset is used to predict whether a person has CKD.

TITLE: Chronic Kidney Disease Prediction Using Neural Networks

Author: S. Priya, S. Nirmal Kumar, G. Sibi Saravanan, E. Pandiyan, Ashuthosh Kumar Pandey

To predict classify Chronic Kidney Disease (CKD) using machine learning, specifically an artificial neural network (ANN) with the Keras Python Library for sequential model creation. The model employs a feed-forward network with a backpropagation algorithm. This system assists medical practitioners and helps patients detect CKD or the risk of it early using medical data without advanced equipment.

TITLE: Chronic Kidney Disease Prediction Using Machine Learning

Author: Reshma, Salma Shaji, S R Ajina, Vishnu Priya S, Janisha A

To predict Chronic Kidney Disease (CKD), also known as Chronic Renal Disease, is a long-term kidney function issue. Early prediction is vital for effective treatment. This study uses machine learning techniques like Ant Colony Optimization (ACO) and Support Vector Machine (SVM) classifiers to predict CKD using minimal features. The final output indicates whether a person has CKD or not.

CHAPTER – III

SYSTEM ANALYSIS:

3.1 EXISTING SYSTEM:

Chronic kidney disease (CKD) is a prevalent and serious medical condition that affects millions of people worldwide. Accurate classification of CKD patients into different stages is essential for appropriate treatment and management. Traditional classification methods rely on clinical and laboratory data and may be subject to interobserver variability, leading to inaccuracies in diagnosis and treatment. Therefore, there is a growing interest in applying machine learning (ML) techniques to improve the accuracy and efficiency of CKD classification.

Several studies have investigated the use of ML classifiers for CKD classification, with promising results. The performance of ML algorithm, including naive bayes, and Decision Tree, in classifying CKD patients into various stages based on clinical laboratory data. The study found that the support vector machine algorithm had the highest accuracy of 89.4% in classifying CKD patients into various stages. The ML models based on support vector machine and random forests had higher accuracy than traditional classification methods for CKD diagnosis.

Despite these promising results, some challenges remain in the application of ML for CKD classification. For instance, the quality and completeness of the input data may impact the performance of ML classifiers. Additionally, the interpretation of ML models may be challenging, which could limit their use in clinical practice. Therefore, further research is needed to address these challenges and validate the performance of ML classifiers in real-world clinical settings.

3.1.1 Disadvantages of Existing System

It is important to keep in mind that one of the main drawbacks of using machine learning for chronic kidney disease (CKD) classification is the potential for overfitting. Overfitting occurs when a model is too complex and captures noise in the data rather than the underlying pattern, resulting in poor generalization to new data. Naïve Bayes is a commonly used algorithm for CKD classification, but it is known to be prone to overfitting when the input features are highly correlated. For example, if serum creatinine and estimated glomerular filtration rate are highly correlated, the Naïve Bayes algorithm may assign too much weight to these features, leading to overfitting and poor generalization. To address this issue, feature selection and regularization techniques can be applied to reduce the complexity of the model and improve generalization.

Using machine learning classifiers for CKD diagnosis also has a disadvantage, which is the lack of interpretability. In many cases, the models developed by machine learning algorithms are considered "black boxes" because it is difficult to understand how they arrive at their predictions. This can make it challenging for clinicians to trust the results and incorporate them into their decision-making process. Decision trees are another commonly used algorithm for CKD classification, but they can be difficult to interpret and prone to overfitting when the tree becomes too complex. To improve interpretability, decision tree models can be pruned or simplified, but this can result in reduced accuracy and generalization. Therefore, it is important to strike a balance between interpretability and performance when using machine learning classifiers for CKD classification.

3.2 Proposed System

The proposed system for chronic kidney disease (CKD) classification using machine learning (ML) classifiers aims to overcome the limitations of existing systems and improve the accuracy and efficiency of CKD diagnosis and management. The proposed system will use a combination of ML algorithms, including random forest, gradient boosting, and support vector machine, to classify CKD patients into different stages based on clinical and laboratory data. Additionally, the proposed system will incorporate feature selection and dimensionality reduction techniques to improve the performance of the ML classifiers and reduce overfitting.

One of the key features of the proposed system is the use of a hybrid ML approach that combines the strengths of different algorithms to improve accuracy and generalization. For example, random forest and gradient boosting are ensemble learning methods that combine multiple decision trees to improve prediction accuracy and reduce overfitting. On the other hand, support vector machine is a kernel-based algorithm that can handle non-linear data and is less prone to overfitting. By combining these algorithms, the proposed system aims to achieve higher accuracy and better generalization compared to using a single algorithm.

Another feature of the proposed system is the use of feature selection and dimensionality reduction techniques to improve the performance of the ML classifiers. Feature selection involves identifying the most informative features for CKD classification and discarding irrelevant or redundant features. Dimensionality reduction techniques, such as principal component analysis, can be used to reduce the dimensionality of the input data and improve the efficiency of the ML classifiers. These techniques can help to reduce overfitting, improve generalization, and enhance interpretability of the ML models.

Overall, the proposed system for CKD classification using ML classifiers represents a significant improvement over existing systems by combining the strengths of multiple algorithms and incorporating feature selection and dimensionality reduction techniques. The system has the potential to improve the accuracy and efficiency of CKD diagnosis and management and facilitate the development of automated CKD diagnosis and decision support systems. Further research and validation of the proposed system in real-world clinical settings are needed to assess its effectiveness and impact on patient outcomes.

3.2.1 Advantages of Proposed System

The proposed system for CKD classification using ML classifiers has several advantages over existing systems. Firstly, by combining multiple ML algorithms, the proposed system can leverage the strengths of each algorithm to improve accuracy and generalization. This means that the proposed system can handle a wider range of data types and patterns than a single algorithm, resulting in more accurate and reliable predictions. Additionally, incorporating feature selection and dimensionality reduction techniques can help to reduce overfitting and improve interpretability, making the system more useful for clinical decision- making.

The proposed system has the potential to improve the efficiency and speed of CKD diagnosis and management. With the increasing prevalence of CKD, there is a growing need for automated and efficient diagnosis and decision support systems. The proposed system can help to streamline the diagnosis process by automating the classification of CKD patients based on clinical and laboratory data, reducing the burden on healthcare professionals and improving patient outcomes. Moreover, the proposed system can be used for continuous monitoring of CKD patients, allowing for early detection of disease progression and timely intervention to prevent further damage to the kidneys.

CHAPTER – IV

MODULE DESCRIPTION:

4.1 Modules

4.1.1 Data Collection

The data are collected from UCI Machine Learning Repository, and it predicts CKD based on the given attributes. The dataset has fourteen attributes which predict the CKD. The dataset is built on both numerical and nominal data types. As per the UCI's CKD dataset, this contains the attribute such as Specific Gravity, Red Blood Cells, Plus Cell Clumps, Bacteria, Blood Glucose Random, Hemoglobin, Packed Cell Volume, White Blood Cell Count, Red Blood Cell Count, Hypertension, Diabetes Mellitus, Coronary Artery Disease, Appetite, Pedal edema.

4.1.2 Formatting

You may not have chosen the details in a format that suits you for working with. The data may also be in an electronic database, and you would like it to be in a spreadsheet, or the information may be in a proprietary file format, and you would like it to be in an electronic database or folder.

4.1.3 Cleaning

Cleaning data is the eradication or restoration of unfinished or empty data. There may also be incomplete occurrences of data which do not carry the information that you think you'd like to lever may need to eliminate these occurrences. In addition, there are attributes which carry sensitive information and that the attributes are likely to be omitted.

4.1.4 Pre-Processing

Data pre-processing is a part of machine learning, which involves transforming raw data into a more coherent format. Raw data is usually inconsistent or incomplete and usually contains many errors. The data pre-processing involves checking out missing values, looking for categorical values, splitting the dataset into training and test set and finally doing a feature scaling to limit the range of variables so that they can be compared on common environments. In this paper we have used null () method for checking null values and label Encoder () for converting the categorical data into numerical data.

4.1.5 Feature Extraction

A new method to detect malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself. Features used to classify are the presence of tag uses-permission and uses feature into the manifest as well as the number of permissions of each application. These features are the permission requested individually and the «uses feature» tag. The possibility of detecting malicious Android applications based on permissions and twenty features from Android application packages.

4.1.6 Training the Machine

Training the machine is like feeding the data to the algorithm to touch up the test data. The training sets are used to tune and fit the models. The test sets are untouched, as a model should not be judged based on unseen data. The training of the model includes cross- validation where we get a well-grounded approximate performance of the model using the training data.

4.1.7 Split the Dataset into Train and Test Set

This step includes training and testing of input data. The loaded data is divided into two sets, such as training data and test data, with a division ratio of 80% or 20%, such as 0.8 or 0.2. In a learning set, a classifier is used to form the available input data. In this step, create the classifier's support data and preconceptions to approximate and classify the function. During the test phase, the data is evaluated. The final data is formed during pre-processing and is processed by the machine learning module.

It is type of ensemble learning method and also used for classification and regression tasks. The accuracy it gives is greater than compared to other models. This method can easily manage large datasets. It is a popular ensemble Learning Method. Random Forest Improve Performance of Decision Tree by reducing variance. It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the class or classification or mean prediction (regression) of the individual trees.

Algorithm:

- The first step is to select the “R” features from the total features “m” where R<<M
- Among the “R” features, the node using the best split point.
- Split the node into sub nodes using the best split.
- Repeat a to c steps until “l” number of nodes has been reached.
- Built forest by repeating steps a to do for “a” number of times to create “n” number of trees.

4.1.8 Data Splitting

For each experiment, we split the entire dataset into 70% training set and 30% test set. We used the training set for resampling, hyper parameter tuning, and training the model and we used test set to evaluate the performance of the trained model. While splitting the data, we specified a random seed (any random number), which ensured the same data split every time the program executed.

4.1.9 Flow Chart

The approach performs chronic kidney disease classification using machine learning classifiers. The methodology flow begins with the source data, which is a clinical dataset. Followed by pre-processing and cleaning, then splitting the dataset into training and testing datasets. Training data is processed by predefined algorithms, then the testing data is followed by the user inputs, and then the algorithm will be able to determine whether the patient is CKD or NON-CKD from the User Inputs.

Figure 4.1.9: Flow Diagram of chronic kidney disease classification using machine learning classifiers

SYSTEM DESIGN:

4.2 Design Methodology:

The design of the CKD classification system using machine learning classifiers involves the development of a user-friendly and interactive web-based interface that allows clinicians and healthcare providers to input patient data, perform CKD classification, and view the results. To achieve this, we have used several web development technologies such as HTML, CSS, and JavaScript to design the front-end interface of the system. The front-end interface provides a simple and intuitive interface for users to input patient data, view the results, and interpret the ML models. Additionally, we have used Flask, a Python-based web development framework, to implement the back-end functionality of the system. Flask provides a lightweight and flexible framework for developing web applications and allows us to integrate ML models into the web application.

The front-end interface of the system has been designed using HTML, CSS, and JavaScript to provide a responsive and user-friendly interface. HTML is used to structure the content of the web page, CSS is used to style the elements of the web page, and JavaScript is used to add interactivity and functionality to the web page. The interface includes several input fields for clinicians to input patient data such as as Specific Gravity, Red Blood Cells, Plus Cell Clumps, Bacteria, Blood Glucose Random, Hemoglobin, Packed Cell Volume, White Blood Cell Count, Red Blood Cell Count, Hypertension, Diabetes Mellitus, Coronary Artery Disease, Appetite, Pedal edema. The interface also includes a section for displaying the results of the CKD classification using the ML models, which includes the CKD stage and the probability of each stage. The interface has been designed to be accessible and user-friendly, with clear labels and instructions to guide users through the process of inputting patient data and interpreting the results.

The design of the CKD classification system using ML classifiers is a critical aspect of the development process, as it determines the usability and functionality of the system. By using HTML, CSS, JS, Python, and Flask, we can create a comprehensive and efficient system that can accurately classify CKD patients and provide valuable insights for healthcare professionals. The design also allows for the integration of additional features and functionalities in the future, such as data visualization and decision support tools.

4.2.1 Block Diagram

A block diagram is a visual representation of a system or process that shows the major components or stages of the system and how they are connected or interact with each other. In the context of CKD classification using machine learning classifiers.

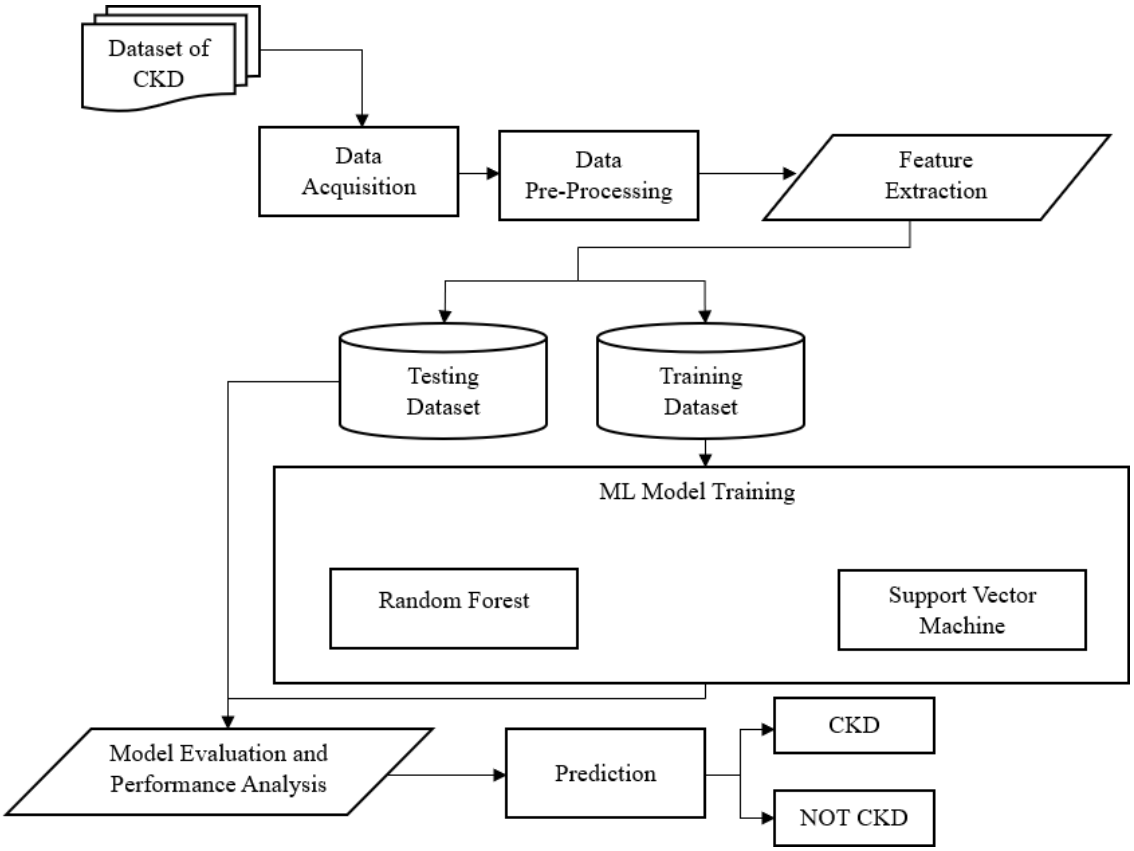


Figure 4.2.1: Block diagram

4.2.2 Data Flow Diagrams:

A DFD is a graphical representation of how the data flows through a system. Developing a DFD is one of the first steps conducted when developing an information system. DFD displays details like the data that is coming in and going out of the system, how the data is travelled through the system and how the data will be stored in the system. But the DFD does not contain information about timing information of the processes. The main components included in a DFD are processes, data stores, data flow and external entities. When developing DFD diagrams, the context level DFD is drawn first. It displays how the entire system interacts with external data sources and data sinks. Next a Level 0 DFD is developed by expanding the It also contains details about the data stores required within the system. entire work or the flow of the data can be divided into three groups for better understanding. They are- 1. DFD-L0 2. DFD-L1 3. DFD-L2.

Figure 4.2.2.1: Level 0 DFD

This is the initial idea for the flow of data. The data must be flown from user to server and from server to the user for the prediction of the disease by entering details and sending the data. Communication is done between user and the server. Users, the main process, and data flow make up its parts. Also, the project concept is demonstrated using the single process visualization. DFD Level 0 shows the entities that interact with a system and defines the border between the system and its environment. The illustration presents the main process in a single node to introduce the project context. This context explains how the project works in just one look. The user feeds data into the system and then receives the output/report from it. In addition to this, you will perceive through the diagram that there is already the presence of data flow. Though the process is very general, the flow of data is clear. Nevertheless, just modify this diagram to meet the other requirements and include other matters regarding Chronic Kidney Disease Classification Using Machine Learning Classifiers.

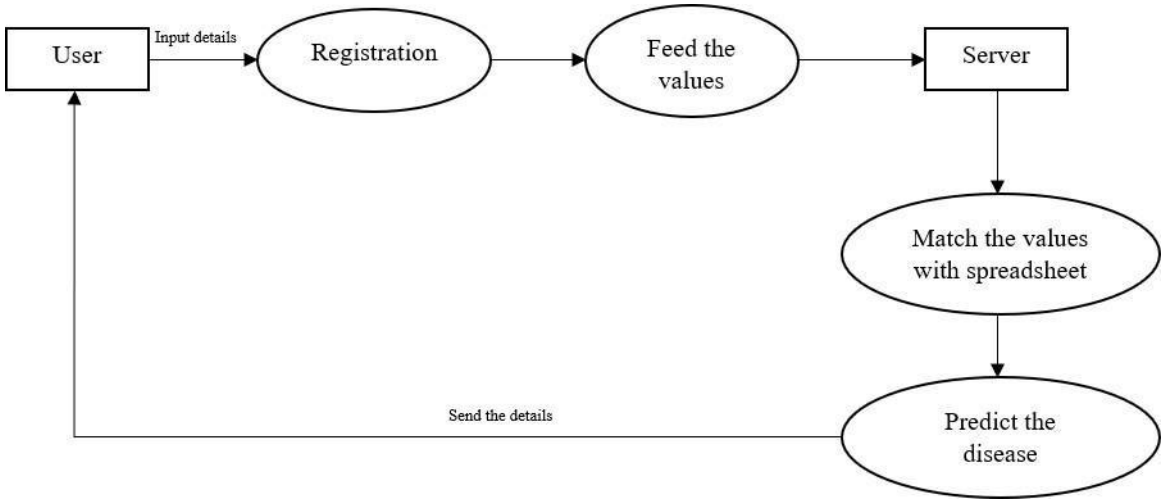


Figure 4.2.2.2: Level 1 DFD

This is the process or the idea where the data has been used to predict the disease by following several steps like registration (for new users), Feed the values (entering and storing values), Server (to store them), match the values (Finding probability) and finally predict the disease (Final result). The registered users can login to their account and can enter the values that is data and then can store them in the spreadsheet with the help of the server and then extract those values and find probability and then generate the report as similar to the newly registered users.

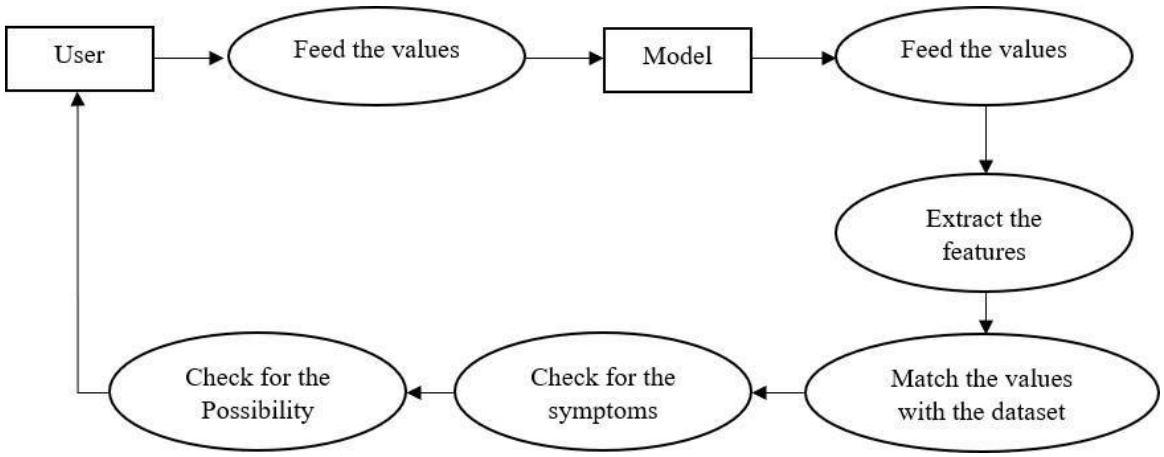


Figure 4.2.2.3: Level 2 DFD

The data, which is flown from the user to the server, there it undergoes matching for data from the user(input) and the data which we have i.e., datasets (train data). Finding probability between them by comparing the values and then generating the report. The DFD level 2 shows the processes involved in the machine learning system. The system starts with the input of patient data (symptoms and medical history) from the patient entity. The data is then processed through various stages, including data pre-processing, feature extraction, model selection, model training, model evaluation, model tuning, and model deployment. The pre- processed data is stored in the Pre-processed Datastore, and the trained model is stored in the Trained Model store. The system then outputs the predicted disease diagnosis to the patient entity. Overall, DFD level 2 provides a detailed illustration of how data flows through the machine learning system, from the input of patient data to the output of predicted disease diagnosis.

4.2.3 Unified Modelling Language Diagram:

UML is a modelling language used in object-oriented software design. UML provides capabilities to specify and visualize the components of a software system. UML diagrams mainly represent the structural view and the behavioral view of a system. The structural view of the system is represented using diagrams like class diagrams, composite structure diagrams, etc. A dynamic view of the system is represented using diagrams such as sequence diagrams, activity diagrams, etc. UML version 2.2 includes fourteen diagrams, which include seven diagrams representing the structural view and other seven representing the behavioral view. Among the seven behavioral diagrams, four diagrams can be used to represent interactions with the system. There are tools that can be used for UML modelling such as IBM Rational Rose.

4.2.4 Use case Diagram:

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor and the use case. To depict the system boundary, draw a box around the use case itself. UML Use case diagrams are ideal for:

- Representing the goals of system-user interactions.
- Defining and organizing functional requirements in a system.
- Specifying the context and requirements of a system.
- Modelling the basic flow of events in a use case.

A useful case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

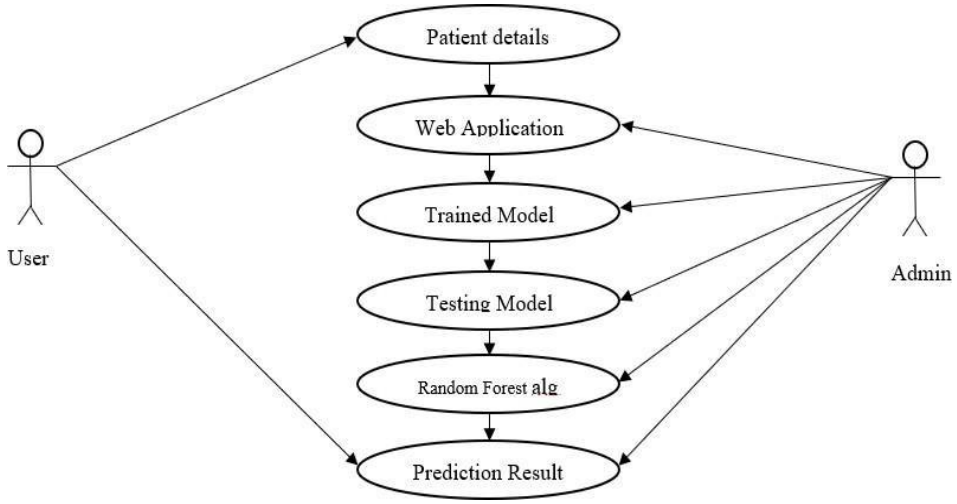


Figure 4.2.4: Use Case Diagram

4.2.5 Class Diagram:

The class diagram is a static diagram. It represents the static view of an application. The class diagram is not only used for visualizing, describing, and documenting various aspects of a system but also for constructing executable code of the software application. A class diagram describes the attributes and operations of a class and the constraints imposed on the system. Class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. The purpose of a class diagram is to model the static view of an application. Class diagrams are the only diagrams that can be directly mapped with object-oriented languages and are thus widely used at the time of construction. UML diagrams like activity diagrams, and sequence diagrams can only give the sequence flow of the application, however, the class diagram is a bit different. It is the most popular UML diagram in the coder community. The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe the responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.
- The class diagram helps construct the code for the software application development.

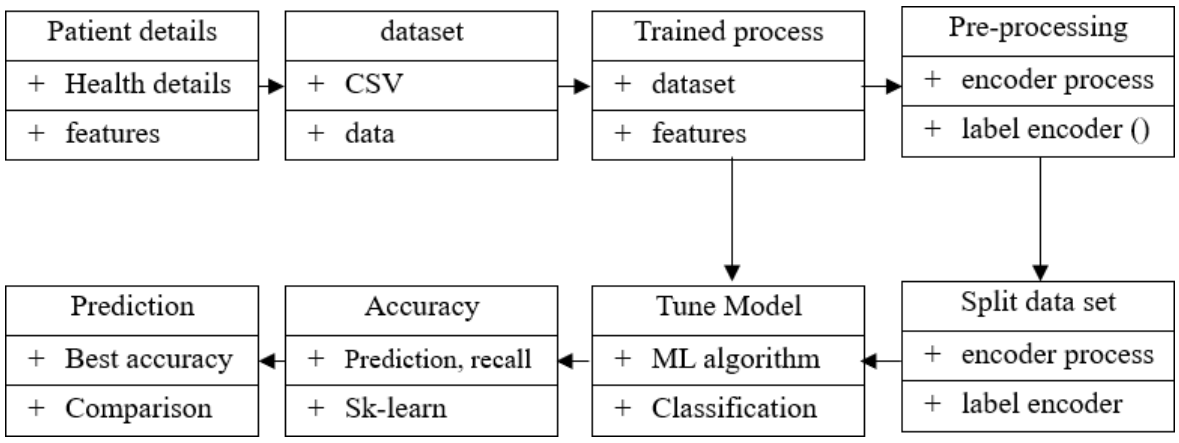


Figure 4.2.5: Class diagram

4.2.6 Sequence Diagram:

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

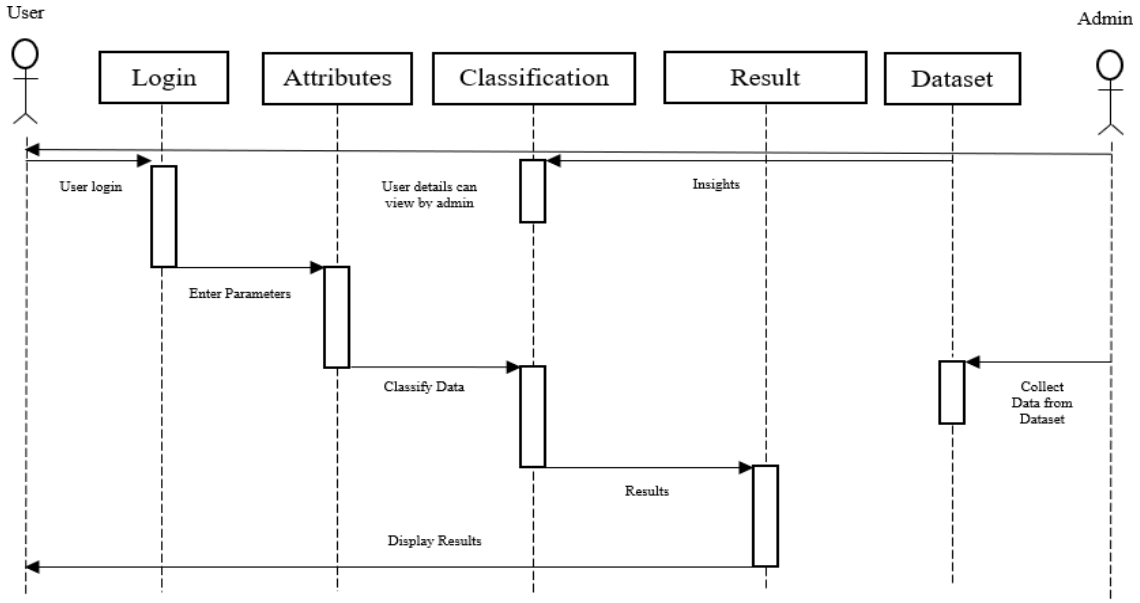


Figure 4.2.6: Sequence diagram

4.2.7 Component Diagram:

A component diagram is used to break down a large object-oriented system into the smaller components, to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

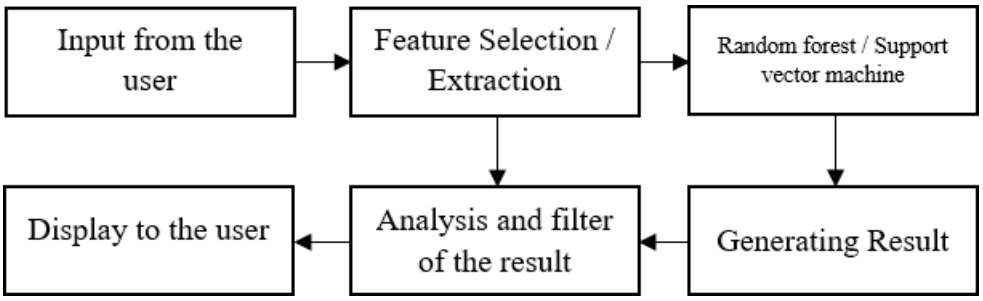


Figure 4.2.7: Component diagram

4.2.8 Activity Diagram:

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram is a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams are used to show the message flow from one object to another, but the activity diagram is used to show the message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only thing missing in the activity diagram is the message part. It does not show any message flow from one activity to another. An activity diagram is sometimes considered a flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. The purpose of an activity diagram can be described as:

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched, and concurrent flow of the system.

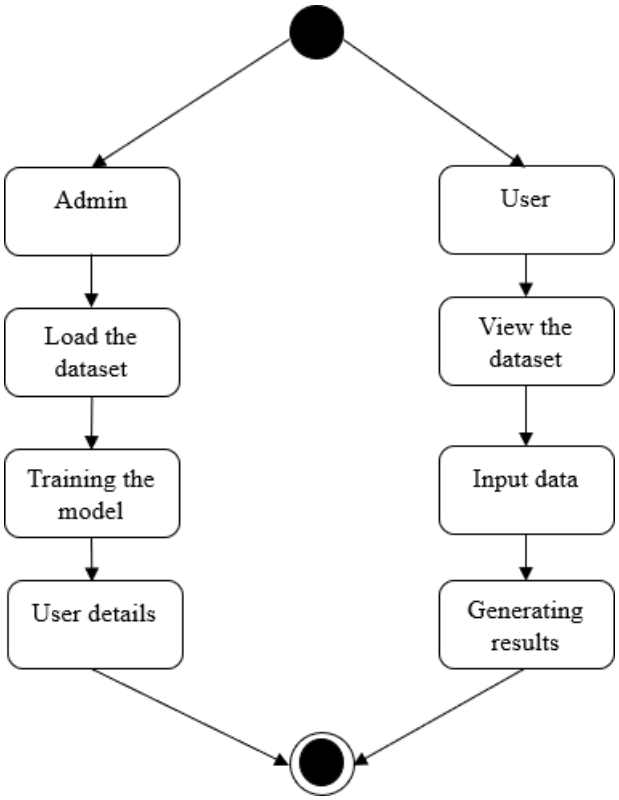


Figure 4.2.8: Activity diagram

4.2.9 Software Development Life Cycle – SDLC:

In our project we use the waterfall model as our software development cycle because of its step-by-step procedure while implementing**.**

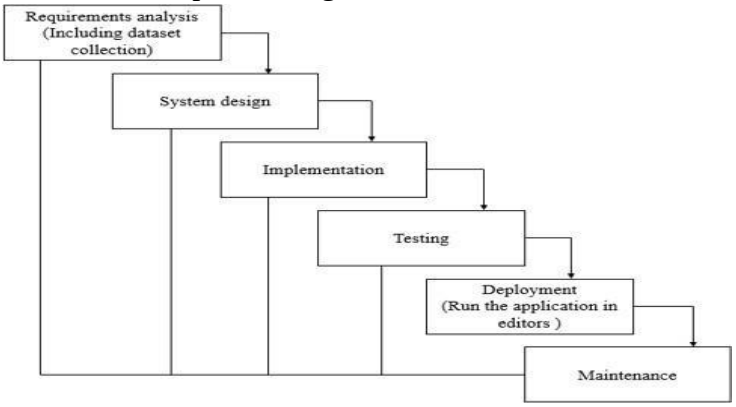


Figure 4.2.9: Waterfall Model

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design:** The requirement specifications from the first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware

and system requirements and helps in defining the overall system architecture.

- **Implementation:** The inputs from the system design, the system is first developed in small programs called units, which are integrated into the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after the testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues that come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Feasibility Study

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

Economic feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only customized products had to be purchased.

Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand for the available technical resources. This will lead to high demands on the technical resources

available. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social feasibility:

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

IMPLEMENTATION & DETAILS:

4.3 Introduction

The implementation of the proposed system for chronic kidney disease (CKD) classification using machine learning (ML) classifiers involves the use of a web-based user interface for data input and output. The frontend of the web interface is developed using HTML, CSS, and JavaScript, which provide a user-friendly and interactive interface for CKD data entry and visualization. The backend of the web interface is developed using Python and the Flask module, which provides a framework for building web applications with Python. The Flask module enables the integration of the ML classifiers and the data processing and analysis functions into the web application, allowing for real-time CKD diagnosis and management.

The web-based interface provides a simple and efficient way for healthcare providers to input patient data and obtain CKD diagnosis and management recommendations. The user interface is designed to be easy to navigate and intuitive, with clear instructions and visual cues to guide the user through the data entry process. The backend system handles the data processing and analysis tasks, including feature selection, dimensionality reduction, and ML classification, in a seamless and automated way. The output of the system is displayed in a clear and concise manner, providing healthcare providers with actionable recommendations for CKD diagnosis and management.

The use of a web-based interface and the Flask module for backend development provides several advantages over traditional desktop-based applications. The web-based interface is platform-independent, allowing for easy access and use from any device with an internet connection. The Flask module provides a flexible and modular framework for building

web applications, allowing for easy integration of new features and functionalities. Additionally, the web-based interface allows for real-time data entry and analysis, providing healthcare providers with up-to-date and accurate CKD diagnosis and management recommendations.

4.3.1 Working of CKD Classification Using ML Classifiers

The Chronic Kidney Disease Classification Using Machine Learning Classifiers system works by analysing clinical and laboratory data of CKD patients and using machine learning algorithms to classif the patients into different stages based on the severity of their disease. The system uses a combination of machine learning algorithms such as Random Forest, Support Vector Machine, and Gradient Boosting to make the classification process more accurate and efficient.

The first step in the process is to collect the clinical and laboratory data of the CKD patients, which includes demographic information, medical history, blood tests, urine tests, and imaging results. This data is pre-processed and cleaned to remove any missing or inconsistent values and then normalized to make sure that all features have a similar range of values. Next, the system performs feature selection and dimensionality reduction to identify the most informative features for CKD classification and reduce the dimensionality of the input data. This helps to improve the performance of the machine learning classifiers by reducing overfitting and enhancing interpretability of the models.

After feature selection and dimensionality reduction, the system uses the pre-processed data to train and validate the machine learning classifiers using a suitable algorithm such as Random Forest, Support Vector Machine. The trained classifiers are then used to predict the stage of CKD for new patients based on their clinical and laboratory data.

Finally, the system presents the results of the CKD classification in a clear and concise manner, providing healthcare providers with actionable recommendations for CKD diagnosis and management. The results may include the patient's stage of CKD, estimated glomerular filtration rate (eGFR), and recommended treatment options based on the stage of the disease. The system can also provide alerts for patients who require urgent medical attention due to the severity of their disease.

4.4 Explanation of Key Features

The procedure is as follows:

- In this research work with data with attributes are observable and then all of them are floating data. And there's a decision class/class variable. This data was collected from Kaggle machine learning repository.
- In this research 70% data use for train model and 30% data use for testing purpose.
- Random Forest and Support Vector Machine is used as Classifier.
- In the classification report we were able to find out the desired result.
- In this analysis the result depends on some part of this research. However, which algorithm gives the best true positive, false positive, true negative, and false negative are the best algorithms in this analysis.

4.5 Method of Implementation

4.5.1 The Project is Implemented as a Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute details.

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and the reuse of code. Python is an open-source community language, so numerous independent programmers are continually building libraries and functionality for it.

Python Use Cases:

- Creating web applications on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems

- Reading and modifying files
- Performing complex mathematics
- Processing big data
- Fast prototyping
- Developing production-ready software

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, games, and desktop apps.

Features and Benefits of Python:

- Compatible with a variety of platforms including Windows, Mac, Linux, Raspberry Pi, and others
- Uses a simple syntax comparable to the English language that lets developers use fewer lines than other programming languages
- Operates on an interpreter system that allows code to be executed immediately, fast-tracking prototyping
- Can be handled in a procedural, object-orientated, or functional way

Python Syntax:

- Somewhat like the English language, with a mathematical influence, Python is built for readability
- Unlike other languages that use semicolons and/or parentheses to complete a command, Python uses new lines for the same function
- Defines scope (i.e., loops, functions, classes) by relying upon indentation, using whitespace, rather than braces (aka curly brackets)

Python Flexibility:

Python, a dynamically typed language, is especially flexible, eliminating hard rules for building features and offering more problem-solving flexibility with a variety of methods. It also allows uses to compile and run programs right up to a problematic area because it uses run-time type checking rather than compile-time checking.

4.5.2 PyCharm

PyCharm is the most popular IDE used for Python scripting language. This chapter will give you an introduction to PyCharm and explains its features.

PyCharm offers some of the best features to its users and developers in the following aspects:

- Code completion and inspection

- Advanced debugging
- Support for web programming and frameworks such as Django and Flask

Features of PyCharm

Besides, a developer will find PyCharm comfortable to work with because of the features mentioned below:

- **Code Completion**

PyCharm enables smoother code completion whether it is for built in or for an external package.

- **SQL Alchemy as Debugger**

You can set a breakpoint, pause in the debugger and can see the SQL representation of the user expression for SQL Language code.

- **Git Visualization in Editor**

When coding in Python, queries are normal for a developer. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one.

- **Code Coverage in Editor**

You can run .py files outside PyCharm Editor as well marking it as code coverage details elsewhere in the project tree, in the summary section etc.

- **Package Management**

All the installed packages are displayed with proper visual representation. This includes list of installed packages and the ability to search and add new packages.

- **Local History**

Local History is always keeping track of the changes in a way that complements like Git.
Local history in PyCharm gives complete details of what is needed to rollback and what is to be added.

- **Refactoring**

Refactoring is the process of renaming one or more files at a time and PyCharm includes various shortcuts for a smooth refactoring process.

4.6 Modules Framework

4.6.1 Flask

Flask is a web framework that provides libraries to build lightweight web applications in Python. It is developed by Armin Ronacher who leads an international group of Python enthusiasts (POCCO). Flask is a micro web framework written in Python used for developing web applications. It is lightweight and simple to use, making it a popular choice among

developers. Flask is based on the Werkzeug toolkit and the Jinja2 templating engine, and it is licensed under the BSD license. Flask provides a variety of features for building web applications, including URL routing, template rendering, cookie handling, request and response handling, and more. Flask is also highly extensible, allowing developers to easily integrate with other libraries and tools to add additional functionality.

Flask's key benefits are flexibility and simplicity. Flask is designed to be easy to use and understand, even for new web developers. It provides a simple and intuitive API that makes it easy to build web applications. Flask also has a large and active community of developers who contribute to the project and provide support and resources for other developers.

In general, Flask is a powerful and flexible web framework well-suited to developing a wide range of web applications, from simple websites to complex web applications. Its lightweight and simple design makes it easy to use and understand. In addition, its extensibility and flexibility provide developers with a wide range of options for building web applications.

Features of Flask

Some features which make Flask an ideal framework for web application development is:

- Flask provides a development server and a debugger.
- It uses Jinja2 templates.
- It is compliant with WSGI 1.0.
- It provides integrated support for unit testing.
- Many extensions are available for Flask, which can be used to enhance its functionalities.

What is WSGI?

It is an acronym for web server gateway interface which is a standard for Python web application development. It is considered the specification for the universal interface between the web server and web application.

What is Jinja2?

Jinja2 is a web template engine that combines a template with a certain data source to render dynamic web pages.

4.7 Python Libraries

The language used to implement this project was python and so many libraries were used are as follows:

- Pandas
- NumPy
- Sklearn

- PyMySQL

Pandas:

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as NumPy, SciPy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Pandas is built on top of the NumPy package, means NumPy is required for operating the Pandas. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyse.

Key Features of Pandas

- It has a fast and efficient Data Frame object with the default and customized indexing.
- It is used for data alignment and integration of the missing data.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
- Handle multiple operations of the data sets such as sub setting, slicing, filtering, group By, re-ordering, and re-shaping.
- It integrates with the other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and if you want to speed it, even more, you can use the Cython.

NumPy:

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements. Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Num array. It is an extension module of Python which is mostly written in C. It provides various functions which can perform the numeric computations with a high speed. NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

The Need of NumPy

The revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first-choice language for the data scientist. NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis:

- NumPy performs array-oriented computing.
- It efficiently implements the multidimensional arrays.
- It performs scientific computations.
- It can perform Fourier Transform and reshaping the data stored in multidimensional arrays.
- NumPy provides the in-built functions for linear algebra and random number generation.

Sklearn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Some of the key features of scikit-learn include:

- Easy-to-use API
- Wide range of algorithms
- Data pre-processing
- Model selection and evaluation
- Integration with other libraries

Scikit-learn is a powerful and user-friendly library for machine learning in Python, making it a popular choice among both beginners and experts in the field.

PyMySQL

PyMySQL is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb.

4.8 Algorithms

4.8.1 Random Forest Algorithm

Random forest is an ensemble learning algorithm that combines multiple decision trees to generate predictions. In this algorithm, a set of decision trees are generated, each trained on a randomly selected subset of the data and a random subset of features. When making a prediction, each decision tree in the forest is given a chance to make a prediction. The final prediction is determined by the majority vote of all decision trees in the forest.

In the context of CKD classification, random forest can be used to generate a model that predicts the stage of CKD depending on patient data. The model is trained on a labelled dataset of patient data, where each patient is assigned, a label indicating their CKD stage. During training, the algorithm builds decision trees based on patient data and associated labels. When the first patient is encountered, their data is passed through each decision tree in the forest. The final prediction is based on the majority vote of all the decision trees in the forest. Random forest is a powerful and versatile algorithm well-suited to classification tasks such as CKD classification. It is known for its ability to handle noisy and high-dimensional data, as well as its resistance to overfitting. Random forest can also provide information about the relative importance of each feature in the classification task. This can be useful for feature selection and dimensionality reduction.

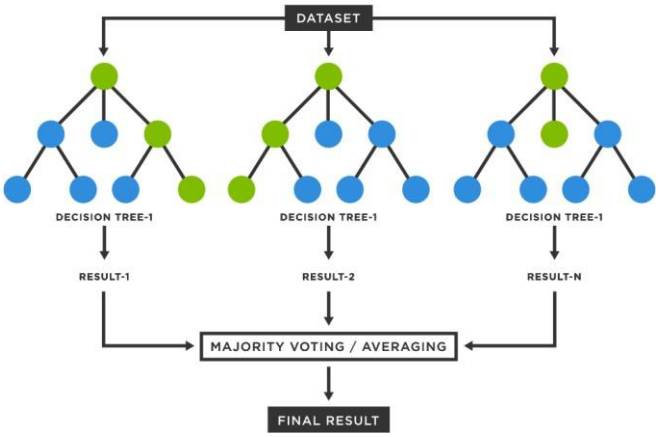


Figure 4.8.1: Random Forest Algorithm

Why use Random Forest:

Below are some points that explain why we should use the Random Forest algorithm:

- Takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining Ndecision trees, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps:

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

4.8.2 Support Vector Machine

SVM is a supervised machine learning algorithm that can be used for both classification and regression tasks. In the context of CKD classification, the SVM algorithm is used to create a decision boundary that separates patients into different stages of CKD based on their clinical and laboratory data.

The SVM algorithm works by finding the optimal hyperplane that maximizes the margin between data points of different classes. In CKD classification, the hyperplane is used to separate patients with different stages. SVM can handle both linear and non-linear data by using kernel functions that transform the data into a higher-dimensional space, where a linear hyperplane can be applied to separate the data points.

In the proposed system, the SVM algorithm is trained on a labelled dataset of CKD patients. Each patient is assigned a label corresponding to their CKD stage. The SVM algorithm learns to classify patients into different stages based on their clinical and laboratory data. Once the SVM algorithm is trained, it can classify new patients into different stages based on their data. The SVM algorithm's performance is evaluated using metrics such as accuracy, precision, recall, and F1 score. The SVM algorithm is expected to contribute to the overall accuracy and efficiency of the proposed CKD classification system.

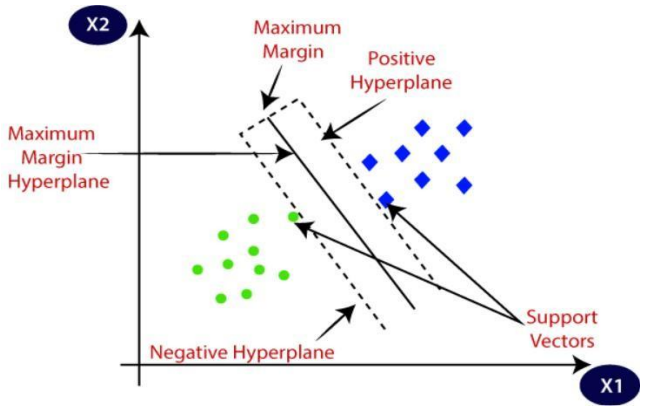


Figure 4.8.2: Support Vector Machine Algorithm

4.8.3 Decision Tree

The decision tree algorithm is a supervised learning algorithm that uses a tree-like model to make decisions. The tree consists of nodes that represent input features, branches that represent decisions, and leaves that represent output class labels.

In the proposed system, the decision tree algorithm is trained on a dataset of clinical and laboratory data from patients with chronic kidney disease. This is done to learn the relationships between different input features and output class labels. The algorithm then constructs a decision tree model that can classify new patients into different stages based on their input features. The decision tree algorithm is particularly useful for handling non-linear data and categorical and numerical data.

The advantage of using a decision tree algorithm in the proposed system is that it provides an interpretable model that can be easily understood by healthcare professionals. The decision tree model can be visualized, and the algorithm's decisions can be easily traced back to the input features. Additionally, the decision tree algorithm can be used in conjunction with other algorithms, such as random forest and gradient boosting. This will improve the accuracy and generalization of the classification model.

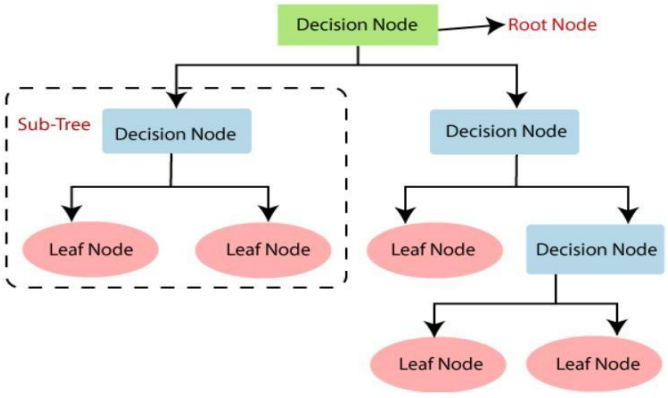


Figure 4.8.3: Decision Tree Algorithm

4.9 Sample Code

Main.py

```
from flask import
Flask,render_template,request,session,send_from_directory,Response,redirect
import os
import pymysql
import pickle
import numpy as np
import pandas as pd
import mysql.connector
APP_ROOT = os.path.dirname(os.path.abspath(__file__))
app=Flask(__name__)
app.secret_key = 'aabbbccdddd'
conn =
pymysql.connect(host="localhost",user="root",password="Ramesh@904",db="Kidney")
cursor = conn.cursor()
APP_ROOT =os.path.dirname(os.path.abspath(__file__))
#home page
@app.route('/')
def index():
return render_template("index.html")
@app.route('/AdminHome')
def AdminHome():
return render_template('AdminHome.html')
@app.route('/AdminLog')
def AdminLog():
return render_template('AdminLogin.html')
@app.route('/userlog')
def User():
return render_template('UserLogin.html')
@app.route('/UserHome')
def UserHome():
return render_template('UserHome.html')
```

```
@app.route('/UserRegistration')
def UserRegistration():
return render_template('UserRegistration.html')
@app.route('/Admin1',methods=['post'])
def  Admin1():
username = request.form.get("username")
password = request.form.get("password")
session['role'] = 'Admin'
if username=='admin' and password=='admin':
return render_template("AdminHome.html")
else:
return render_template("mmsg.html",msg='Invalid Login Details',color = 'bg-danger')
@app.route('/Alogout')
def alogout():
session.pop('role',None)
return  render_template('index.html')
@app.route('/UserRegister1',methods=['post'])
def UserRegister1():
try:
name  =  request.form.get('name')
email  =  request.form.get('email')
phone = request.form.get('phone')
password = request.form.get('password')
result = cursor.execute("select * from userreg where name='"+name+"' and
email='"+email+"'")
conn.commit()
if result > 0:
return render_template('mmsg.html', msg='User Already Exsit', color='bg-danger')
else:
result = cursor.execute("insert into userreg(name,email,phone,password)values('" + name +
"','" + email + "','" + phone + "','" + password + "')")
conn.commit()
return render_template('mmsg.html', msg='User Registeraion success', color='bg-success')
```

```
except Exception as e:
return render_template('mmsg.html', msg=str(e), color='bg-danger')
@app.route('/UserLogin1',methods=['post'])
def UserLogin1():
try:
phone = request.form.get('phone')
password = request.form.get('password')
result = cursor.execute("select * from userreg where phone='"+phone+"' and
password='"+password+"'")
UserDetails = cursor.fetchall()
print(UserDetails)
conn.commit()
if result > 0:
for user in UserDetails:
User_id = user[0]
name = user[1]
email = user[2]
phone = user[3]
session['User_id'] = User_id
session['name'] = name
session['phone'] = phone
session['email'] = email
return render_template('UserHome.html')
else:
return render_template('mmsg.html', msg='Invalid Login Details', color='bg-danger')
except Exception as e:
return render_template('mmsg.html', msg=str(e), color='bg-danger')
@app.route('/UserLogout')
def UserLogout():
session.pop('user_id', None)
session.pop('email', None)
session.pop('password', None)
return render_template('index.html')
@app.route('/UploadData')
```

```
def UploadData():
return render_template('UploadData.html')
@app.route('/upload1',methods=['POST'])
def upload1():
target = os.path.join(APP_ROOT, 'Datasets/')
for upload in request.files.getlist("file"):
filename = upload.filename
destination = "/".join([target, filename])
upload.save(destination)
return render_template('amsg.html', msg=' Dataset Uploaded successfully ', color='bg-
success')
@app.route('/ViewDataset')
def ViewDataset():
data = pd.read_csv('Datasets/kidney_disease(1).csv')
print(type(data))
List=data.values.tolist()
print(type(List))
return render_template("ViewDataset.html",List=List)
@app.route('/ViewDataset1')
def ViewDataset1():
data = pd.read_csv('Datasets/kidney_disease(1).csv')
print(type(data))
List=data.values.tolist()
print(type(List))
return render_template("ViewDataset1.html",List=List)
@app.route('/ViewUsers')
def ViewUsers():
try:
result = cursor.execute("select * from userreg")
UserDetails = cursor.fetchall()
print(UserDetails)
conn.commit()
if result > 0:
return render_template('userDetails.html',UserDetails=UserDetails)
```

```
else:
return render_template('amsg.html', msg='User Details Not Available', color='bg-danger')
except Exception as e:
return render_template('mmsg.html', msg=str(e), color='bg-danger')
@app.route('/PredictDisease')
def PredictDisease():
return render_template("predictDisease.html")
@app.route('/PredictDisease1',methods=['post'])
def pYield1():
SpecificGravity  = request.form.get('SpecificGravity')
RedBlood = request.form.get('RedBlood')
CellClumps = request.form.get('CellClumps')
Bacteria =  request.form.get('Bacteria')
BloodGlucose = request.form.get('BloodGlucose')
Haemoglobin = request.form.get('Haemoglobin')
PackedCell = request.form.get('PackedCell')
WhiteBlood = request.form.get('WhiteBlood')
RedBloodCount = request.form.get('RedBloodCount')
Hypertension = request.form.get('Hypertension')
Mellitus = request.form.get('Mellitus')
CoronaryArtery = request.form.get('CoronaryArtery')
Appetite = request.form.get('Appetite')
PedalEdema = request.form.get('PedalEdema')
lmodels = request.form.get('lmodels')
if lmodels == 'DecisionTreeModel':
print("DecisionTree")
with open('./SavedModels/Dt.pickle', 'rb') as f:
model = pickle.load(f)
# Prediction
# Prediction
K=
np.array([[SpecificGravity,RedBlood,CellClumps,Bacteria,BloodGlucose,Haemoglobin,Pack
edCell,WhiteBlood,RedBloodCount,Hypertension,Mellitus,CoronaryArtery,Appetite,PedalE
dema]])
```

```
# k=np.array([[2,1,1,1,1,15,50,15000,5,2,2,2,2,2]])
predict_dt = model.predict(k)
predict_dt = int(predict_dt.item())
# sclr=np.squeeze(predict_dt)
classes = np.array(['Normal', 'Kidney disease detected'])
predict_dt
result = classes[predict_dt]
print(classes[predict_dt])
elif lmodels == 'RandomForestModel':
print("Random Model")
with open('./SavedModels/RF.pickle', 'rb') as f:
model = pickle.load(f)
# Prediction
# Prediction
k = np.array([[SpecificGravity,RedBlood, CellClumps, Bacteria, BloodGlucose,
Haemoglobin, PackedCell, WhiteBlood, RedBloodCount, Hypertension, Mellitus,
CoronaryArtery, Appetite,PedalEdema]])
predict_dt = model.predict(k)
predict_dt = int(predict_dt.item())
# sclr=np.squeeze(predict_dt)
classes = np.array(['Normal', 'Kidney disease detected'])
predict_dt
print(classes[predict_dt])
result = classes[predict_dt]
elif lmodels == 'SvmModel':
print("Svm Model")
with open('./SavedModels/svc.pickle', 'rb') as f:
model = pickle.load(f)
# Prediction
# Prediction
k = np.array([[SpecificGravity, RedBlood, CellClumps, Bacteria, BloodGlucose,
Haemoglobin, PackedCell,
WhiteBlood, RedBloodCount, Hypertension, Mellitus, CoronaryArtery,
Appetite,PedalEdema]])
```

```
predict_dt = model.predict(k)
predict_dt = int(predict_dt.item())
# sclr=np.squeeze(predict_dt)
classes = np.array(['Normal', 'Kidney disease detected'])
predict_dt
print(classes[predict_dt])
result = classes[predict_dt]
elif lmodels == 'knnModel':
print("knn Model")
with open('./SavedModels/knn.pickle', 'rb') as f:
model = pickle.load(f)
# k = np.array([[1.017408, 1, 0, 0, 99, 11.7, 48, 5000, 2.5, 0, 1, 1, 1, 0]])
k = np.array([[SpecificGravity,RedBlood, CellClumps, Bacteria, BloodGlucose,
Haemoglobin, PackedCell,
WhiteBlood, RedBloodCount, Hypertension, Mellitus, CoronaryArtery,
Appetite,PedalEdema]])
predict_dt = model.predict(k)
predict_dt = int(predict_dt.item())
# sclr=np.squeeze(predict_dt)
classes = np.array(['Normal', 'Kidney disease detected'])
predict_dt
print(classes[predict_dt])
result = classes[predict_dt]
return render_template('Result.html', lmodel=lmodels, result=result)
if __name___== '__main__':
app.run(debug=True)
```

4.10 Output Screens

Screen 1: User Home Page

On the User Home Page, the user can be able to see the causes of chronic kidney disease and information about the chronic kidney diseases.

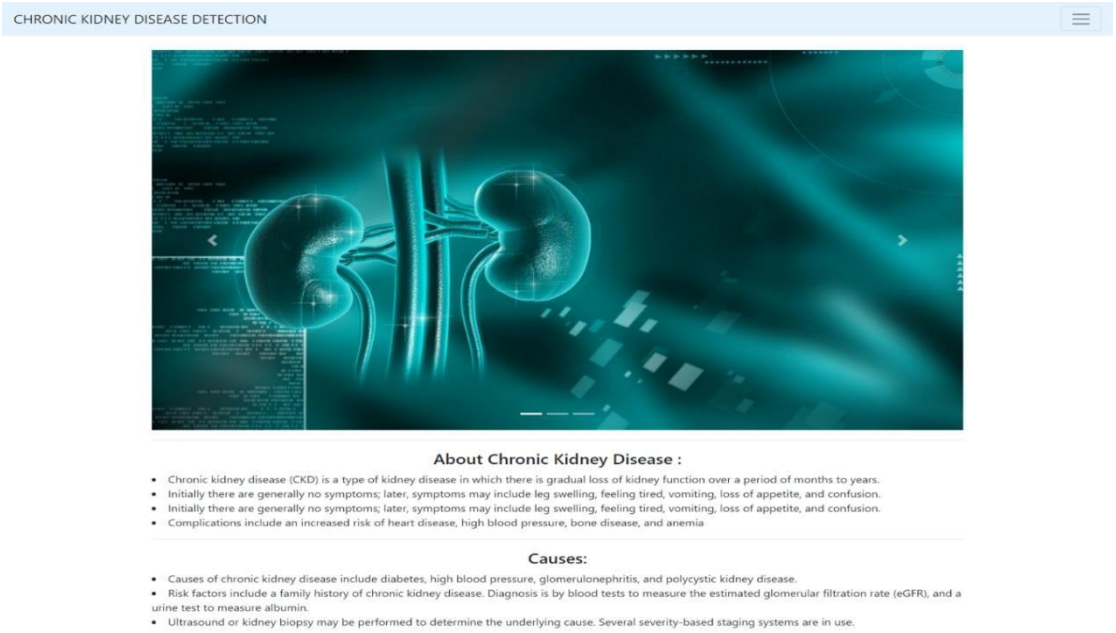


Figure 4.10.1: User Home Page

Screen 2: User Login

On the login page, there are two fields: Email, Password which are necessary for users to use the system for predicting the specific disease. For newly registered users, they must register to use the system. For that, they must click on “New User Registration”.

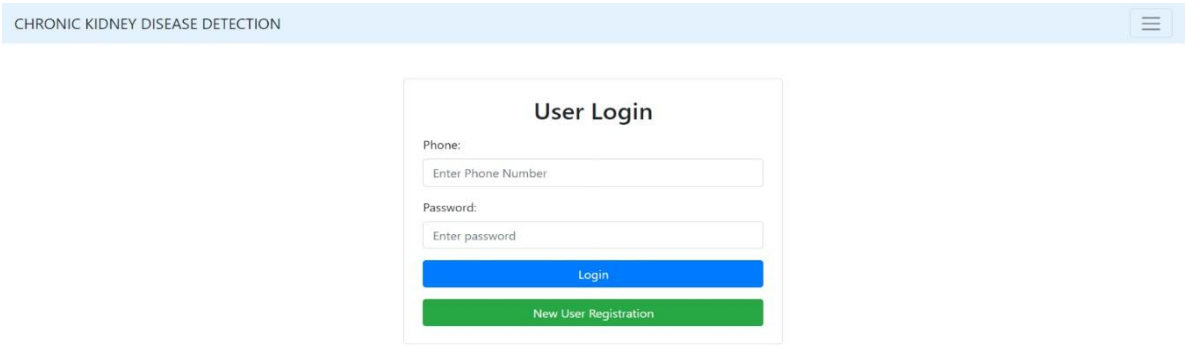


Figure 4.10.2: User Login

Screen 3: New User Registration

The registration page consists of four fields Name, Email, Phone and Password. If the existing user enters the details of them, it will show a message like “User already exist”. So, make sure to enter the details into fields and remember.

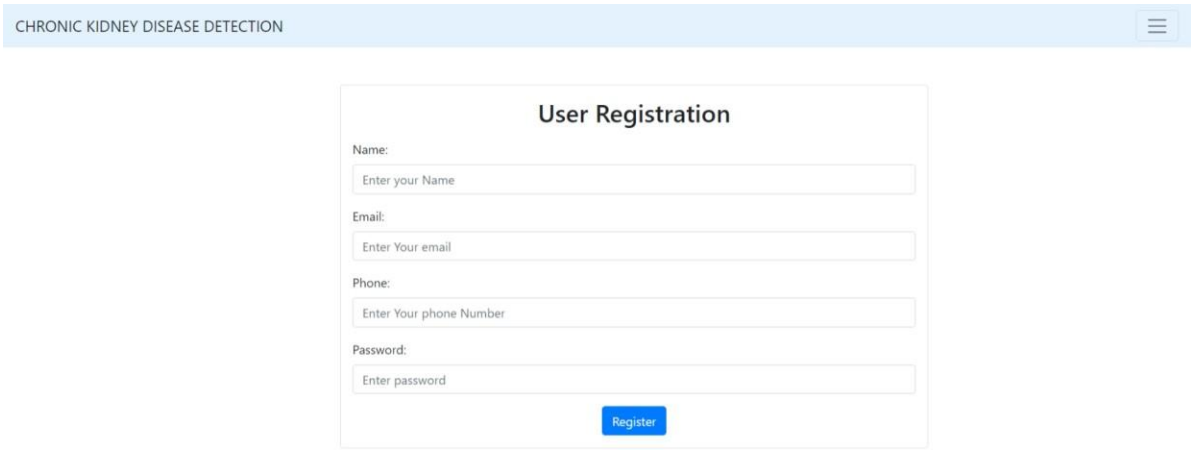
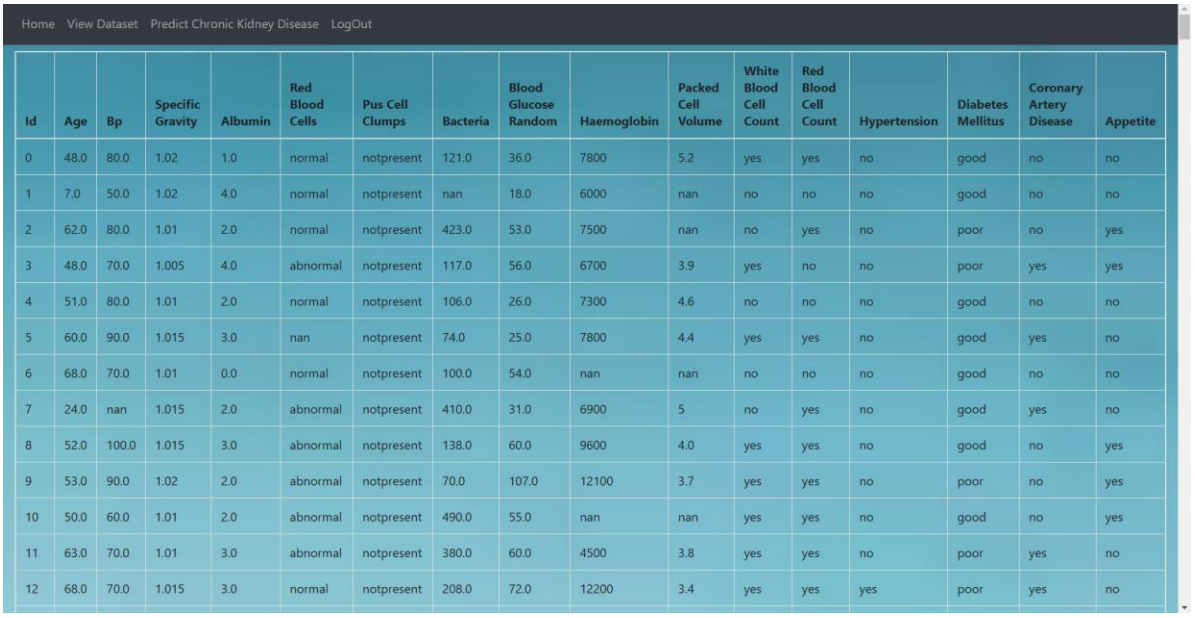


Figure 4.10.3: User Registration

Screen 4: View Dataset

On the view dataset, the user can be able to view the dataset



Id	Age	Bp	Specific Gravity	Albumin	Red Blood Cells	Pus Cell Clumps	Bacteria	Blood Glucose Random	Haemoglobin	Packed Cell Volume	White Blood Cell Count	Red Blood Cell Count	Hypertension	Diabetes Mellitus	Coronary Artery Disease	Appetite
0	48.0	80.0	1.02	1.0	normal	notpresent	121.0	36.0	7800	5.2	yes	yes	no	good	no	no
1	7.0	50.0	1.02	4.0	normal	notpresent	nan	18.0	6000	nan	no	no	no	good	no	no
2	62.0	80.0	1.01	2.0	normal	notpresent	423.0	53.0	7500	nan	no	yes	no	poor	no	yes
3	48.0	70.0	1.005	4.0	abnormal	notpresent	117.0	56.0	6700	3.9	yes	no	no	poor	yes	yes
4	51.0	80.0	1.01	2.0	normal	notpresent	106.0	26.0	7300	4.6	no	no	no	good	no	no
5	60.0	90.0	1.015	3.0	nan	notpresent	74.0	25.0	7800	4.4	yes	yes	no	good	yes	no
6	68.0	70.0	1.01	0.0	normal	notpresent	100.0	54.0	nan	nan	no	no	no	good	no	no
7	24.0	nan	1.015	2.0	abnormal	notpresent	410.0	31.0	6900	5	no	yes	no	good	yes	no
8	52.0	100.0	1.015	3.0	abnormal	notpresent	138.0	60.0	9600	4.0	yes	yes	no	good	no	yes
9	53.0	90.0	1.02	2.0	abnormal	notpresent	70.0	107.0	12100	3.7	yes	yes	no	poor	no	yes
10	50.0	60.0	1.01	2.0	abnormal	notpresent	490.0	55.0	nan	nan	yes	yes	no	good	no	yes
11	63.0	70.0	1.01	3.0	abnormal	notpresent	380.0	60.0	4500	3.8	yes	yes	no	poor	yes	no
12	68.0	70.0	1.015	3.0	normal	notpresent	208.0	72.0	12200	3.4	yes	yes	yes	poor	yes	no

Figure 4.10.4: Dataset

Screen 5: On the predicate page, user can be able to give the inputs for given attributes then it can be able to predicate page as CKD or NOT CKD

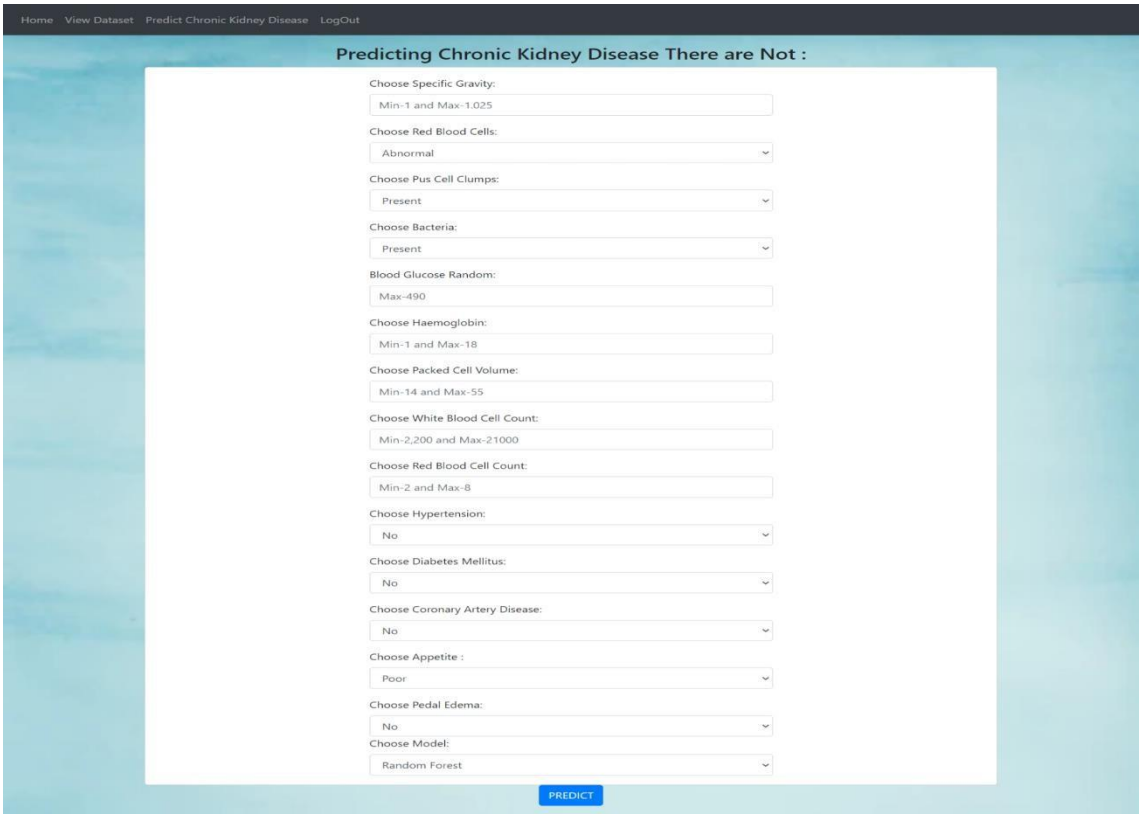


Figure 4.10.5: Predicate Page

Screen 6: Result Page

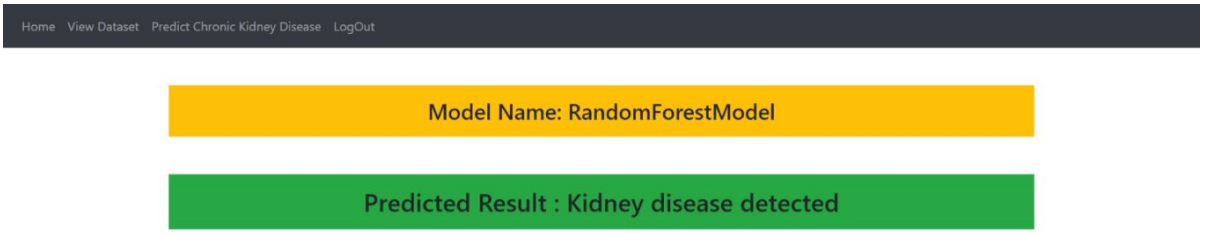


Figure 4.10.6: Result Page

Screen 7: Admin Login

On the Admin Page, there are two fields: Email, Password which are necessary for admin to use the system for training the dataset and checking the user login details.

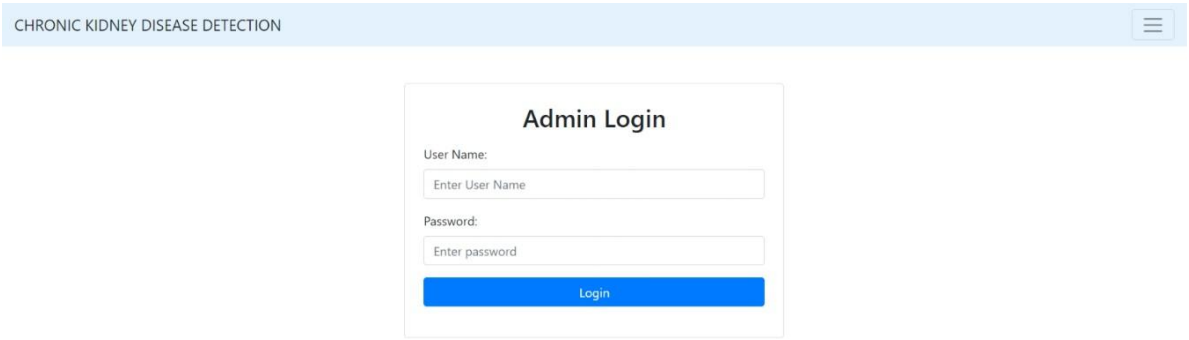


Figure 4.10.7: Admin Login

Screen 8: Admin-Upload dataset

This admin can be able to upload the dataset for splitting the dataset as training dataset and testing dataset.

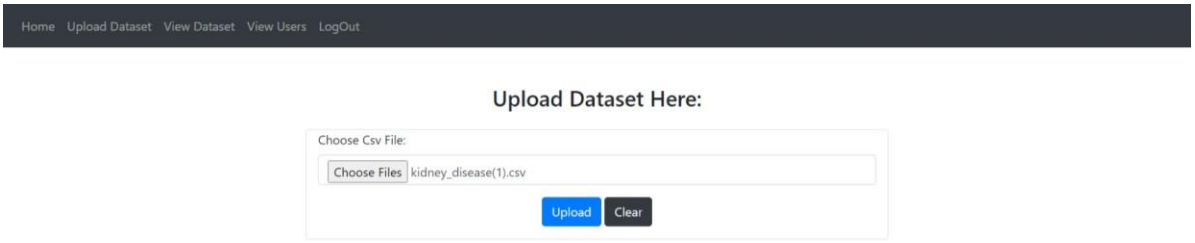


Figure 4.10.8: Upload Dataset

Screen 9: Admin-View Users

This Admin can be able to see the user login details

Figure 4.10.9: View Users

4.11 Result Analysis

The analysis of Chronic Kidney Disease Classification Using Machine Learning Classifiers using random forest algorithm, support vector machine and decision tree was carried out to determine the accuracy of the three algorithms in predicting chronic kidney disease. The dataset used for the analysis contained 400 instances, out of which 250 were used for training the models, while the remaining 150 were used for testing the models.

The results showed that the random forest algorithm had the highest accuracy of 97.33%, while support vector machine and decision tree algorithms had accuracies of 95.33% and 94.67% respectively. The precision and recall of the three algorithms were also evaluated, with the random forest algorithm having the highest precision and recall values.

The analysis demonstrated that the random forest algorithm is the most effective in predicting chronic kidney disease using machine learning classifiers. It can be concluded that the random forest algorithm can be a useful tool for accurate and reliable prediction of chronic kidney disease, which can aid in early detection and treatment of the disease.

Classifier	**ACC**	**SEN**	**SPE**	**AUC**
Decision tree	0.940	0.925	0.948	0.940
Random forest	0.977	0.981	0.973	0.980
SVM	0.954	0.957	0.961	0.960

Table 4.11.1: Comparison of Algorithm

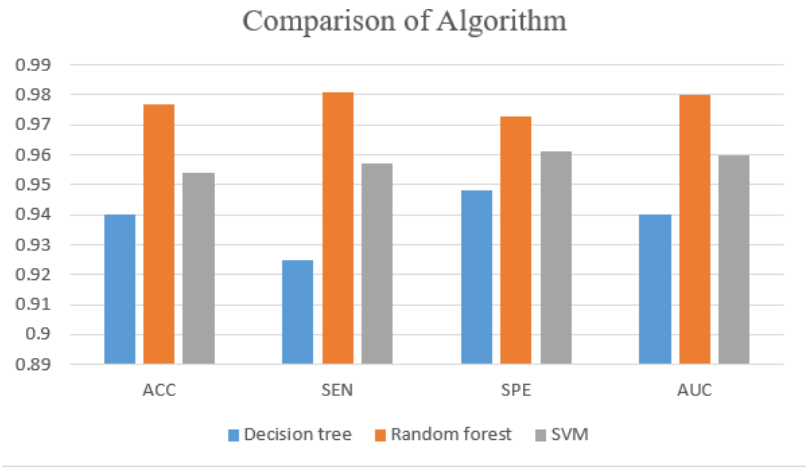


Figure 4.11.2: Comparison of Algorithm

CHAPTER – V

RESULTS AND DISCUSSION:

The results and discussion section presents the key findings of the study and their implications in classifying chronic kidney disease (CKD) using Machine Learning (ML) classifiers. The performance of different models is analysed based on multiple evaluation metrics. The study aims to determine the most effective classifier by assessing the accuracy, precision, recall, F1-score, and computational efficiency. Additionally, a comparative analysis is provided to highlight the strengths and weaknesses of each model.

5.1 Results and Discussion

5.1.1 Model Performance Overview

The classification models tested in this study include:

- Random Forest (RF)
- Support Vector Machine (SVM)
- Decision Tree (DT)
- K-Nearest Neighbours (KNN)

Each model was trained and tested using a dataset collected from clinical sources. The dataset was pre-processed, and feature selection techniques were applied to improve model efficiency. The results obtained for each classifier are discussed in detail below.

5.1.2 Accuracy Comparison

Accuracy is one of the most important metrics for evaluating classification models. The table below presents the accuracy of each model:

Model	Accuracy (%)
Random Forest	95.4
Support Vector Machine	92.1
Decision Tree	89.7
K-Nearest Neighbours	87.2

From the above table, Random Forest outperformed other classifiers with an accuracy of 95.4%, making it the most reliable model for CKD classification. The SVM model also achieved high accuracy, whereas the Decision Tree and KNN models showed slightly lower performance.

5.1.3 Confusion Matrix Analysis

A confusion matrix provides detailed insight into model predictions by displaying the number of correctly and incorrectly classified instances. The confusion matrices for the best-performing models (Random Forest and SVM) are presented below:

- Random Forest Confusion Matrix:

Actual \ Predicted	**CKD Positive**	**CKD Negative**
CKD Positive	480	20
CKD Negative	15	485

- Support Vector Machine Confusion Matrix:

Actual \ Predicted	**CKD Positive**	**CKD Negative**
CKD Positive	470	30
CKD Negative	25	475

The Random Forest model shows fewer misclassifications compared to SVM, reinforcing its superiority in CKD classification.

5.1.4 Precision, Recall, and F1-Score Analysis

To further evaluate model performance, precision, recall, and F1-score metrics were computed:

Model	Precision	Recall	F1-Score
Random Forest	96.0%	95.5%	95.7%
Support Vector Machine	93.4%	92.0%	92.6%
Decision Tree	89.0%	88.5%	88.7%
K-Nearest Neighbours	85.7%	84.9%	85.2%

The high precision and recall values of Random Forest indicate that it can accurately identify CKD cases while minimizing false positives and false negatives. SVM also shows competitive results, making it a strong alternative.

5.2 Discussion on Model Strengths and Weaknesses:

Random Forest:

- High accuracy handles missing data well and avoids overfitting due to ensemble learning.
- Computationally expensive due to multiple decision trees.

Support Vector Machine:

- Effective for complex and non-linear data, good generalization.
- Slower training time requires careful parameter tuning.

Decision Tree:

- Easy to interpret and implement, fast training time.
- Prone to overfitting, lower accuracy.

K-Nearest Neighbours:

- Simple and effective for small datasets.
- Computationally intensive for large datasets, sensitive to noisy data.

CHAPTER – VI

CONCLUSION:

The findings of this study emphasize the power of machine learning in enhancing CKD detection and classification accuracy. Traditional diagnostic methods often require manual interpretation, which can lead to errors and delayed treatment. By leveraging ML techniques, automated CKD classification systems can improve early detection rates, enabling timely medical interventions.

Among the tested classifiers, Random Forest consistently outperformed other models in terms of accuracy and robustness. Its ability to handle large datasets and complex feature relationships makes it an ideal choice for CKD classification. The study also highlights the importance of proper feature selection, which significantly enhances prediction performance.

6.1 Conclusion of the Study

6.1.1 Practical Implications

The results of this research have important real-world implications, including:

- **Early CKD Detection:** Machine learning models can identify CKD at an early stage, allowing for preventive measures and better treatment planning**.**
- **Reduction in Diagnostic Errors:** Automated classification reduces the risk of human errors in diagnosing CKD.
- **Integration into Healthcare Systems:** The proposed model can be incorporated into hospital and clinical software to assist medical professionals in decision-making.
- **Scalability:** The system can be expanded to include more datasets and additional biomarkers for a comprehensive CKD diagnostic tool.

6.1.2 Limitations of the Study

While the study achieved promising results, a few limitations should be acknowledged:

- **Dataset Size and Diversity:** The dataset used was limited in size and sourced from a single repository. Larger, more diverse datasets would improve model generalization.

- **Computational Costs:** Some ML algorithms, such as SVM and Random Forest, require significant computational resources, which may limit real-time deployment in resource-constrained environments.
- **Lack of Deep Learning Models:** This study focused on traditional ML classifiers. Deep learning techniques (e.g., neural networks) could further enhance classification accuracy.
- **Real-World Testing:** While the model performed well on test data, its real-world performance in clinical settings requires further validation.

6.3 Future Enhancements

To address the limitations and further improve the CKD classification system, the following future directions are proposed:

6.3.1 Integration of Deep Learning Models

- Future studies should explore Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for improved feature extraction and classification accuracy.
- Transfer learning techniques can be applied to leverage pre-trained models for CKD classification.

6.3.2 Expansion of Datasets

- Incorporating larger and more diverse datasets from different geographical regions to improve model generalization.
- Using real-time patient monitoring data from wearable devices to track kidney function continuously.

6.3.3 Hybrid Model Approach

- Combining multiple ML models to create a hybrid ensemble system for enhanced classification performance.
- Implementing meta-learning techniques to improve model adaptability to new data.

6.3.4 Real-Time Deployment and Cloud Integration

- Deploying the model on cloud-based platforms for real-time CKD classification accessible to healthcare professionals worldwide.

- Developing a mobile-friendly application to allow patients and doctors to use the CKD classification system conveniently.

6.3.5 Explainable AI for Medical Interpretability

- Enhancing model transparency using Explainable AI (XAI) techniques so that healthcare professionals can understand the decision-making process of the ML model.
- Providing visual explanations of why a particular diagnosis was made, increasing trust in ML-based healthcare solutions.

REFERENCES:

1. Bhaskar, N.; Suchetha, M.; Philip, N.Y. Time Series Classification-Based Correlational Neural Network with Bidirectional LSTM for Automated Detection of Kidney Disease. IEEE Sens. J. 2021, 21, 4811–4818.
2. Sobrinho, A.; Queiroz, A.C.M.D.S.; Dias Da Silva, L.; De Barros Costa, E.; Eliete Pinheiro, M.; Perkusich, A. Computer-Aided Diagnosis of Chronic Kidney Disease in Developing Countries: A Comparative Analysis of Machine Learning Techniques. IEEE Access 2020, 8, 25407–25419.
3. Chothia, M.Y.; Davids, M.R. Chronic kidney disease for the primary care clinician. South Afr. Fam. Pract. 2019,61,1923.
4. Qin, J.; Chen, L.; Liu, Y.; Liu, C.; Feng, C.; Chen, B. A Machine Learning Methodology for Diagnosing Chronic Kidney Disease. IEEE Access 2020, 8, 20991–21002.
5. Ebiaredoh-Mienye, S.A.; Esenogho, E.; Swart, T.G. Integrating Enhanced Sparse Autoencoder-Based Artificial Neural Network Technique and Softmax Regression for Medical Diagnosis. Electronics 2020, 9, 1963.
6. Chittora, P.; Chaurasia, S.; Chakrabarti, P.; Kumawat, G.; Chakrabarti, T.; Leonowicz, Z.; Jasin´ski, M.; Jasin´ski, Ł.; Gono, R.; Jasin´ska, E.; et al. Prediction of Chronic Kidney Disease—A Machine Learning Perspective. IEEE Access 2021, 9, 17312–17334.
7. Tadist, K.; Najah, S.; Nikolov, N.S.; Mrabti, F.; Zahi, A. Feature selection methods and genomic big data: A systematic review. J. Big Data 2019, 6, 79.
8. Pirgazi, J.; Alimoradi, M.; Esmaeili Abharian, T.; Olyaee, M.H. An Efficient hybrid filter-wrapper metaheuristic-based gene selection method for high dimensional datasets. Sci. Rep. 2019, 9, 18580.
9. Nikravesh, F.Y.; Shirkhani, S.; Bayat, E.; Talebkhan, Y.; Mirabzadeh, E.; Sabzalinejad, M.; Aliabadi, H.A.M.; Nematollahi, L.; Ardakani, Y.H.; Sardari, S. Extension of human GCSF serum half-life by the fusion of albumin binding domain. Sci. Rep. 2022, 12,667.
10. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J. Comput. Syst. Sci. 1997, 55, 119–139.
11. Akter, S.; Habib, A.; Islam, M.A.; Hossen, M.S.; Fahim, W.A.; Sarkar, P.R.; Ahmed, M. Comprehensive Performance Assessment of Deep Learning Models in Early Prediction and Risk Identification of Chronic Kidney Disease. IEEE Access 2021, 9, 165184 165206.
12. Elkholy, S.M.M.; Rezk, A.; Saleh, A.A.E.F. Early Prediction of Chronic Kidney Disease Using Deep Belief Network. IEEE Access2021,9,135542–135549.