A Project Report On

CHRONIC KIDNEY DISEASE PREDICTION AND DIET RECOMMENDATION USING MACHINE LEARNING TECHNIQUES

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

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Submitted By

A. Y R Nikshiptha	**(21MH1A4203)**
K. Sri Lalitha	**(21MH1A4229)**
N. Samuel Satya Paul	**(21MH1A4237)**
G. Akash Kumar	**(21MH1A4213)**

Under the esteemed supervision of

Mr. K.N.S.K. Santhosh, M.Tech. (Ph.D.)

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY(A)

Permanently Affiliated to JNTUK, Kakinada * Approved by AICTE New Delhi Accredited by NBA,

Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh.

2021-2025

VISION & MISSION OF THE INSTITUTE

VISION

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Value based instruction and to emerge as a premiere institute.

MISSION

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative Research and development
- Industry Institute Interaction
- Empowered Manpower

PRINCIPAL

VISION & MISSION OF THE DEPARTMENT

VISION

To be a recognized Computer Science and Engineering hub striving to meet the growing needs of the Industry and Society.

MISSION

M1: Imparting Quality Education through state-of-the-art infrastructure with industry Collaboration

M2: Enhance Teaching Learning Process to disseminate knowledge.

M3: Organize Skill based, Industrial and Societal Events for overall Development.

HOD

DEPARTMENT OF CSE - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

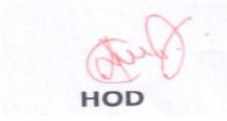
PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1:	**Proficiency in Al&ML. Applications** Expertise in designing, developing and deploying Al and ML solutions for health care, finance, and autonomous system problems
PEO2:	**Lifelong Learning and Adaptability** Adapt to evolving technologies and industry trends in Al and M. through continuous learning and self-improvement
PEO3:	**Effective Communication and Collaboration** Possess strong communication mating them to effectively collaborate with interdisciplinary teams and solve complex problems using Al and ML concepts

PROGRAMMEOUTCOMES (POS)

After successful completion of the program, the graduates will be able to

PO1:	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2:	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3:	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4:	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5:	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6:	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7:	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8:	**PO8:** Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9:	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10:	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11:	**Project management and finance:** Demonstrate knowledge and understanding of member and leader in a team, to manage projects and in multidisciplinary environments.
PO12:	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



PROGRAMME SPECIFIC OUTCOMES (PSOS)

PSO1:	**AI and ML system development** Design and develop, and implement AI and ML systems by applying fundamental principles, algorithms and techniques.
PSO2:	**Data-Driven Decision-Making** Collect, Preprocess and analyze large and diverse datasets to extract meaningful insights using AI and ML techniques to support data-driven decision-making processes in various domains.

ADITYA COLLEGE OF ENGINEERING &TECHNOLOGY(A)

(An Autonomous Institution)

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

Accredited by NBA, Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

This is to certify that the project work entitled "**The Chronic Kidney Disease Prediction and Diet Recommendation system using Machine Learning Techniques"**, is a bonafide work carried out by **Anumalasetti. Y. R. Nikshiptha (21MH1A4203), Koti. Sri Lalitha (21MH1A4229), Nakka. Samuel Satya Paul (21MH1A4237), Gummapu. Akash Kumar (21MH1A4213)**, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Artificial Intelligence & Machine Learning** from Aditya College of Engineering & Technology during the academic year 2021- 2025.

Project Guide

K.N.S.K. Santhosh, M.Tech. (Ph.D.)
Assistant Professor

Head Of The Department

Dr. B. Kiran Kumar, M.Tech., Ph.D.
Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "**The Chronic Kidney Disease Prediction and Diet Recommendation system using Machine Learning Techniques**", has been undertaken by us and this work has been submitted to **Aditya College of Engineering & Technology** affiliated to JNTUK, Kakinada, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Artificial Intelligence & Machine Learning.**

We further declare that this project work has not been submitted in full or part for the award of any degree of this or in any other educational institutions.

PROJECT ASSOCIATES

A.Y.R.Nikshiptha	**(21MH1A4203)**
K.SriLalitha	**(21MH1A4229)**
N.Samuel Satya Paul	**(21MH1A4237)**
G.Akash Kumar	**(21MH1A4213)**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to our Project Supervisor, **Mr. K.N.S.K. Santhosh,** **M.Tech. (Ph.D.)** who has guided us a lot and encouraged us in every step of the project work, his valuable moral support and guidance throughout the project helped us to a great extent.

We wish to express our sincere thanks to the Head of the Department **Dr. B. Kiran Kumar,** **M.Tech., Ph.D.** for his valuable guidance given to us throughout the period of the project work and throughout the program.

We feel elated to thank **Dr. Pullela SVVSR Kumar,** **M.Tech., Ph.D.** Dean – Academics of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We feel elated to thank **Dr. K. Manoz Kumar Reddy,** **M.Tech., Ph.D.** Dean – Evaluation of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We feel elated to thank **Dr. A. Ramesh,** **M.Tech., Ph.D.** Principal of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program. We wish to express our sincere thanks to all **faculty members, lab programmers** for their valuable assistance throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the Management of **Aditya College Of Engineering & Technology** for providing a great support for us in completing our project and also throughout the program.

PROJECT ASSOCIATES

A.Y.R.Nikshiptha	**(21MH1A4203)**
K.SriLalitha	**(21MH1A4229)**
N.Samuel Satya Paul	**(21MH1A4237)**
G.Akash Kumar	**(21MH1A4213)**

ABSTRACT

Chronic Kidney Disease is a major health problem worldwide, but very few cases are properly diagnosed until the later stages, leading to severe complications and kidney failure. This project presents a machine-learning-based model that predicts early CKD using various medical attributes: age, blood pressure, serum creatinine, levels of hypertension, etc. Machine learning algorithms such as Extra Tree Classifier, Random Forest, Decision Tree, Support Vector Machine, AdaBoost, Gaussian Naive Bayes, Gradient Boosting, K-Nearest Neighbor, and others are used to ensure accuracy and reliability.

The proposed system helps automate the detection of CKD, giving the ability of the system to predict the CKD stages based on the Glomerular Filtration Rate in real-time. It also recommends diets that can assist the patients in managing their health better. The model is implemented in Python with the assistance of Flask so that the web deployment model is also accessible to doctors and patients.

The present system thus aims the efficiency and the accuracy of CKD diagnosis suing data science approaches with lower intervention. The project proves the effective of artificial intelligence in healthcare in aiding early diagnosis early diagnosis and personalized treatment plan for CKD patients. Future scope may include the incorporation of deep learning methods and an Android app for enhanced accessibility.

CONTEXT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The healthcare industry generates terabytes of data every year. The medical documents maintained are a pool of information regarding patients. The task of extracting useful information or quality healthcare is tricky and important. By analysing these voluminous data, we can predict the occurrence of the disease and safeguard people. Thus, an intelligent system for disease prediction plays a major role in controlling disease and maintaining the good health status for people by providing accurate and trustworthy disease risk prediction.

Machine learning is a field concerned with the study of large and numerous variable information. In Health Care discerning, Machine learning guarantees to help doctors to form perfect determination, suggests the leading medicines for the patient's, spot patients at high-risk for pitiable results and particularly progressing patient's physical condition whereas minimizing costs. Machine learning has demonstrated a victory in forecast and conclusion of different basic illness.

Chronic Kidney disease is worldwide health disease with higher burden with regard to the wellbeing within the show circumstance. Chronic Kidney infection is characterized as a glomerular filtration rate(GFR)<60mL/min or Kidney harm or both for at slightest a period of3 months. End-stage renal illness is completely connected with mortality. Chronic Kidney is recognized with research facility tests. Major downside of this disease is, most of the time CKD is recognized at its last stage and which too leads to kidney failure. Within the early stages of chronic kidney illness, there will be few signs or side effects. CKD may not ended up clear until kidney work is altogether disabled.

Chronic kidney malady can be advance to conclusion organize of kidney failure, which is fatal without dialysis or a kidney transplantation. CKD is a complicated illness by influencing the parts of the body by causing anemia, cardiovascular disease, Decreased Immune system, harm to central nervous system. It is exceptionally critical to urge check-up patients within short period of time.

1.1 OBJECTIVES (AIM OF THE PROJECT)

- System is an health care application which is an efficient tool for disease prediction.
- System is an real time application which is meant for physician and peoples.
- System is an automation for chronic kidney disease prediction.
- System makes use of "Machine learning" algorithms for CKD prediction. System also predict the stages of the CKD patients.
- System predicts CKD prediction based on the attributes such as age, sugar, serum creatinine, hypertension and some others.

1.2EXISTING SYSTEM

The health care need more support for its development and developing countries like India. Previously prediction of CKD was done by checking and testing the various attributes which is depended on the kidney disease. Result of the test will test more time and some hospitals will not give proper test report because of amount. Patients are getting high risk for their treatment in critical condition. Another method is to check whether the patient had high blood pressure, a history of cardiovascular disease, the patient's relative who have kidney disease and dialysis of kidney takes more time to analysis the working of kidney. Suppose we are available data set which contains the attributes which affects the CKD and that data is collected and build the model which helps patients that provide the CKD prediction and Diet recommendation for the CKD patients.

Limitations of Existing System

- No automation for CKD prediction.
- Lack of Proper decision
- No proper medication in emergency
- Requires more time for the test report
- Understanding the test report is difficult for peoples.

1.3PROPOSED SYSTEM

Prediction is a statement about future events. Chronic Kidney Disease Prediction has become the need of the patients and Physician. Although future events are uncertain, accurate prediction is not possible. This paper includes a decision-making support model that can be helpful for doctors to provide better medication and also for patients, it provides diet recommendation which has to be maintained. The comparision the execution of Extra Tree Classifier, Random forest Classifier, Decision tree Classifier, Support Vector Machine, Adaboost, Gaussian Naïve Bayes, Gradient Boosting, K-Nearest Neighbor (KNN) classifier on the basis of its accuracy for CKD prediction.
Stages of CKD is anticipated based on the GFR rate. Show the diet suggestion video for the CKD patient for better recovery.

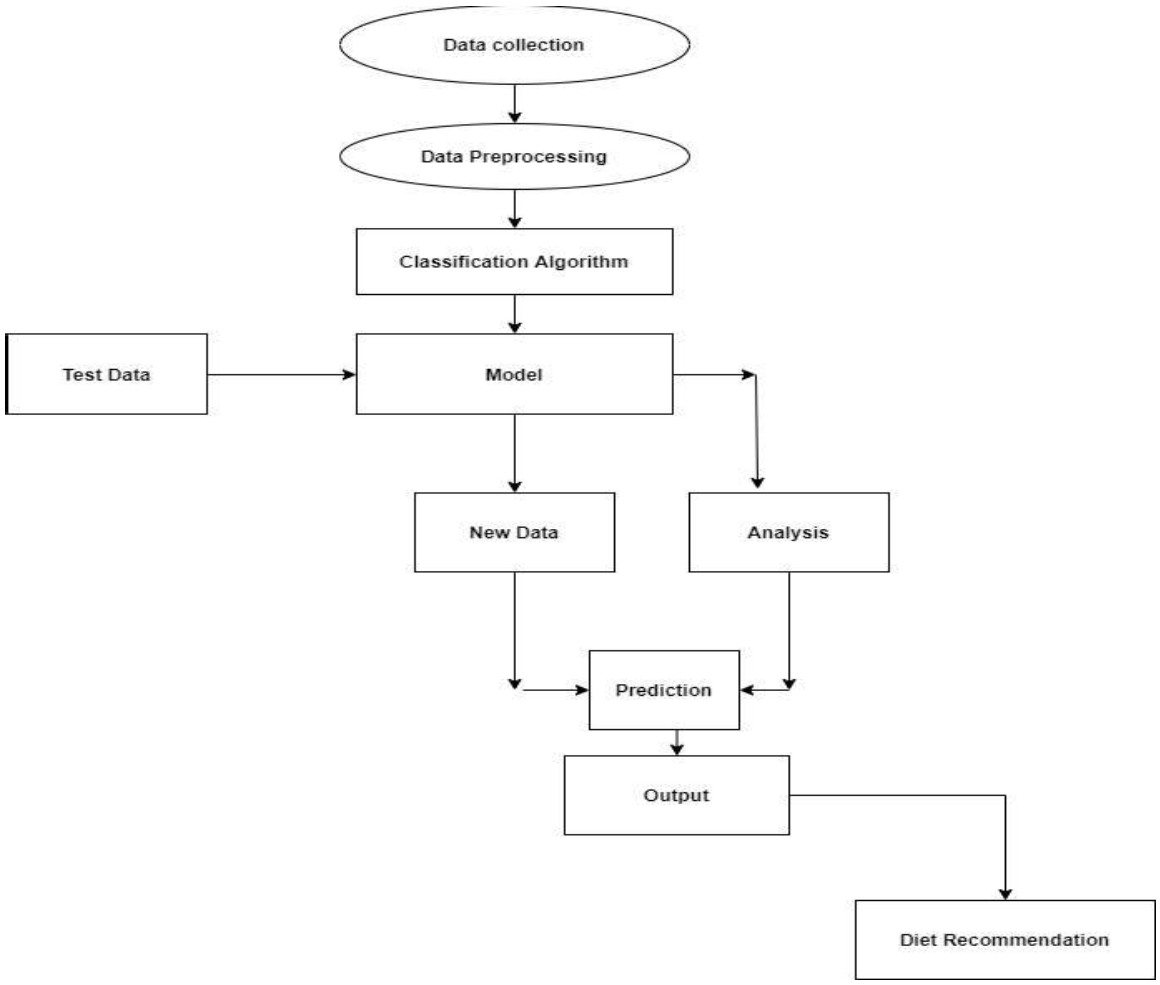


Fig 1.1 Architecture of Proposed system

As shown in fig 1.1 Architecture of Proposed system for CKD prediction for health industry, Firstly, the data set which fits the model, then model will be able to provide CKD Prediction and along with the stages which will be beneficial for patients to make better decisions on their health and diet suggests on the attributes which is affecting the CKD.

1.4 ADVANTAGES OF PROPOSED SYSTEM

- ✦ Useful to health department to predict the CKD.
- ✦ Useful for the patients to take better recovery.
- ✦ We use data science techniques for accurate results.
- ✦ On click of button output will be generated, no too much time required for CKD prediction. No need to analyze manually.
- ✦ All records stored on server (SQL Server) for easy accessing.

1.5 APPLICATIONS

- ✦ Proposed system can be used in medical department for the prediction of CKD .
- ✦ Proposed system can be used by patients to know the if the CKD is present or not by inputting data such as "age", "blood pressure", "serum creatinine", "sugar" and "bacteria" etc.
- ✦ Proposed system provide diet recommendation for the CKD patients for better recovery.

1.6 SCOPE OF THE PROJECT

- System is an health care application which is an efficient tool for disease prediction.
- System is an real time application which is meant for physician and peoples
- System is an real time application which is meant for physician and peoples.
- System is an automation for chronic kidney disease prediction.
- System provides Diet suggestion for the CKD patients based on the variations in the attributes from the dataset. System can be extended to an android application that displays the diet recommendation and doctor recommendation for the CKD patient for better improvement of the health.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Descriptions Of The Software Used

Limitations of C:

- C developers are forced to contend with manual memory management.
- Ugly pointer arithmetic.
- C is structured programming language.
- Programmers require complete knowledge of best programming technique.

Limitations of C++:

- C++ can be thought as an Object Oriented layer on top of C.
- It involves manual memory management.
- Ugly pointer arithmetic.
- Ugly syntactical constructs.

Limitations of JAVA/J2EE:

- Java programmers must use java front to back during development cycle.
- It is not appropriate for many graphical or numerical intensive applications.
- .NET provides solution to all the above-mentioned problems.

Limitations of SQL:

- SQL is most commonly used database.
- It has a lot of capabilities (ex. For loop and functions)
- Easy to maintain
- Data warehouse function for decision support, integration closely related to many other server software, good cost, performance, etc.

2.2 Literature review done in connection with the work

This section consists of the reviews of various technical and review articles on data mining techniques applied to predict Kidney Disease.

- DSVGK Kaladhar, Krishna Apparao Rayavarapu and Varahalarao Vadlapudi et al . described in their research to understand machine learning techniques to predict kidney stones. They predicted good accuracy with C4.5, Classification tree and Random forest (93%) followed by Support Vector Machines (SVM) (91.98%). Logistic and NN has also shown good accuracy results with zero relative absolute error and 100% correctly classified results. ROC and Calibration curves using Naive Bayes has also been constructed for predicting accuracy of thedata. Machine learning approaches provide better results in the treatment of kidney stones.
- J.Van Eyck, J.Ramon, F.Guiza, G.Meyfroidt, M.Bruynooghe, G.Van den Berghe, K.U.Leuvenet al. Explored data mining techniques for predicting acute kidney injury after elective cardiacsurgery with Gaussian process & machine learning techniques (classification task & regression task).
- K.R.Lakshmi, Y.Nagesh and M.VeeraKrishna et al. presented performance comparison of Artificial Neural Networks, Decision Tree and Logical Regression are used for Kidney dialysis survivability. The data mining techniques were evaluated based on the accuracy measures such as classification accuracy, sensitivity and specificity. They achieved results using 10 fold cross- validations and confusion matrix for each technique. They found ANN shows better results. Hence ANN shows the concrete results with Kidney dialysis of patient records.
- Morteza Khavanin Zadeh, Mohammad Rezapour, and Mohammad Mehdi Sepehri et al described in their research by using supervised techniques to predict the early risk of AVF failure in patients. They used classification approaches to predict probability of complication in new hemodialysis patients whom have been referred by nephrologists to AVF surgery.
- Abeer Y. Al-Hyari et al .proposed in their research by using Artificial Neural Network (NN), Decision Tree (DT) and Naïve Bayes (NB) to predict chronic kidney disease. The proposed NN algorithm as well as the other data mining algorithms demonstrated high potential in successful kidney disease.
- Xudong Song, Zhanzhi Qiu, Jianwei Mu et al .introduced data mining decision tree classification method, and proposed a new variable precision rough set decision tree classification algorithm based on weighted limit number explicit region.
- N. SRIRAAM, V. NATASHA and H. KAUR et al .presented data mining approach for parametric

evaluation to improve the treatment of kidney dialysis patient. Their experimental result shows that classification accuracy using Association mining between the ranges 50– 97.7% is obtained based on the dialysis parameter combination. Such a decision-based approach helps the clinician to decide the level of dialysis required for individual patient

- Jicksy Susan Jose, R.Sivakami, N. Uma Maheswari, R.Venkatesh et al . Their research describes an efficient Diagnosis of Kidney Images Using Association Rules. Their approach is divided into four major steps: pre-processing, feature extraction and selection, association rule generation, and generation of diagnosis suggestions from classifier.
- Divya Jain et al presented effect of diabetes on kidney using C4.5 algorithm with Tanagra tool. The performance of classifier is evaluated in terms of recall, precision and error rate.
- Koushal Kumar and Abhishek et al. their research describes comparison of all three neural networks such as (MLP, LVQ, RBF) on the basis of its accuracy, time taken to build model, and training data set size.

In 2015, Konstantina Kourou et.al [1] proposed a study of Machine learning applications in cancer prognosis and prediction. In this paper, they have presented a review of various recent ML approaches that are applied for the prediction of cancer detection. Here they have presented review of newly published content for the work done so far in cancer detection. In 2015 P.Swathi Baby et. al [2] proposed a project to diagnosis and prediction system based on predictive mining. Here kidney disease data set is used and analysed using Weka and Orange software. Here the Machine learning algorithms such as AD Trees, J48, K star, Naïve Bayes, Random forest are used for the performance study of each algorithm which gives the Statistical analysis and predicting kidney diseases using the algorithms. Their observation shows that the best algorithms K-Star and Random Forest for the used Dataset, where Build the models are less time(0 sec and 0.6 sec) and the ROC values are 1. In 2015, Konstantina Kourou et.al [1] proposed a study of Machine learning applications in cancer prognosis and prediction. In this paper, they have presented a review of various recent ML approaches that are applied for the prediction of cancer detection. Here they have presented review of newly published content for the work done so far in cancer detection. In 2015 P.Swathi Baby et. al [2] proposed a project to diagnosis and prediction system based on predictive mining. Here kidney disease data set is used and analysed using Weka and Orange software.

Here the Machine learning algorithms such as AD Trees, J48, K star, Naïve Bayes, Random forest are used for the performance study of each algorithm which gives the Statistical analysis and predicting kidney diseases using the algorithms. Their observation shows that the best algorithms K-Star and Random Forest for the used Dataset, where Build the models are less time (0 sec and 0.6 sec) and the ROC values.

2.3 Background techniques:

Decision tree:

Decision Tree algorithm is supervised learning algorithms. The decision tree algorithm solves the problem, by using tree representation. every internal node of the tree corresponds to each leaf node corresponds to a class label and attribute.

KNN:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

Extra Trees Classifier:

It's an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy and control overfitting by introducing randomness in the tree- building process.

Random Forest Classifier:

Similar to Extra Trees, Random Forest is an ensemble method that constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Support Vector Machine (SVM):

SVM is a powerful supervised learning algorithm used for classification tasks. It works by finding the hyperplane that best separates the data into different classes, maximizing the margin between classes.

AdaBoost (Adaptive Boosting):

AdaBoost is an ensemble learning technique that combines multiple weak classifiers to create a strong classifier. It adjusts the weights of incorrectly classified instances so that subsequent

classifiers focus more on difficult cases, improving overall accuracy.

Gaussian Naive Bayes:

This is a probabilistic classifier based on Bayes' theorem with an assumption of independence among features. It calculates the probability of an instance belonging to a class based on the features' probabilities.

Gradient Boosting:

Gradient Boosting is another ensemble method that builds trees sequentially, where each new tree corrects errors made by the previous ones. It's particularly effective in regression and classification tasks, offering high predictive power.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

Introduction

The presentation of the Software Requirements Specification (SRS) gives a review of the whole SRS with reason, scope, definitions, abbreviations, contractions, references and diagram of the SRS. The point of this report is to assemble, dissect, and give a top to bottom knowledge of the total "Chronic disease prediction" by characterizing the difficult articulationin detail. The point-by-point necessities of the Indian car purchasing conduct – client related capacities are given in this archive.

3.1 Purpose

The Purpose of the Software Requirements Specification is to give the specialized, Functional and non-useful highlights, needed to build up a web application App. The whole application intended to give client adaptability to finding the briefest as well as efficient way. To put it plainly, the motivation behind this SRS record is to give an itemized outline of our product item, its boundaries and objectives. This archive depicts the task's intended interest group and its UI, equipment and programming prerequisites. It characterizes how our customer, group and crowd see the item and its usefulness.

3.2 Scope

The Scope of this framework is to presents a survey on information digging strategies utilized for the expectation of Chronic disease prediction. It is obvious from the framework that information mining strategy, similar to grouping, is profoundly productive in expectation of Indian car.

3.2 Software Architecture

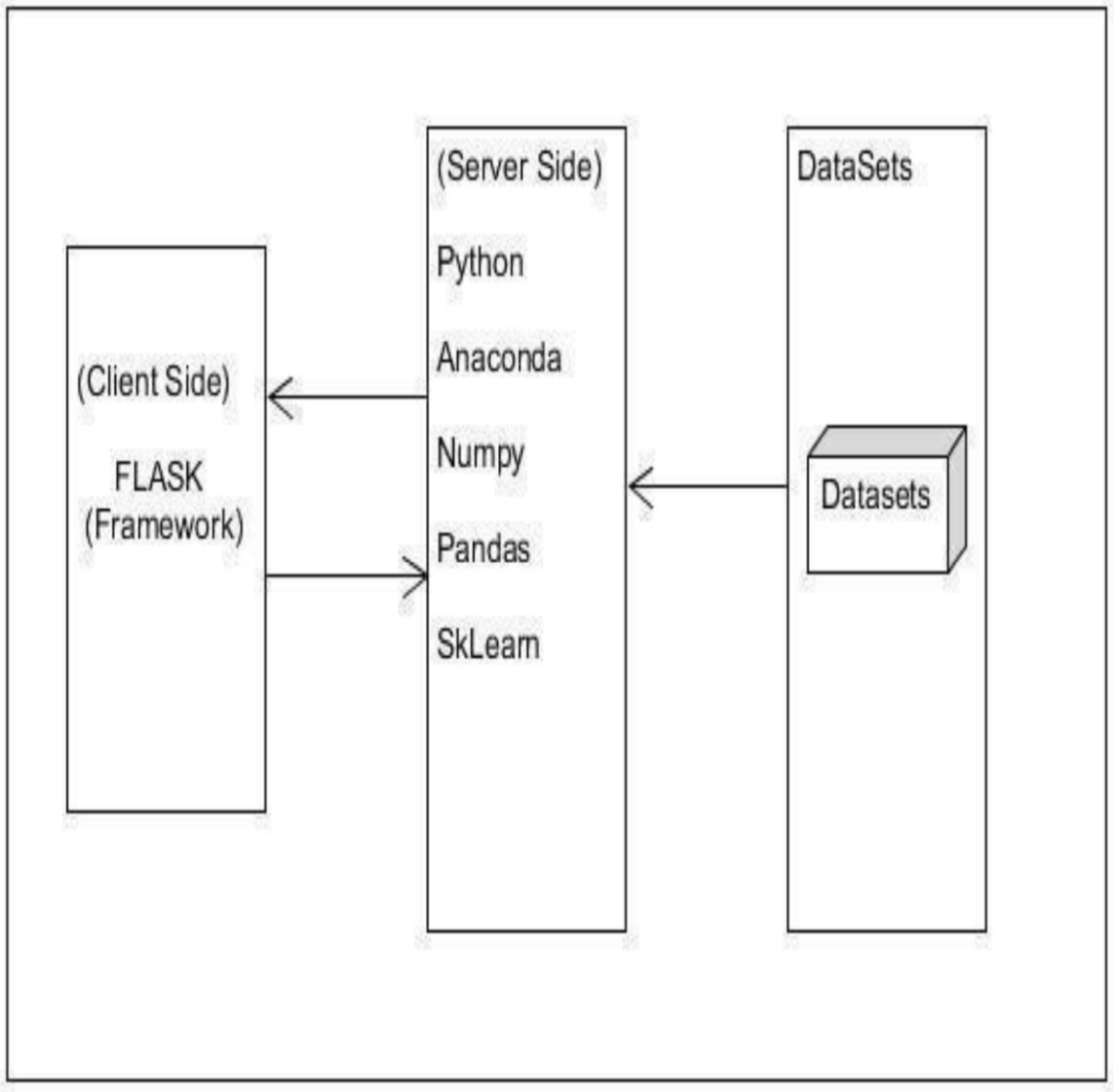


Figure 3.1: Software Architecture

Acronyms And Abbreviation:

3.2.1 Python:

Python is a deciphered, significant level, broadly useful programming language. Made by Guido van Rossum and first delivered in 1991, Python's plan reasoning accentuates code meaningfulness with its prominent utilization of critical whitespace. Its language develops and object-arranged methodology plan to assist software engineers with composing clear, consistent code for little and huge scope ventures.

Python is progressively composed and trash gathered. It underpins numerous programming standards, including procedural, object-arranged, and practical programming. Python is frequently portrayed as a "batteries included" language because of its thorough standard library.

Python is a multi-worldview programming language. Article arranged programming and organized writing computer programs are completely upheld, and a significant number of its highlights uphold useful programming and angle situated programming (counting by metaprogramming and metaobjects (enchantment methods)). Many different standards are upheld by means of expansions, including plan by agreement and rationale programming.

3.2.2 Flask:

Flask is a miniature web system written in Python. It is delegated a microframework in light of the fact that it doesn't need specific apparatuses or libraries.[3] It has no information base deliberation layer, structure approval, or whatever other segments where prior outsider libraries give normal capacities. In any case, Flask upholds augmentations that can include application includes as though they were executed in Flask itself. Augmentations exist for object-social mappers, structure approval, transfer dealing with, different open confirmation advancements and a few basic system related devices. Augmentations are refreshed unmistakably more as often as possible than the center Flask program

3.2.3 Anaconda:

Anaconda is a free and open-source circulation of the programming dialects Python and R . The dissemination accompanies the Python translator and different bundles identified with AI and information science.

Essentially, the thought behind Anaconda is to make it simple for individuals inspired by those fields to introduce all (or a large portion) of the bundles required with a solitary establishment.

An open source bundle and condition the executives framework called Conda, which makes it simple

to introduce/update bundles and make/load situations

AI libraries like TensorFlow, scikit-learn and Theano. Information science libraries like pandas, NumPy and Dask. Perception libraries like Bokeh, Datashader, matplotlib and Holoviews. Jupyter Notebook, a shareable note pad that joins live code, representations and text.

3.2.4 Numpy:

NumPy is the principal bundle for logical registering with Python. It contains in addition toother things:

- Amazing N-dimensional cluster object
- Sophisticated (broadcasting) capacities
- Tools for incorporating C/C++ and Fortran code
- Useful straight polynomial math, Fourier change, and arbitrary number abilities

3.2.5 Pandas:

Pandas is an open source, BSD-authorized library giving elite, simple to-utilize information structures and information investigation apparatuses for the Python programming language.

Pandas is a Num FOCUS supported undertaking. This will help guarantee the achievement of improvement of pandas as an a-list open-source venture, and makes it conceivable to give to the task.

3.3 Feasibility Study:

The feasibility study helps to find solutions to the problems of the project. The solution is given how looks like a new system look like.

3.3.1 Technical Feasibility

The project entitled "Prediction of Chronic disease" is technically feasible because of the below mentioned features. The project is developed in Python. The web server is used to develop "Prediction Chronic disease" is local serve. The local server very neatly coordinates between the design and coding parts. It provides a Graphical User Interface to design an
application while the coding is done in python. At the same time, it provides high-level reliability, availability, and compatibility.

3.3.2 Economic Feasibility

In economic feasibility, cost-benefit analysis is done in which costs and benefits are evaluated. Economic analysis is used for the effectiveness of the proposed system. In economic feasibility, the main task is cost-benefit analysis. The system "Prediction of Chronic disease using Data Mining Techniques" is feasible because it does not exceed the estimated cost and the estimated benefits are equal.

3.3.3 Operational Feasibility

The Project entitled "Prediction of Chronic Kidney disesase " is feasible because of the below mentioned features. The system predicts the chronic disease prediction based on the historical data, further the details of the patient are added to the Data Base. The performance of the Data mining techniques are compared based on their execution time and displayed it through a graph.

3.3.4 Behavior Feasibility

The project entitled "Prediction of Chronic disease using deep learning and Machine Learning" is beneficial because it satisfies the objectives when developed and installed.

3.4 OVERVIEW

Following a section of this document will focus on describing the system in terms of product functions. In the next section, we will address specific requirements of the system, which willing close functional requirements and non-functional requirements.

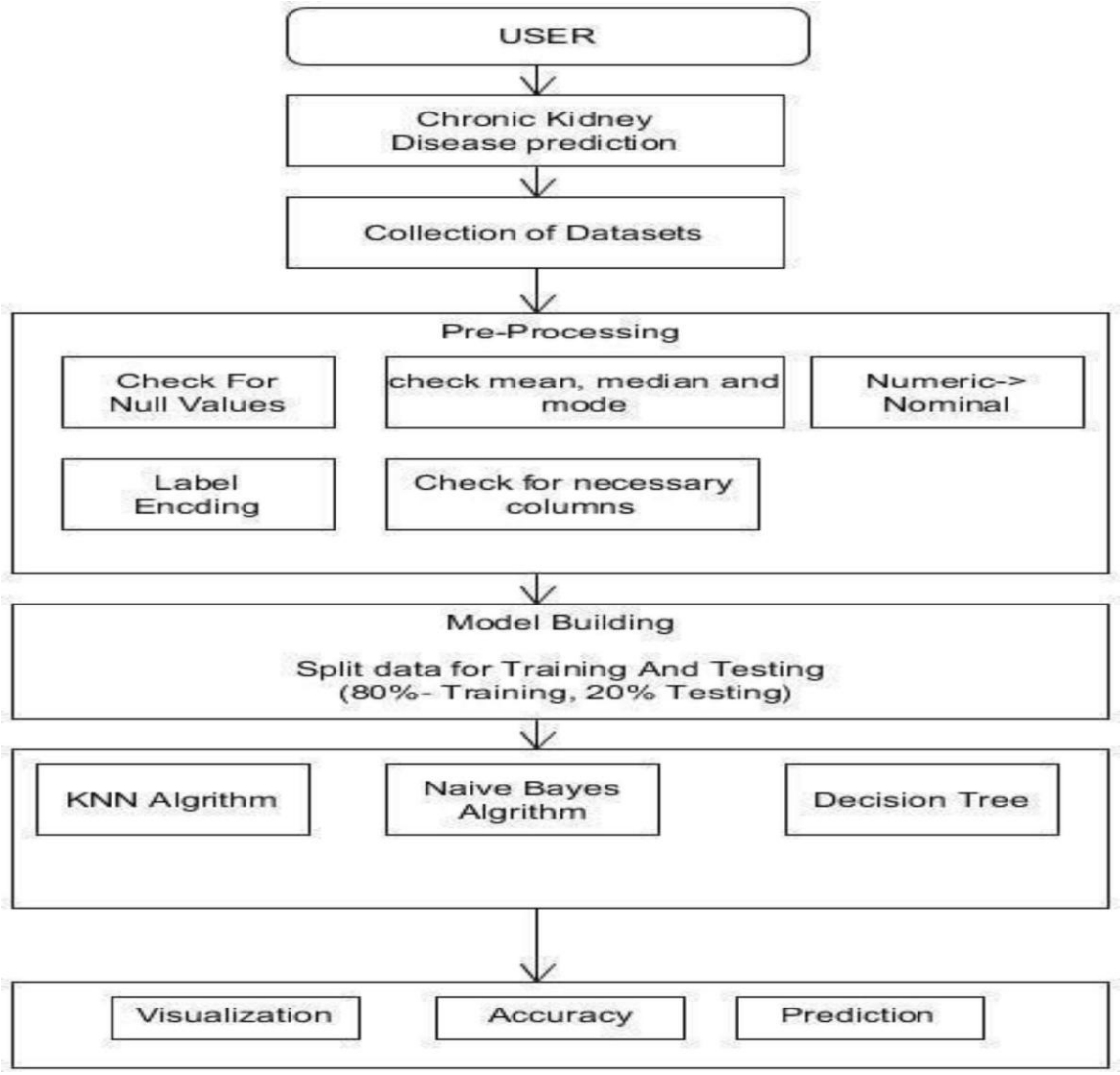


Figure 3.2: Overview

3.5 General description

3.5.1 Product Functions

- Collected datasets of chronic disease prediction from Kaggle
- Pre-processing of obtained datasets
- Select Attributes which helps in predicting the stock
- The selected datasets are trained using SVN Naïve Bayes and KNN
- The trained data sets are tested for Accuracy
- The obtained result is showed in the graph

General constraints

- The system should have enough RAM and Disk Storage Space.
- The Source code must be written in Python for ML.
- The results generated have to be entered in to the system and any error or any value entered out of the boundary will not be understood by the system.

3.6 Specific Requirements

3.6.1 Functional Requirements

A functional requirement defines a function of a system or its component. A role is described as a set of inputs, behaviors, and outputs. Functional requirements may be calculations, technical details, data manipulation, and processing.

The Methods of the system are as follows.

Data preprocessing: Dataset will be added to the preprocessing

a) **Input**: Chronic dataset

b) **Process**: Preprocessing will find missing value and also does feature remove

c) **Output**: preprocessed dataset

d) **Error handling:** If the input file is not a valid one

Feature selection: Selection of the data from a dataset.

a) **Input**: preprocessed dataset

b) **Process**: It will select only important data which is required

c) **Output**: Selected data will be displayed

Splitting of the Data: Training data and Test Data

a) **Input**: Feature selected data

b) **Process**: It will split the data into the train set and test set

c) **Output**: Dataset will be displayed as Train set and Test set and it will be tested for the specific algorithms and performance analysis will be carried out

Functional requirements:

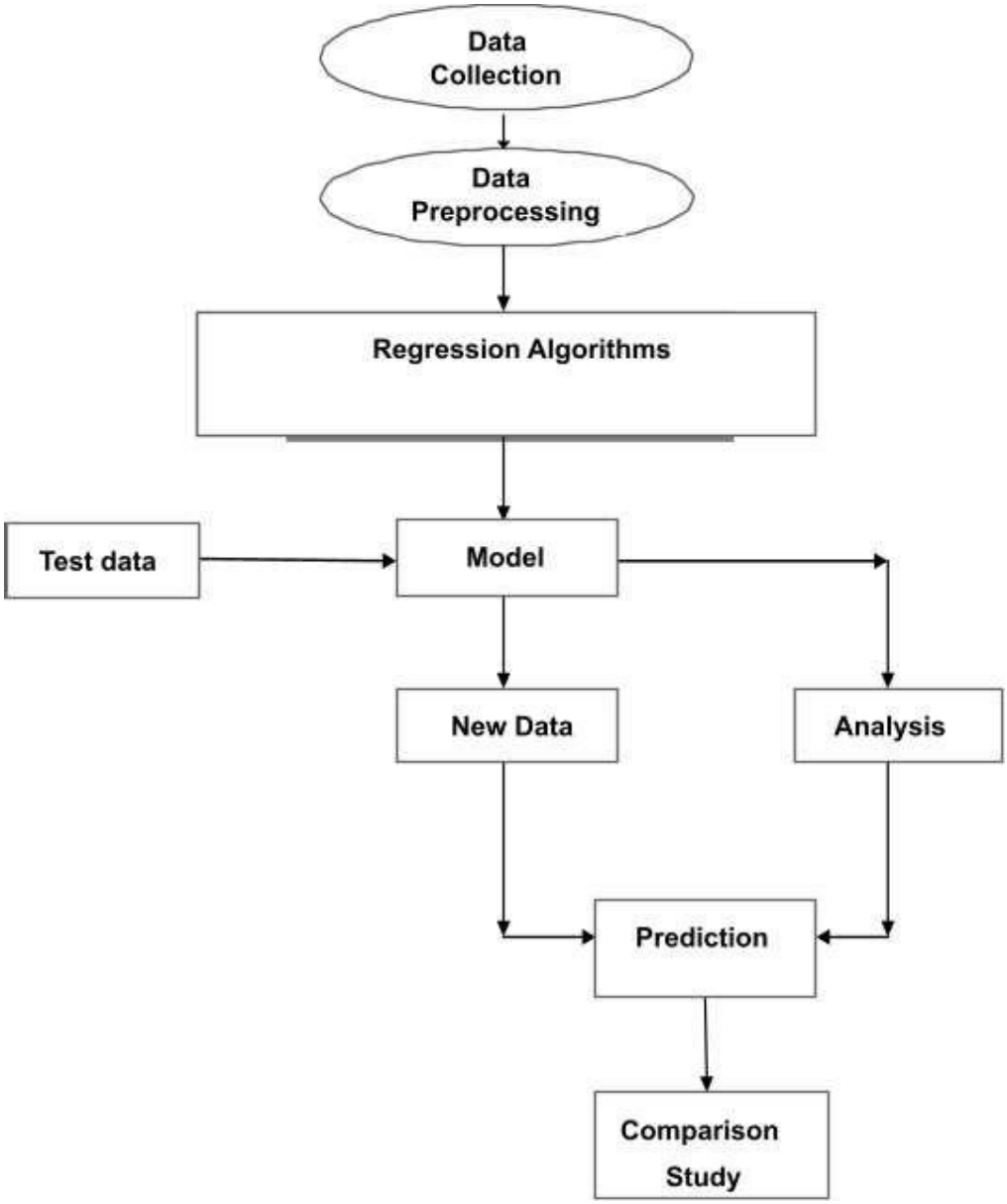


Figure 3.3: Functional requirements

The product consists of a model that functions based on :

1. Collecting Data The data is collected from previous chronic disease records in Kaggle datasets.
2. Then pre-processing the data pre-processing is adding the data.
3. Performing data mining algorithms The data mining algorithms include (Decision tree ,Naïve Bayes and KNN).

4. The algorithm helps in predicting the result based on the parameters
5. The analysis help in the prediction of Disease

Product Functions

1. Uploading Data Uploading data sets.
2. Perform Prediction is done by each algorithm based on the constraints.
3. Comparison Study Prediction results and its stages of each algorithm is represented through graph.

3.7 SYSTEM REQUIREMENTS

3.7.1 HARDWARE REQUIREMENTS

- PROCESSOR : Intel i3
- HARD-DISK : 500GB
- RAM : 4GB or Above

3.7.2 SOFTWARE REQUIREMENTS

- OPERATING SYSTEM : Windows 7 and above
- FRONT END : Html, CSS
- FRAMEWORK : Flask
- LANGUAGE : Python version 3.7
- LIBRARIES : Pandas, Numpy , Sklearn , Scikit
- EDITOR : Jupyter Note Book

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The Software Design will be used to aid in software development for android application by providing the details for how the application should be built. Within the Software Design, specifications are narrative and graphical documentation of the software design for the project includes use case models, sequence diagrams, and other supporting requirement information.

4.2 SCOPE

The design Document is for a primary level system, which will work as a basement for building a system that provides a base level of functionality to show feasibility for large-scaleproduction use. The software Design Document, the focus placed on the generation and modification of the documents. The system will be used in conjunction with other pre-existing systems and will consist largely of a document interaction faced that abstracts document interactions and handling of the document objects.
This Document provides the Design specifications of "Chronic Disease detection".

4.3 DATA FLOW DIAGRAM

LEVEL 0 DFD: Here Dataset will be given as input and will be processed for further implementation.

Fig 4.1: LEVEL 0 DFD

LEVEL 1 DFD: Using python libraries and algorithms prediction will be carried out

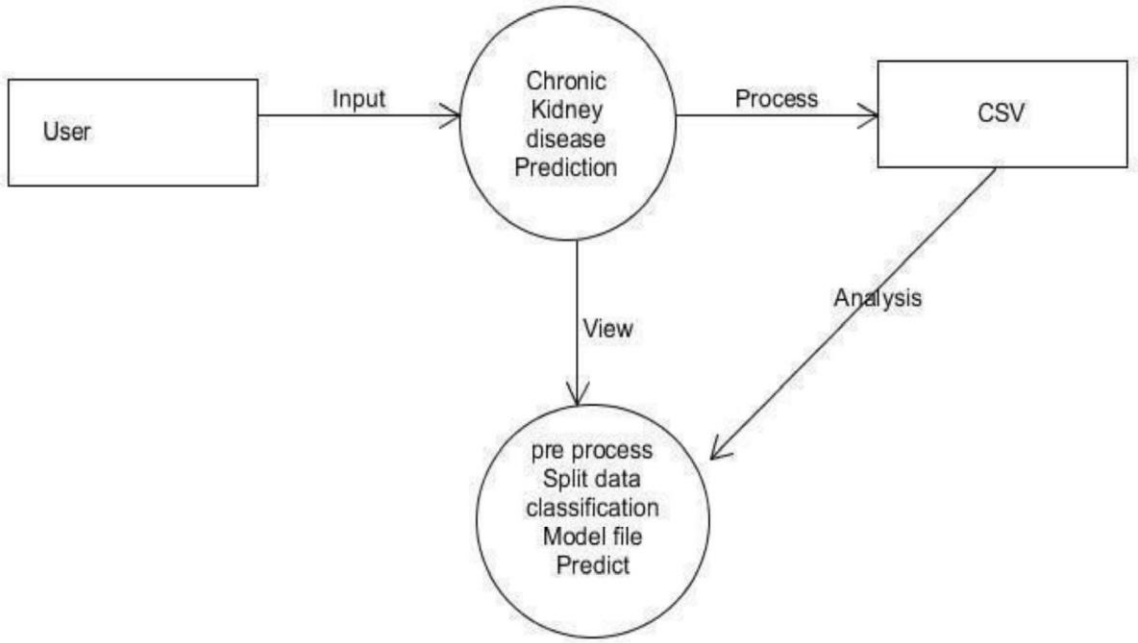
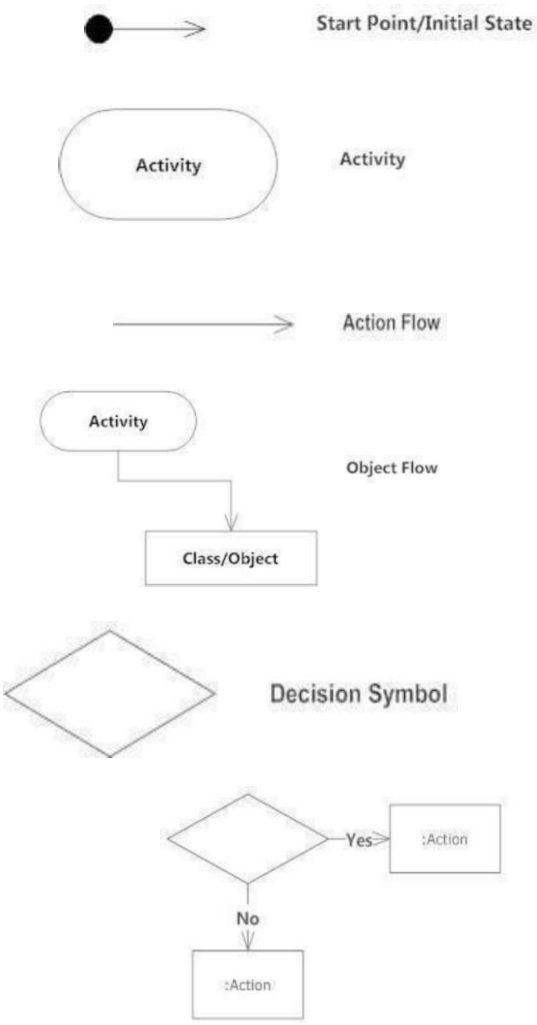


Fig 4.2: LEVEL 1 DFD

4.4 Activity Diagram

An activity diagram outwardly presents a progression of activities or stream of control in a framework like a flowchart or an information stream chart. Action graphs are regularly utilized in business measure demonstrating. They can likewise depict the means in an utilization case chart. Exercises demonstrated can be consecutive and simultaneous. In the two cases, an action outline will have a start (an underlying state) and an end (a last state).


Start Point/Initial State
Activity
Activity
Action Flow
Activity
Object Flow
Class/Object
Decision Symbol
Yes
:Action
No
:Action
Guard Symbols

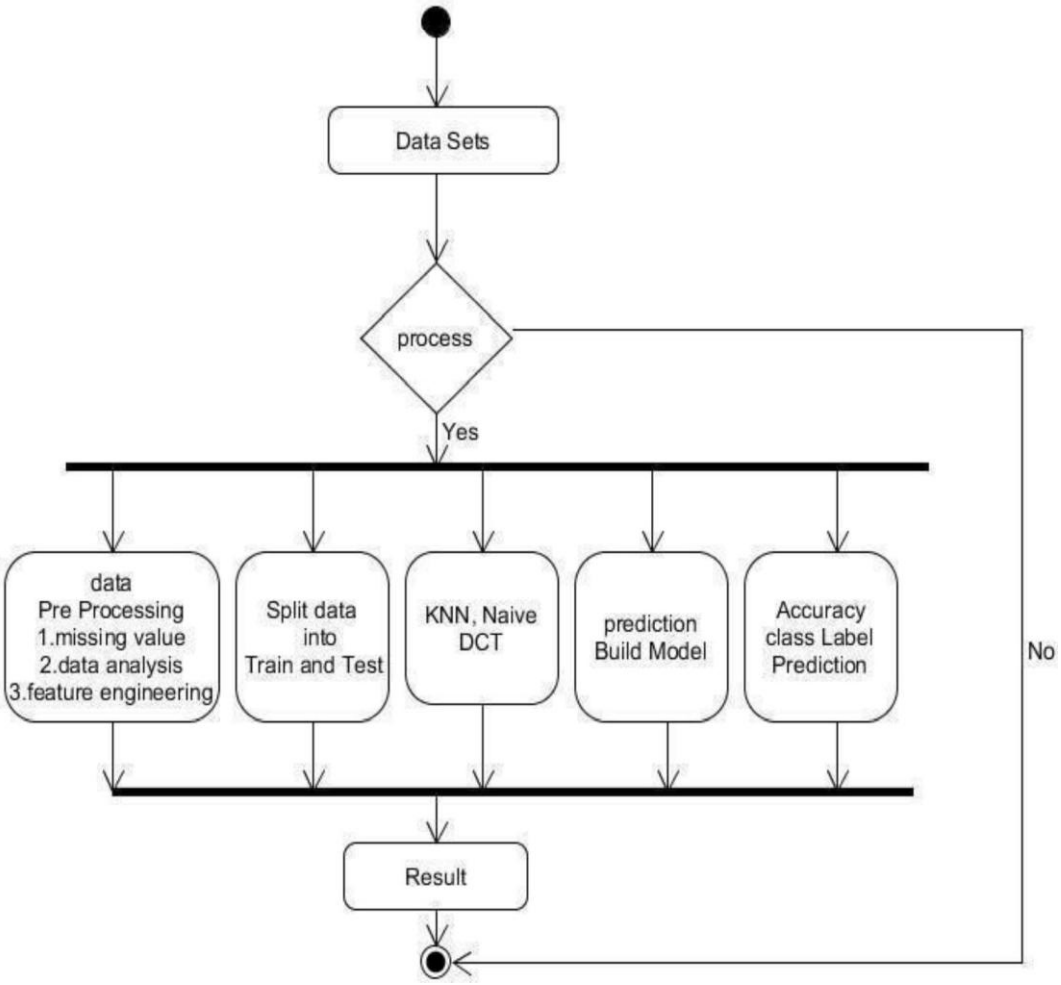


Fig 4.3: ACTIVITY DIAGRAM

4.4 SEQUENCE DIAGRAM

Sequence diagram depict cooperations among classes as far as a trade of messages after some time. They're likewise called occasion charts. A grouping chart is a decent method to envision and approve different runtime situations. These can assist with anticipating how a framework will act and to find duties a class may need to have during the time spent demonstrating another framework.

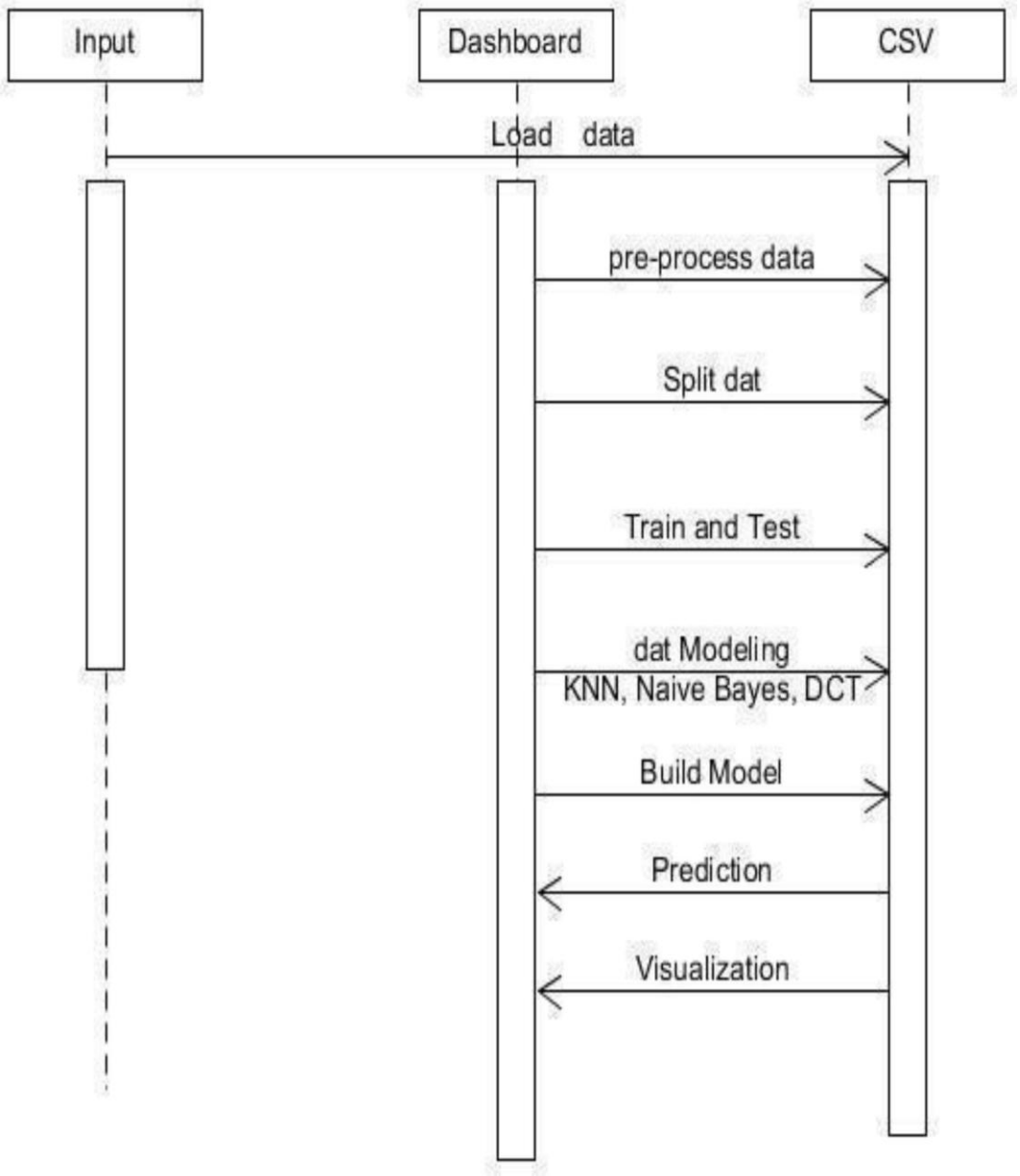


Fig 4.4: SEQUENTIAL DIAGRAM

4.5 USE CASE DIAGRAM

The motivation behind use case diagram is to catch the dynamic part of a framework. In any case, this definition is too nonexclusive to even think about describing the reason, as other four outlines (action, grouping, cooperation, and Statechart) likewise have a similar reason. We will investigate some particular reason, which will recognize it from other four charts.

Use case graphs are utilized to accumulate the prerequisites of a framework including inside and outside impacts. These prerequisites are generally plan necessities. Consequently, when a framework is investigated to accumulate its functionalities, use cases are readied and entertainers are distinguished

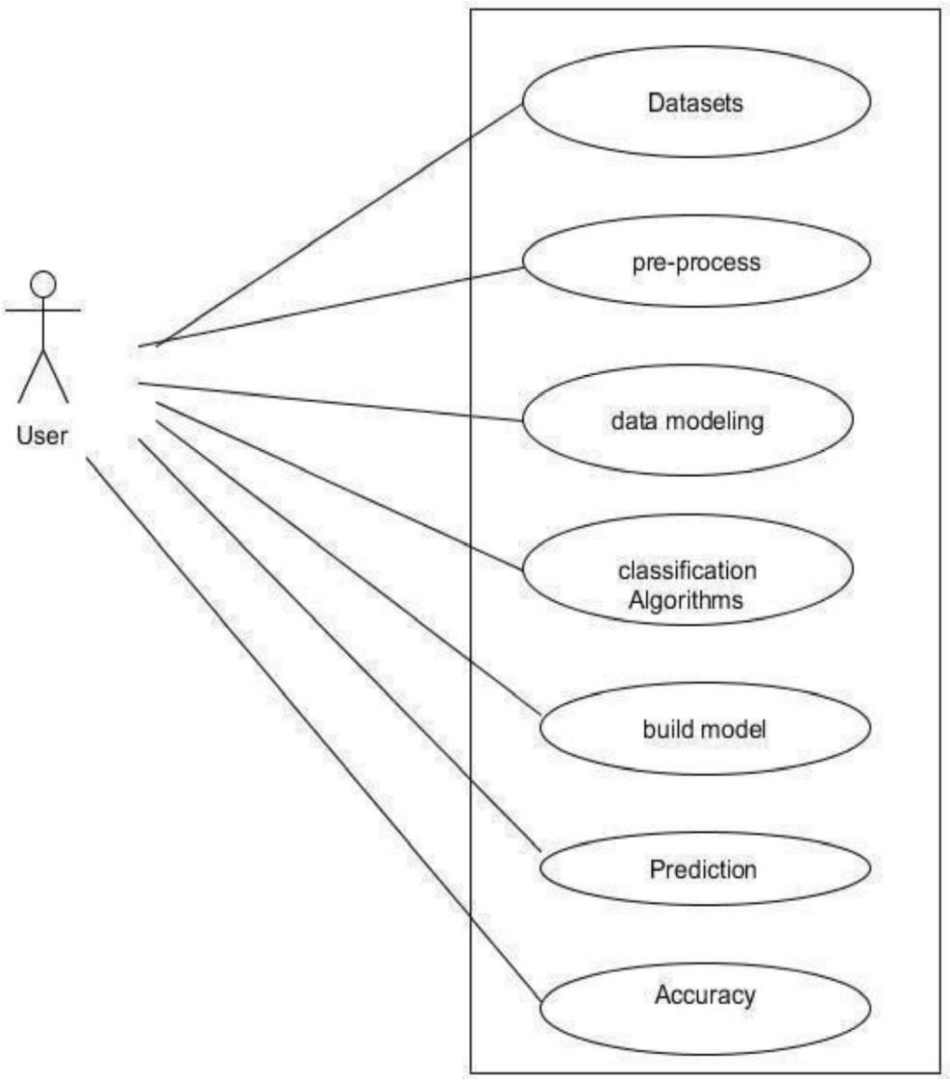


Fig 4.5: USE CASE DIAGRAM

CHAPTER 5

Implementation

The project is implemented using Python which is an object oriented programming language and procedure oriented programming language. Object oriented programming is an approach that provides a way of modularizing program by creating partitioned memory area of both data and function that can be used as a template for creating copies of such module on demand.

This project is implemented using python programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object- oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. The machine Learning techniques are used in this project.

5.1 Machine Learning overview

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. In this article, we'll see basics of Machine Learning, and implementation of a simple machine learning algorithm using python.

Machine learning involves a computer to be trained using a given data set, and use this training to predict the properties of a given new data. For example, we can train a computer by feeding it 1000 images of cats and 1000 more images which are not of a cat, and tell each time to the computer whether a picture is cat or not. Then if we show the computer a new image, then from the above training, the computer should be able to tell whether this new image is a cat or not. The process of training and prediction involves the use of specialized algorithms. We feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. One such algorithm is K- Nearest- Neighbor classification (KNN classification). It takes a test data, and finds k nearest data values to this data from test data set. Then it selects the neighbor of maximum frequency and gives its properties as the prediction result.

5.2 CHALLENGES IN IMPLEMENTING MACHINE LEARNING:

Most insurers recognize the value of machine learning in driving better decision- making and streamlining business processes. Research for the Accenture Technology Vision 2018 shows that more than 90 percent of insurers are using, plan to use or considering using machine learning or AI in the claims or underwriting process.

Some of the challenges insurers typically encounter when adopting machine learning are.

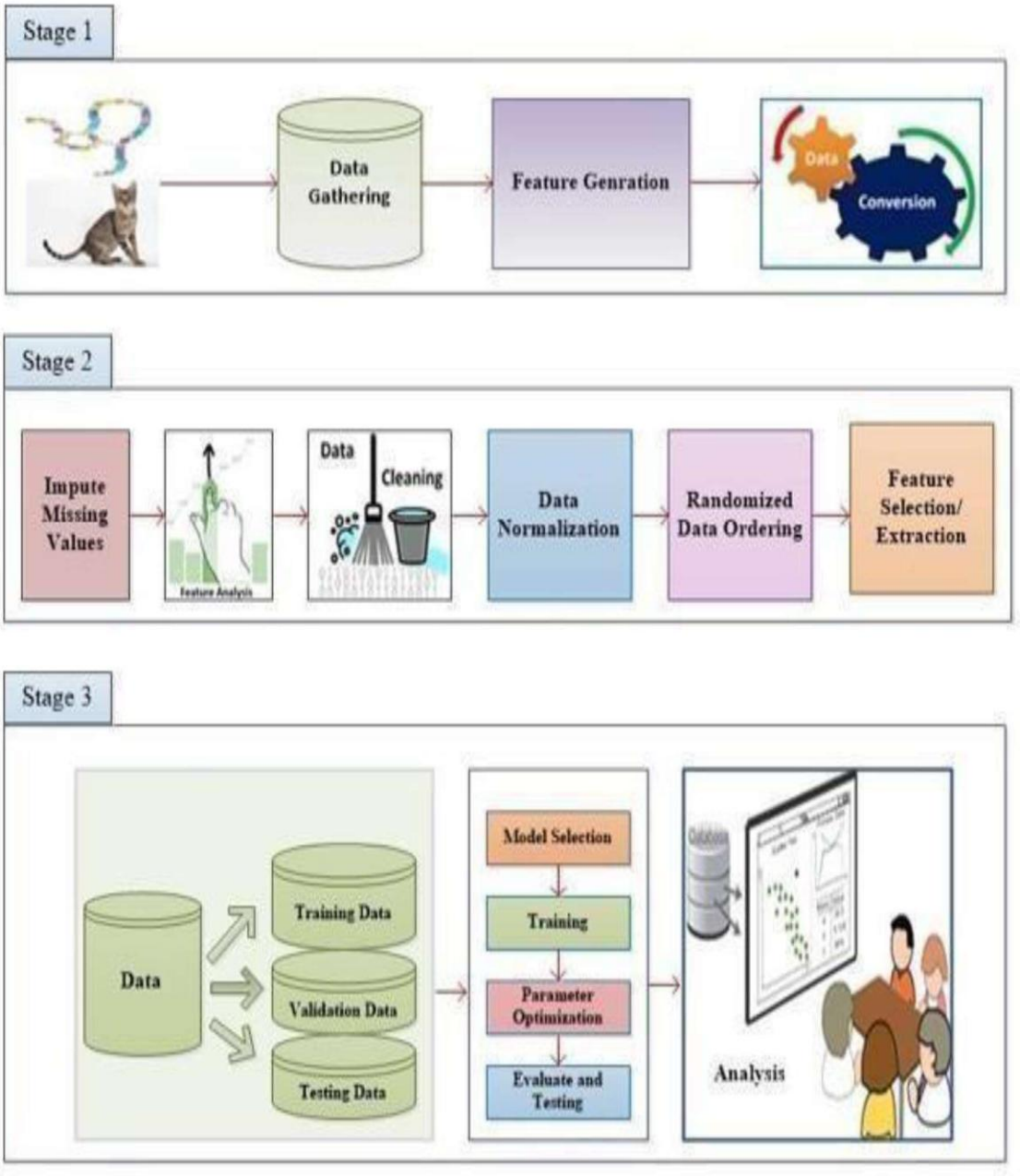
Training requirements AI-powered intellectual systems must be trained in a domain, e.g., claims or billing for an insurer. This requires a separate training system, which insurers find hard to provide for training the AI model. Models need to be trained with huge volumes of documents/transactions to cover all possible scenarios.

Right data source The quality of data used to train predictive models is equally important as the quantity, in the case of machine learning. The datasets need to be representative and balanced so that they can give a better picture and avoid bias. This is important to train predictive models. Generally, insurers struggle to provide relevant data for training AI models

Difficulty in predicting returns It's not very easy to predict improvements that machine learning can bring to a project. For example, it's not easy to plan or budget a project using machine learning, as the funding needs may vary during the project, based on the findings. Therefore, it is almost impossible to predict the return on investment. This makes it hard to get everyone on board the concept and invest in it.

Data security The huge amount of data used for machine learning algorithms has created an additional security risk for insurance companies. With such an increase in collected data and connectivity among applications, there is a risk of data leaks and security breaches. A security incident could lead to personal information falling into the wrong hands. This creates fear in the minds of insurers.

5.3 Architecture:



Stage1:

There are 25 features and 1 class label for every chronic kidney disease record, and the features include basic etc age, bp, sugar, serum creatine, sodium, hemoglobin etc.

age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc
48	80	1.02	1	0		normal	notpreser	notpreser	121	36	1.2			15.4	44	7800
7	50	1.02	4	0		normal	notpreser	notpresent		18	0.8			11.3	38	6000
62	80	1.01	2	3	normal	normal	notpreser	notpreser	423	53	1.8			9.6	31	7500
48	70	1.005	4	0	normal	abnormal	present	notpreser	117	56	3.8	111	2.5	11.2	32	6700
51	80	1.01	2	0	normal	normal	notpreser	notpreser	106	26	1.4			11.6	35	7300
60	90	1.015	3	0			notpreser	notpreser	74	25	1.1	142	3.2	12.2	39	7800
68	70	1.01	0	0		normal	notpreser	notpreser	100	54	24	104	4	12.4	36	
24		1.015	2	4	normal	abnormal	notpreser	notpreser	410	31	1.1			12.4	44	6900
52	100	1.015	3	0	normal	abnormal	present	notpreser	138	60	1.9			10.8	33	9600
53	90	1.02	2	0	abnormal	abnormal	present	notpreser	70	107	7.2	114	3.7	9.5	29	12100
50	60	1.01	2	4		abnormal	present	notpreser	490	55	4			9.4	28	
63	70	1.01	3	0	abnormal	abnormal	present	notpreser	380	60	2.7	131	4.2	10.8	32	4500
68	70	1.015	3	1		normal	present	notpreser	208	72	2.1	138	5.8	9.7	28	12200
68	70						notpreser	notpreser	98	86	4.6	135	3.4	9.8		
68	80	1.01	3	2	normal	abnormal	present	present	157	90	4.1	130	6.4	5.6	16	11000
40	80	1.015	3	0		normal	notpreser	notpreser	76	162	9.6	141	4.9	7.6	24	3800
47	70	1.015	2	0		normal	notpreser	notpreser	99	46	2.2	138	4.1	12.6		
47	80						notpreser	notpreser	114	87	5.2	139	3.7	12.1		
60	100	1.025	0	3		normal	notpreser	notpreser	263	27	1.3	135	4.3	12.7	37	11400
62	60	1.015	1	0		abnormal	present	notpreser	100	31	1.6			10.3	30	5300
61	80	1.015	2	0	abnormal	abnormal	notpreser	notpreser	173	148	3.9	135	5.2	7.7	24	9200
60	90						notpreser	notpresent		180	76	4.5		10.9	32	6200

Stage2:

Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, analysis of data, feature engineering, noisy data etc.

Missing Data:

This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

1. **Ignore the tuples:**

   This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

2. **Fill the Missing values:**

   There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value

Stage 3:

The obtained data from stage is taken into consideration then data is trained using the classification algorithm and obtained result is analyzed and Showed in the graph using python library.

The obtained data is also trained using Machin Learning Algorithms like Extra Tree Classifier , Random Forest Classifier , Decision Tree Classifier , Support Vector Machine , AdaBoost , Gaussian Naïve Bayes , Gradient Boosting , KNN. The obtained result are compared for better Accuracy.

CHAPTER 6

Testing

6.1 Introduction

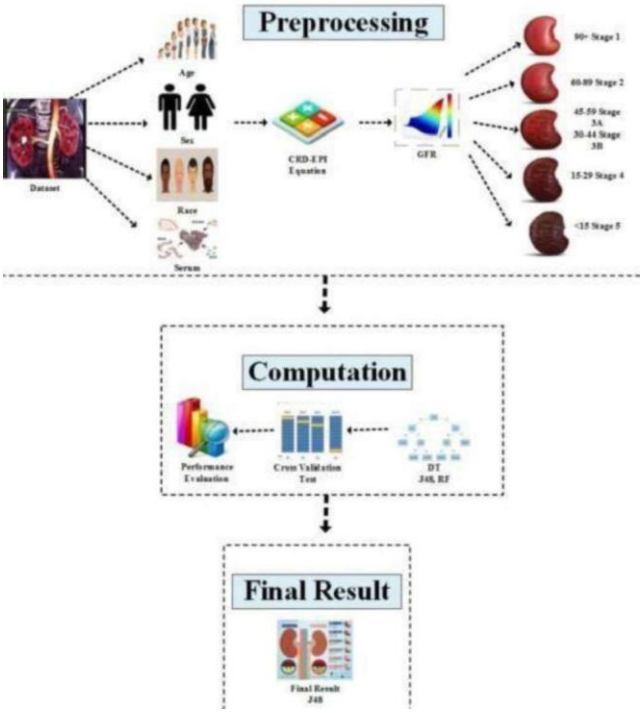
Testing is the way toward running a framework with the expectation of discovering blunders. Testing upgrades the uprightness of the framework by distinguishing the deviations in plans and blunders in the framework. Testing targets distinguishing blunders – prom zones. This aides in the avoidance of mistakes in the framework. Testing additionally adds esteems to the item by affirming the client's necessity.

The primary intention is to distinguish blunders and mistake get-prom zones in a framework. Testing must be intensive and all around arranged. A somewhat tried framework is as terrible as an untested framework. Furthermore, the cost of an untested and under-tried framework is high. The execution is the last and significant stage. It includes client preparation, framework testing so as to guarantee the effective running of the proposed framework. The client tests the framework and changes are made by their requirements. The testing includes the testing of the created framework utilizing different sorts of information. While testing, blunders are noted and rightness is the mode.

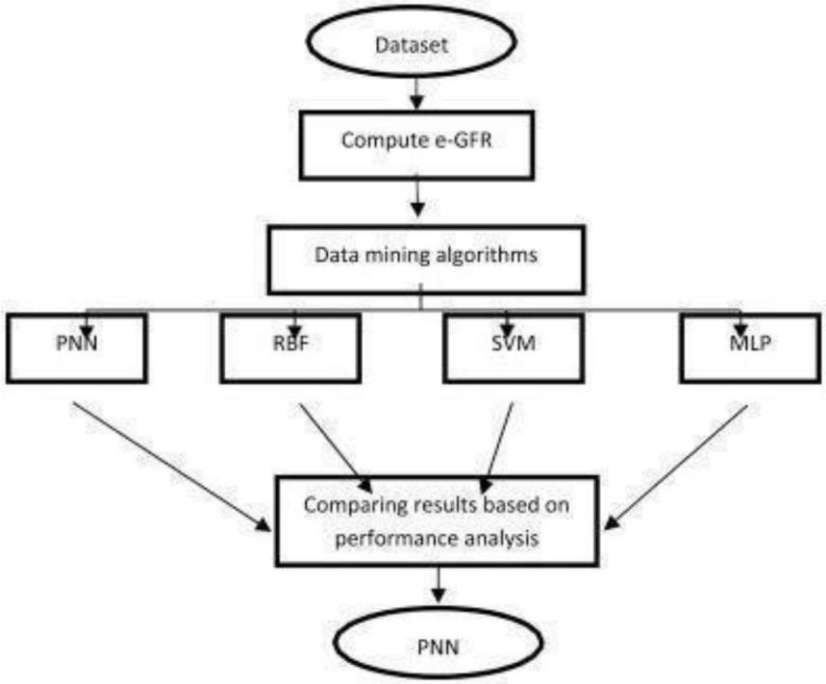
6.2 Objectives Of Testing

- Testing in a cycle of executing a program with the expectation of discovering mistakes.
- A effective experiment is one that reveals an up 'til now unfamiliar blunder.

Framework testing is a phase of usage, which is pointed toward guaranteeing that the framework works accurately and productively according to the client's need before the live activity initiates. As expressed previously, testing is indispensable to the achievement of a framework. Framework testing makes the coherent presumption that if all the framework is right, the objective will be effectively accomplished. A progression of tests are performed before the framework is prepared for the client acknowledgment test.



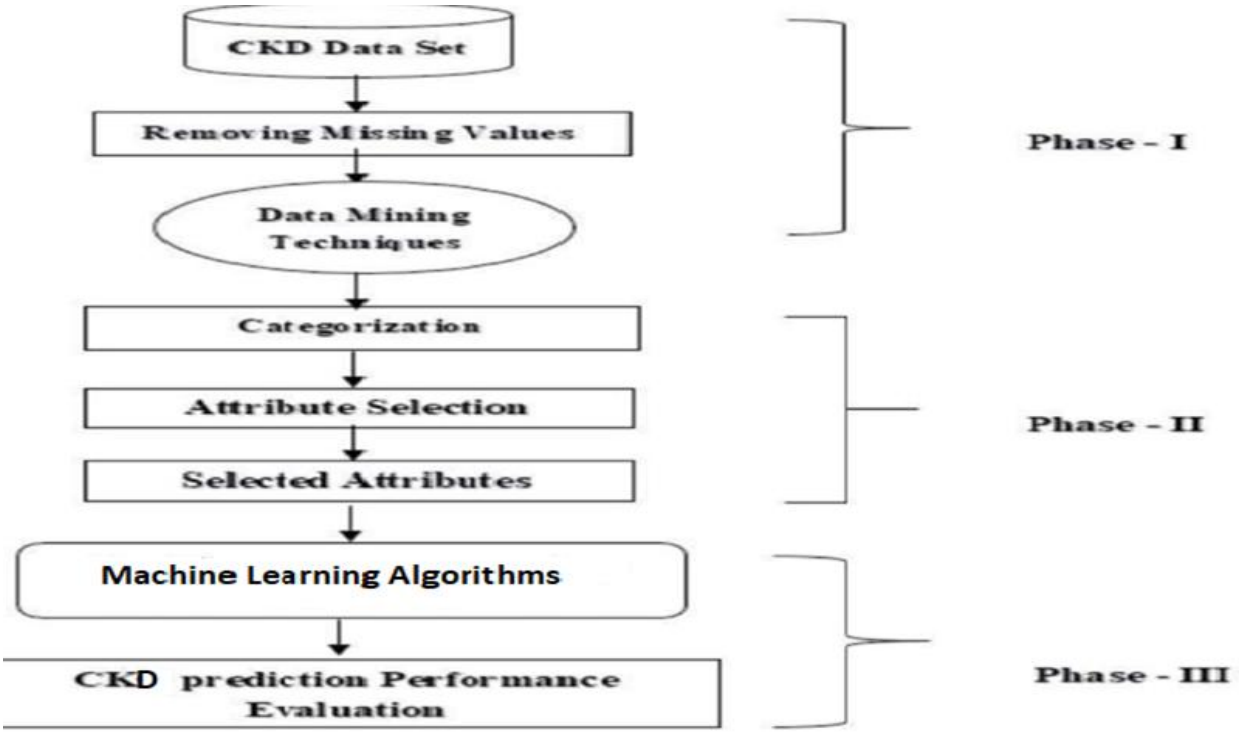
Here is a diagram related to the prediction of kidney disease stages using data mining algorithms:



STAGE	Description	GFR (mL/min/1.73 m^2)
1	Normal	≥ 90
2	Mild	60-89
3	Mild to Moderate	45-59
4	Moderate	30-44
5	Severe	15-29
6	Kidney Failure	<15

Table 1: Stages of CKD based on GFR

Methodology Block Diagram of Chronic Kidney Disease (CKD)



Practical work on JUPYTER NOTEBOOK

Step 1: Import necessary packages.

In [1]:
```
# necessary imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')
%matplotlib inline
pd.set_option('display.max_columns', 26)
```

Step 2: Load the dataset in csv format

In [2]:
```
#loadind data
df= pd.read_csv(r'C:\Users\vikas\OneDrive\Desktop\New folder\Chronic_Kidney_Disease-prediction-Usinag-Machine-Learning\kidney_dis
```

In [3]:
```
df.head()
```

Out[3]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44	7800	5.2	yes	yes	no	good	no	no
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3	38	6000	NaN	no	no	no	good	no	no
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8	NaN	NaN	9.6	31	7500	NaN	no	yes	no	poor	no	yes
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8	111.0	2.5	11.2	32	6700	3.9	yes	no	no	poor	yes	yes
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4	NaN	NaN	11.6	35	7300	4.6	no	no	no	good	no	no

Step 3: Drop the id column and see the header

In [5]:
```
# dropping id column
df.drop('id', axis = 1, inplace = True)
```

In [6]:
```
# rename column names to make it more user-friendly

df.columns = ['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell',
              'pus_cell_clumps', 'bacteria', 'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
              'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count',
              'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'peda_edema',
              'aanemia', 'class']
```

Step 4: Using describe function see the attributes along with description.

```
In [7]: df.head(300)
```

Out[7]:

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria	blood_glucose_random	blood_urea	serum_
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	
...	...	...	...	...	...	...	...	...	...	...	...	
295	44.0	70.0	NaN	NaN	NaN	NaN	NaN	notpresent	notpresent	106.0	25.0	
296	41.0	70.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	125.0	38.0	
297	53.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	116.0	26.0	
298	34.0	60.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	91.0	49.0	
299	73.0	60.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	127.0	48.0	

300 rows × 25 columns

```
In [8]: df.describe()
```

Out[8]:

	age	blood_pressure	specific_gravity	albumin	sugar	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	hae
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	3
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	

Step 5: Using the info() see the values that attributes hold

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       391 non-null    float64
 1   blood_pressure            388 non-null    float64
 2   specific_gravity          353 non-null    float64
 3   albumin                   354 non-null    float64
 4   sugar                     351 non-null    float64
 5   red_blood_cells           248 non-null    object
 6   pus_cell                  335 non-null    object
 7   pus_cell_clumps           396 non-null    object
 8   bacteria                  396 non-null    object
 9   blood_glucose_random      356 non-null    float64
 10  blood_urea                381 non-null    float64
 11  serum_creatinine          383 non-null    float64
 12  sodium                    313 non-null    float64
 13  potassium                 312 non-null    float64
 14  haemoglobin               348 non-null    float64
 15  packed_cell_volume        330 non-null    object
 16  white_blood_cell_count    295 non-null    object
 17  red_blood_cell_count      270 non-null    object
 18  hypertension              398 non-null    object
 19  diabetes_mellitus         398 non-null    object
 20  coronary_artery_disease   398 non-null    object
 21  appetite                  399 non-null    object
 22  peda_edema                399 non-null    object
 23  aanemia                   399 non-null    object
 24  class                     400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

Step 6: Then convert the columns to numerical type

```
In [10]: # converting necessary columns to numerical type

         df['packed_cell_volume'] = pd.to_numeric(df['packed_cell_volume'], errors='coerce')
         df['white_blood_cell_count'] = pd.to_numeric(df['white_blood_cell_count'], errors='coerce')
         df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors='coerce')
```

```
In [11]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       391 non-null    float64
 1   blood_pressure            388 non-null    float64
 2   specific_gravity          353 non-null    float64
 3   albumin                   354 non-null    float64
 4   sugar                     351 non-null    float64
 5   red_blood_cells           248 non-null    object
 6   pus_cell                  335 non-null    object
 7   pus_cell_clumps           396 non-null    object
 8   bacteria                  396 non-null    object
 9   blood_glucose_random      356 non-null    float64
 10  blood_urea                381 non-null    float64
 11  serum_creatinine          383 non-null    float64
 12  sodium                    313 non-null    float64
 13  potassium                 312 non-null    float64
 14  haemoglobin               348 non-null    float64
 15  packed_cell_volume        329 non-null    float64
 16  white_blood_cell_count    294 non-null    float64
 17  red_blood_cell_count      269 non-null    float64
 18  hypertension              398 non-null    object
 19  diabetes_mellitus         398 non-null    object
 20  coronary_artery_disease   398 non-null    object
 21  appetite                  399 non-null    object
 22  peda_edema                399 non-null    object
 23  aanemia                   399 non-null    object
 24  class                     400 non-null    object
dtypes: float64(14), object(11)
```

Step 7: Look for unique values in column.

```
In [13]: # looking at unique values in categorical columns

         for col in cat_cols:
             print(f"{col} has {df[col].unique()} values\n")
```

```
red_blood_cells has [nan 'normal' 'abnormal'] values

pus_cell has ['normal' 'abnormal' nan] values

pus_cell_clumps has ['notpresent' 'present' nan] values

bacteria has ['notpresent' 'present' nan] values

hypertension has ['yes' 'no' nan] values

diabetes_mellitus has ['yes' 'no' ' yes' '\tno' '\tyes' nan] values

coronary_artery_disease has ['no' 'yes' '\tno' nan] values

appetite has ['good' 'poor' nan] values

peda_edema has ['no' 'yes' nan] values

aanemia has ['no' 'yes' nan] values

class has ['ckd' 'ckd\t' 'notckd'] values
```

Step 8: Replace the incorrect values

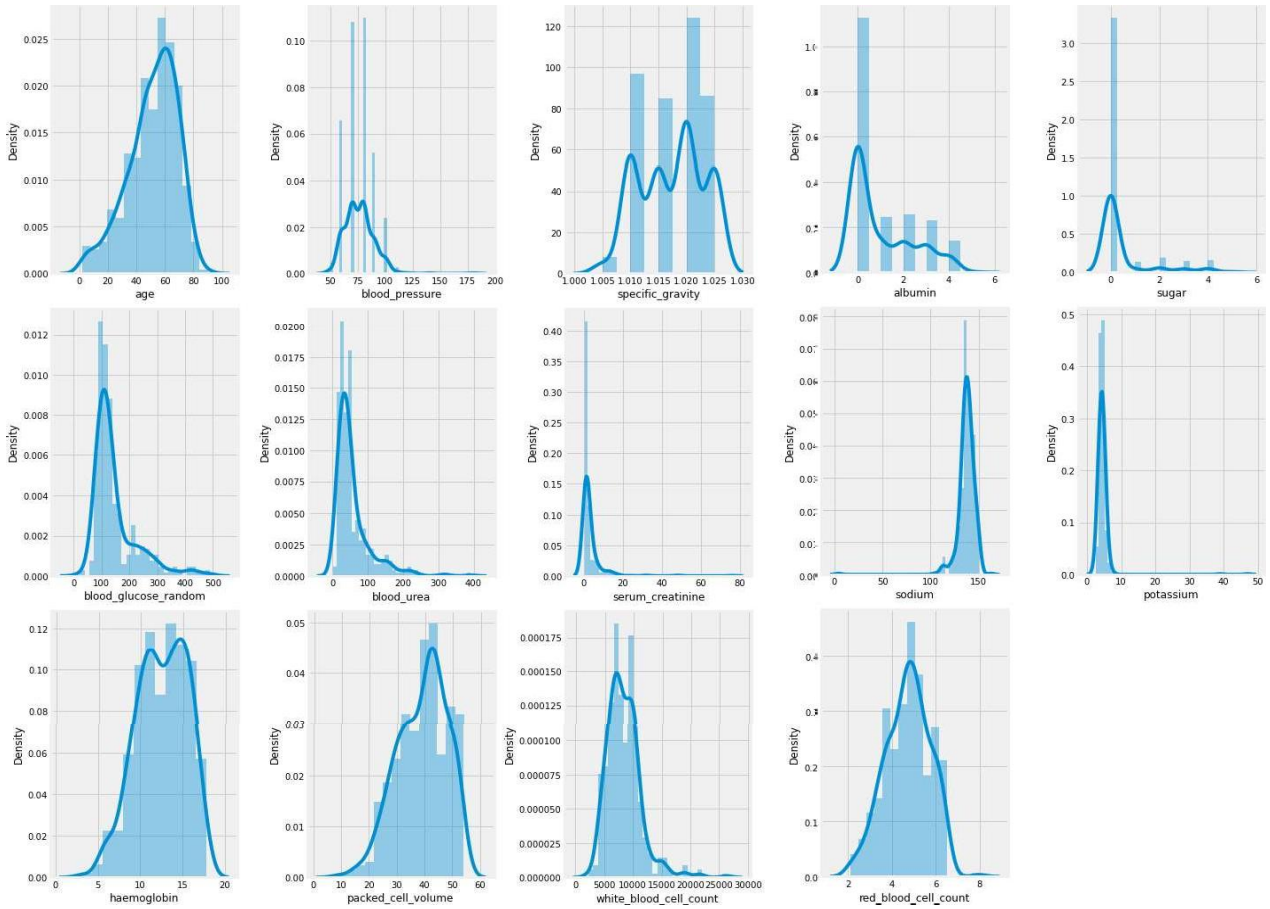
```
In [14]: # replace incorrect values

         df['diabetes_mellitus'].replace(to_replace = {'\tno':'no','\tyes':'yes',' yes':'yes'},inplace=True)

         df['coronary_artery_disease'] = df['coronary_artery_disease'].replace(to_replace = '\tno', value='no')

         df['class'] = df['class'].replace(to_replace = {'ckd\t': 'ckd', 'notckd': 'not ckd'})
```

```
In [15]: df['class'] = df['class'].map({'ckd': 0, 'not ckd': 1})
         df['class'] = pd.to_numeric(df['class'], errors='coerce')
```

```
In [16]: cols = ['diabetes_mellitus', 'coronary_artery_disease', 'class']

         for col in cols:
             print(f"{col} has {df[col].unique()} values\n")
```

```
diabetes_mellitus has ['yes' 'no' nan] values

coronary_artery_disease has ['no' 'yes' nan] values

class has [0 1] values
```

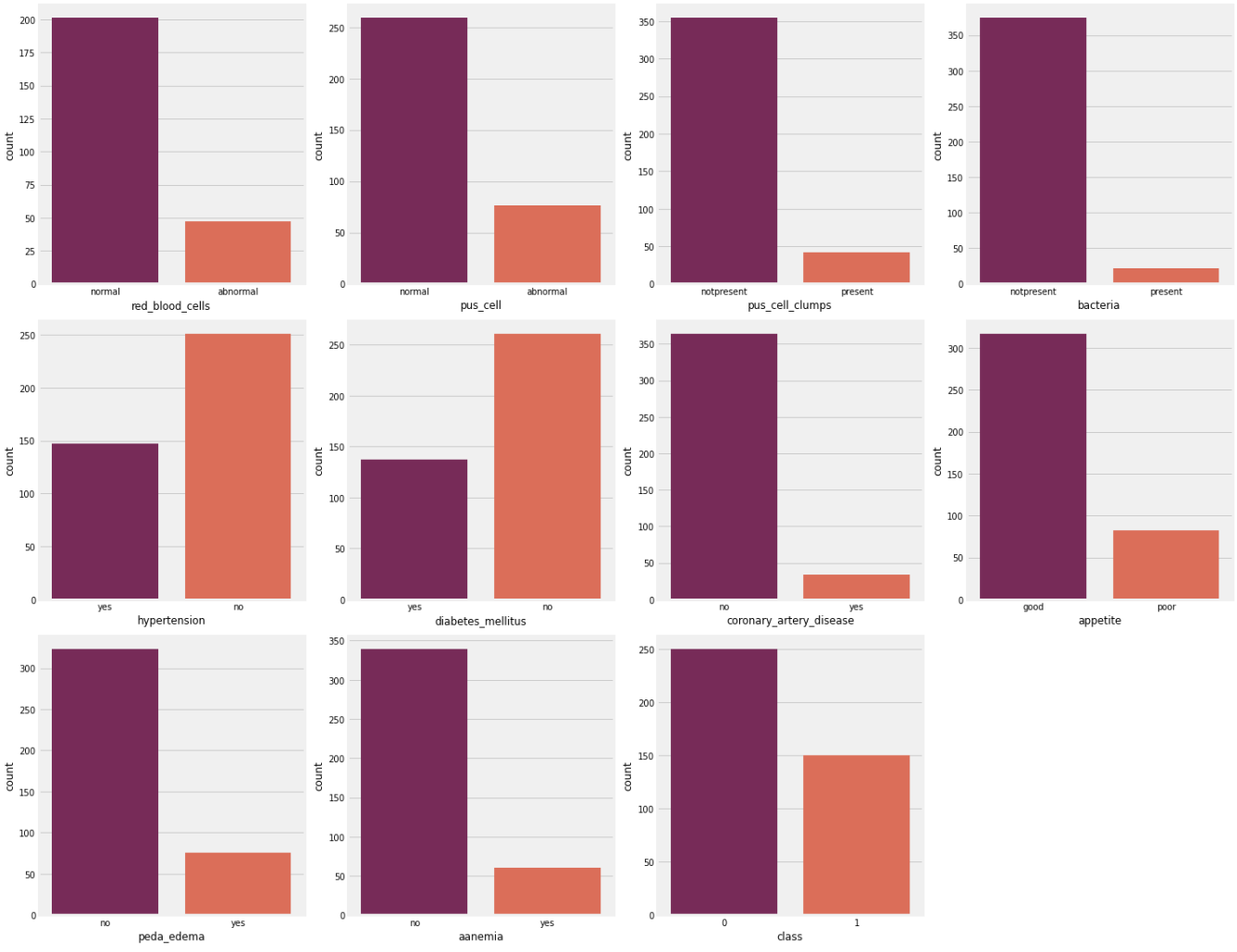
Step 9: Show in graphical form



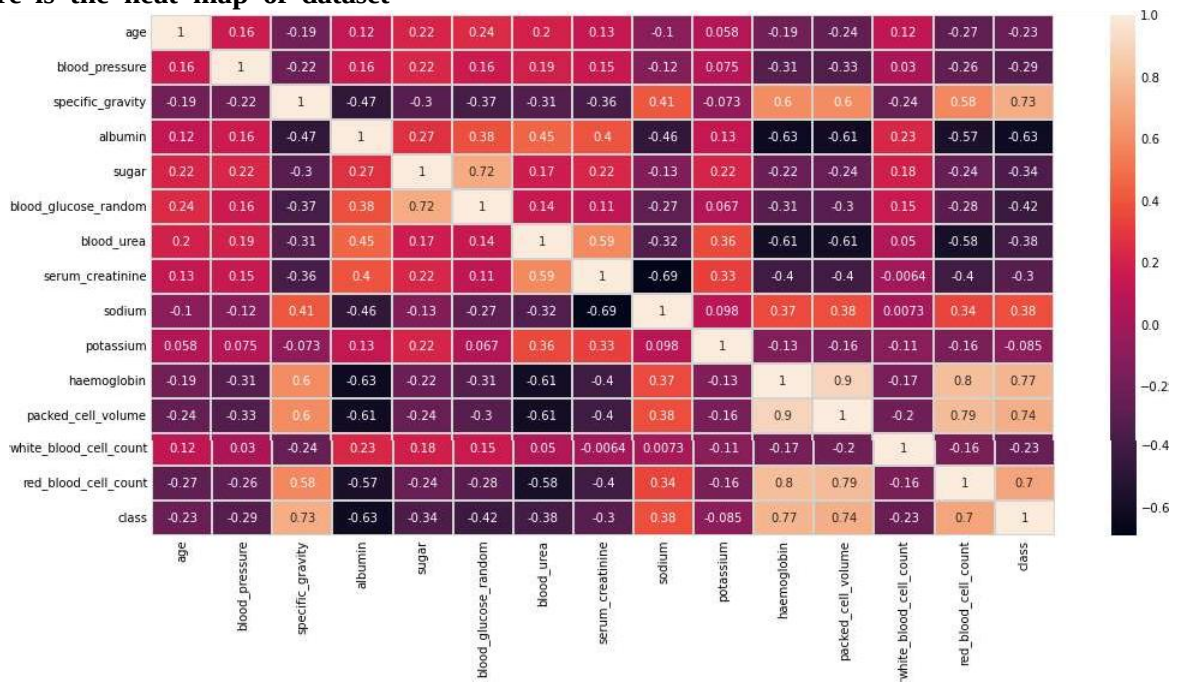
Step 10: Plot the figure of categories

```
In [19]: # looking at categorical columns

plt.figure(figsize = (20, 15))
plotnumber = 1

for column in cat_cols:
    if plotnumber <= 11:
        ax = plt.subplot(3, 4, plotnumber)
        sns.countplot(df[column], palette = 'rocket')
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

GRAPH



Here is the heat map of dataset



Data Preprocessing

```
In [22]: # checking for null values

         df.isna().sum().sort_values(ascending = False)
```

```
Out[22]: red_blood_cells            152
         red_blood_cell_count       131
         white_blood_cell_count     106
         potassium                   88
         sodium                      87
         packed_cell_volume          71
         pus_cell                    65
         haemoglobin                 52
         sugar                       49
         specific_gravity            47
         albumin                     46
         blood_glucose_random        44
         blood_urea                  19
         serum_creatinine            17
         blood_pressure              12
         age                          9
         bacteria                     4
         pus_cell_clumps              4
         hypertension                 2
         diabetes_mellitus            2
         coronary_artery_disease      2
         appetite                     1
         peda_edema                   1
         aanemia                      1
         class                        0
         dtype: int64
```

In [23]:
```
df[num_cols].isnull().sum()
```

Out[23]:
```
age                          9
blood_pressure              12
specific_gravity            47
albumin                     46
sugar                       49
blood_glucose_random        44
blood_urea                  19
serum_creatinine            17
sodium                      87
potassium                   88
haemoglobin                 52
packed_cell_volume          71
white_blood_cell_count     106
red_blood_cell_count       131
dtype: int64
```

In [24]:
```
# filling null values, we will use two methods, random sampling for higher null values and
# mean/mode sampling for lower null values

def random_value_imputation(feature):
    random_sample = df[feature].dropna().sample(df[feature].isna().sum())
    random_sample.index = df[df[feature].isnull()].index
    df.loc[df[feature].isnull(), feature] = random_sample

def impute_mode(feature):
    mode = df[feature].mode()[0]
    df[feature] = df[feature].fillna(mode)
```

In [25]:
```
# filling num_cols null values using random sampling method

for col in num_cols:
    random_value_imputation(col)
```

In [26]:
```
df[num_cols].isnull().sum()
```

Out[26]:
```
age                        0
blood_pressure             0
specific_gravity           0
albumin                    0
sugar                      0
blood_glucose_random       0
blood_urea                 0
serum_creatinine           0
sodium                     0
potassium                  0
haemoglobin                0
packed_cell_volume         0
white_blood_cell_count     0
red_blood_cell_count       0
dtype: int64
```

In [27]:
```
# filling "red_blood_cells" and "pus_cell" using random sampling method and rest of cat_cols using mode imputation

random_value_imputation('red_blood_cells')
random_value_imputation('pus_cell')

for col in cat_cols:
    impute_mode(col)
```

In [28]:
```
df[cat_cols].isnull().sum()
```

Out[28]:
```
red_blood_cells            0
pus_cell                   0
pus_cell_clumps            0
bacteria                   0
hypertension               0
diabetes_mellitus          0
coronary_artery_disease    0
appetite                   0
peda_edema                 0
aanemia                    0
class                      0
dtype: int64
```

In [29]:
```
####FEATURE ENCODING
```

In [30]:
```
for col in cat_cols:
    print(f"{col} has {df[col].nunique()} categories\n")
```

```
red_blood_cells has 2 categories

pus_cell has 2 categories

pus_cell_clumps has 2 categories

bacteria has 2 categories

hypertension has 2 categories

diabetes_mellitus has 2 categories

coronary_artery_disease has 2 categories

appetite has 2 categories

peda_edema has 2 categories

aanemia has 2 categories

class has 2 categories
```

In [35]:
```
###MODEL BUILDING
```

In [36]:
```
ind_col = [col for col in df.columns if col != 'class']
dep_col = 'class'

X = df[ind_col]
y = df[dep_col]
```

In [37]:
```
# splitting data intp training and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

KNN classifier

In [40]:
```
##KNN
```

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of knn

knn_acc = accuracy_score(y_test, knn.predict(X_test))

print(f"Training Accuracy of KNN is {accuracy_score(y_train, knn.predict(X_train))}")
print(f"Test Accuracy of KNN is {knn_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, knn.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, knn.predict(X_test))}")
```

```
Training Accuracy of KNN is 0.7785714285714286
Test Accuracy of KNN is 0.6416666666666667

Confusion Matrix :-
[[52 20]
 [23 25]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.69      0.72      0.71        72
           1       0.56      0.52      0.54        48

    accuracy                           0.64       120
   macro avg       0.62      0.62      0.62       120
weighted avg       0.64      0.64      0.64       120
```

Gradient Boosting

In [43]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

SEED = 23

gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of GradientBoosting

gbc_acc=accuracy_score(y_test, gbc.predict(X_test))

print(f"Training Accuracy of GradientBoosting is {accuracy_score(y_train, gbc.predict(X_train))}")
print(f"Test Accuracy of GradientBoosting is {gbc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, gbc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, gbc.predict(X_test))}")
```

```
Training Accuracy of GradientBoosting is 1.0
Test Accuracy of GradientBoosting is 0.9916666666666667

Confusion Matrix :-
[[72  0]
 [ 1 47]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        72
           1       1.00      0.98      0.99        48

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

Gaussian Naive Bayes

In [45]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

gnb = GaussianNB()
gnb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of GradientBoosting

gnb_acc=accuracy_score(y_test, gnb.predict(X_test))

print(f"Training Accuracy of Gaussian Naive Bayes is {accuracy_score(y_train, gnb.predict(X_train))}")
print(f"Test Accuracy of Gaussian Naive Bayes is {gnb_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, gnb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, gnb.predict(X_test))}")
```

```
Training Accuracy of Gaussian Naive Bayes is 0.9642857142857143
Test Accuracy of Gaussian Naive Bayes is 0.9416666666666667

Confusion Matrix :-
[[69  3]
 [ 4 44]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.96      0.95        72
           1       0.94      0.92      0.93        48

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

AdaBoost

In [47]:
```
from sklearn.ensemble import AdaBoostClassifier
import warnings
warnings.filterwarnings("ignore")

adb = AdaBoostClassifier()
adb_model = adb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of GradientBoosting

adb_acc=accuracy_score(y_test, adb.predict(X_test))

print(f"Training Accuracy of Gaussian Naive Bayes is {accuracy_score(y_train, adb.predict(X_train))}")
print(f"Test Accuracy of Gaussian Naive Bayes is {gnb_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, adb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, adb.predict(X_test))}")
```

```
Training Accuracy of Gaussian Naive Bayes is 1.0
Test Accuracy of Gaussian Naive Bayes is 0.9416666666666667

Confusion Matrix :-
[[70  2]
 [ 2 46]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        72
           1       0.96      0.96      0.96        48

    accuracy                           0.97       120
   macro avg       0.97      0.97      0.97       120
weighted avg       0.97      0.97      0.97       120
```

Support Vector Machine

In [49]:
```
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)

svm_model = svm.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of GradientBoosting

svm_acc=accuracy_score(y_test, svm.predict(X_test))

print(f"Training Accuracy of Support vector machine is {accuracy_score(y_train, svm.predict(X_train))}")
print(f"Test Accuracy of Support vector machine is {svm_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, svm.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, svm.predict(X_test))}")
```

```
Training Accuracy of Support vector machine is 1.0
Test Accuracy of Support vector machine is 0.6

Confusion Matrix :-
[[72  0]
 [48  0]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.60      1.00      0.75        72
           1       0.00      0.00      0.00        48

    accuracy                           0.60       120
   macro avg       0.30      0.50      0.38       120
weighted avg       0.36      0.60      0.45       120
```

Decision Tree

In [51]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

Out[51]:

In [52]:
```
# accuracy score, confusion matrix and classification report of decision tree

dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train, dtc.predict(X_train))}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, dtc.predict(X_test))}")
```

```
Training Accuracy of Decision Tree Classifier is 1.0
Test Accuracy of Decision Tree Classifier is 0.9916666666666667

Confusion Matrix :-
[[72  0]
 [ 1 47]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        72
           1       1.00      0.98      0.99        48

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

Random Forest

In [56]:
```
##RANDOM FOREST CLASSIFIER
```

In [57]:
```
from sklearn.ensemble import RandomForestClassifier

rd_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 11, max_features = 'sqrt', min_samples_leaf = 2, min_samples_s
rd_clf.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of random forest

rd_clf_acc = accuracy_score(y_test, rd_clf.predict(X_test))

print(f"Training Accuracy of Random Forest Classifier is {accuracy_score(y_train, rd_clf.predict(X_train))}")
print(f"Test Accuracy of Random Forest Classifier is {rd_clf_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, rd_clf.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, rd_clf.predict(X_test))}")
```

```
Training Accuracy of Random Forest Classifier is 0.9964285714285714
Test Accuracy of Random Forest Classifier is 0.9666666666666667

Confusion Matrix :-
[[72  0]
 [ 4 44]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        72
           1       1.00      0.92      0.96        48

    accuracy                           0.97       120
   macro avg       0.97      0.96      0.96       120
weighted avg       0.97      0.97      0.97       120
```

Extra Tree Classifier

In [59]:
```
from sklearn.ensemble import ExtraTreesClassifier

etc = ExtraTreesClassifier()
etc.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of extra trees classifier

etc_acc = accuracy_score(y_test, etc.predict(X_test))

print(f"Training Accuracy of Extra Trees Classifier is {accuracy_score(y_train, etc.predict(X_train))}")
print(f"Test Accuracy of Extra Trees Classifier is {etc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, etc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, etc.predict(X_test))}")
```

```
Training Accuracy of Extra Trees Classifier is 1.0
Test Accuracy of Extra Trees Classifier is 0.9833333333333333

Confusion Matrix :-
[[72  0]
 [ 2 46]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        72
           1       1.00      0.96      0.98        48

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

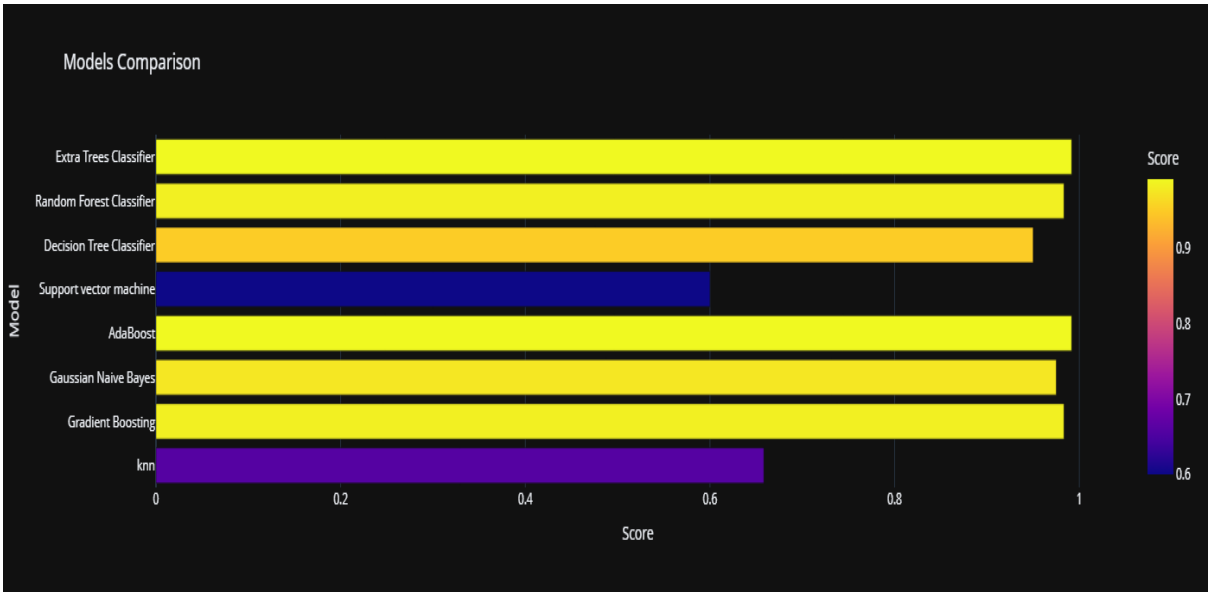
Model Comparison

In [61]:
```
models = pd.DataFrame({
    'Model' : [ 'knn', 'Gradient Boosting' ,'Gaussian Naive Bayes', 'AdaBoost','Support vector machine','Decision Tree Classifier
    'Score' : [knn_acc, gbc_acc ,gnb_acc,adb_acc,svm_acc, dtc_acc, rd_clf_acc, etc_acc]
})


models.sort_values(by = 'Score', ascending = False)
```

Out[61]:

	Model	Score
3	AdaBoost	0.991667
1	Gradient Boosting	0.983333
7	Extra Trees Classifier	0.983333
6	Random Forest Classifier	0.975000
2	Gaussian Naive Bayes	0.966667
5	Decision Tree Classifier	0.916667
0	knn	0.758333
4	Support vector machine	0.600000

In [62]:
```
import plotly.express as px
px.bar(data_frame = models, x = 'Score', y = 'Model', color = 'Score', template = 'plotly_dark',
       title = 'Models Comparison')
```



FLASK PROGRAMMING CODE

```
from flask import Flask, render_template, request, flash, redirect
import pickle
import numpy as np
from PIL import Image

app = Flask(__name__)

def predict(values, dic):

    if len(values) == 18:
        model = pickle.load(open('model.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]

@app.route("/")
def home():
    return render_template('home.html')

@app.route("/kidney", methods=['GET', 'POST'])
def kidneyPage():
    return render_template('kidney.html')

@app.route("/predict", methods = ['POST', 'GET'])
def predictPage():
    try:
        if request.method == 'POST':
            to_predict_dict = request.form.to_dict()
            to_predict_list = list(map(float, list(to_predict_dict.values())))
            pred = predict(to_predict_list, to_predict_dict)
    except Exception as e:
        print(e)
```

```
        message = "Please enter valid Data"
        return render_template("home.html", message = message)

    return render_template('predict.html', pred = pred)

if __name__== '__main__':
 app.run(debug = True)
```

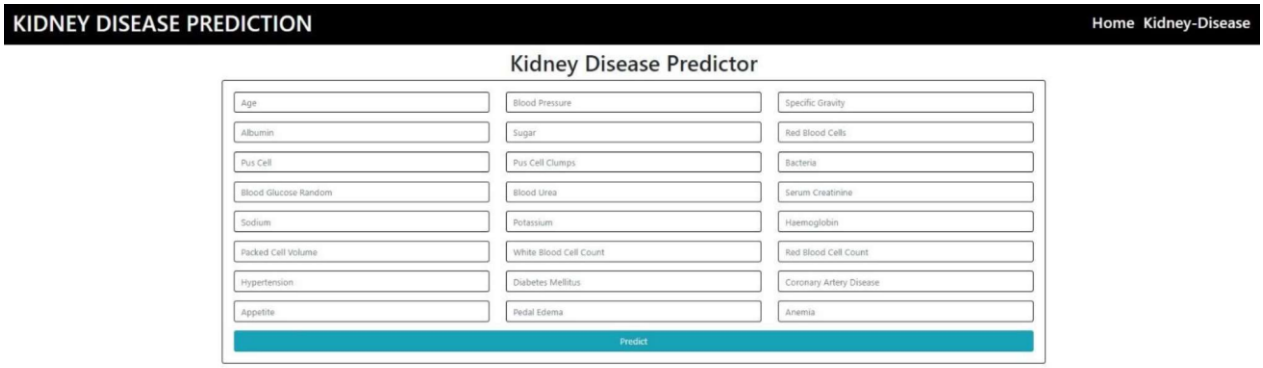
FRONTEND HTML CODE PAGES

Predict .Html :-

```
{% extends 'main.html' %}
{% block content %}
    <div class="row"  style="margin-bottom: 477px;">
        <div class="col-md-3"></div>
        <div class="col-md-6">
            {% if pred == 1 %}
                <div class="jumbotron ">
  <h1 class="display-4">You have a Kidney Disease !</h1>
  <p class="lead">Please Consult the Doctor Immideately. It was too risky without consultation. Make sure of health in your diet.</p>
  <hr class="my-4">
  <p>Proper Doctor Consultation Needed.</p>
  <p class="lead">
    <a class="btn btn-primary btn-lg" href="https://www.who.int/" role="button">Learn more</a>
  </p>
</div>
            {% else %}
                <div class="jumbotron">
  <h1 class="display-4">Great! You are Healthy</h1>
  <p class="lead">You are Absolutely Alright ! There is no Marks for Kidney Disease. Enjot=y you life with full of Happiness.</p>
  <hr class="my-4">
  <p>Be careful at your health. Nothing is important than your health.</p>
```

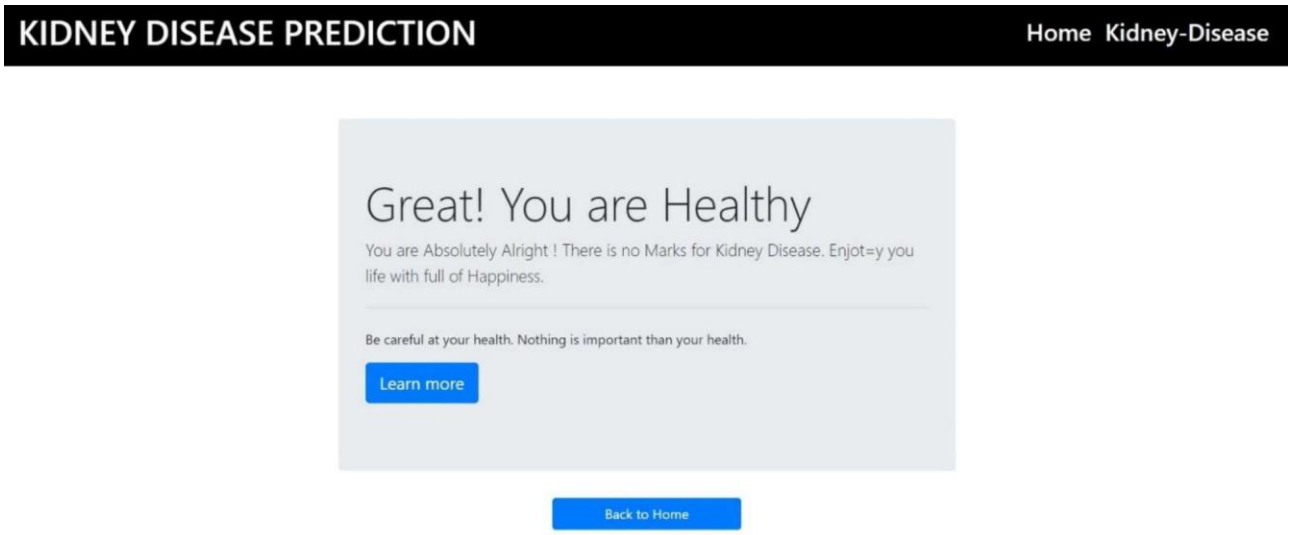
```
  <p class="lead">
    <a class="btn btn-primary btn-lg" href="https://www.who.int/" role="button">Learn more</a>
  </p>
</div>
          {% endif %}
          <div class="row">
             <div class="col-md-4"></div>
             <div class="col-md-4"><a href="{{ url_for('home') }}" class="btn btn-block btn-primary">Back
to Home</a></div>
             <div class="col-md-4"></div>
          </div>
        </div>
        <div class="col-md-3"></div>
    </div>
{% endblock %}
```

Predict.Html page output:-

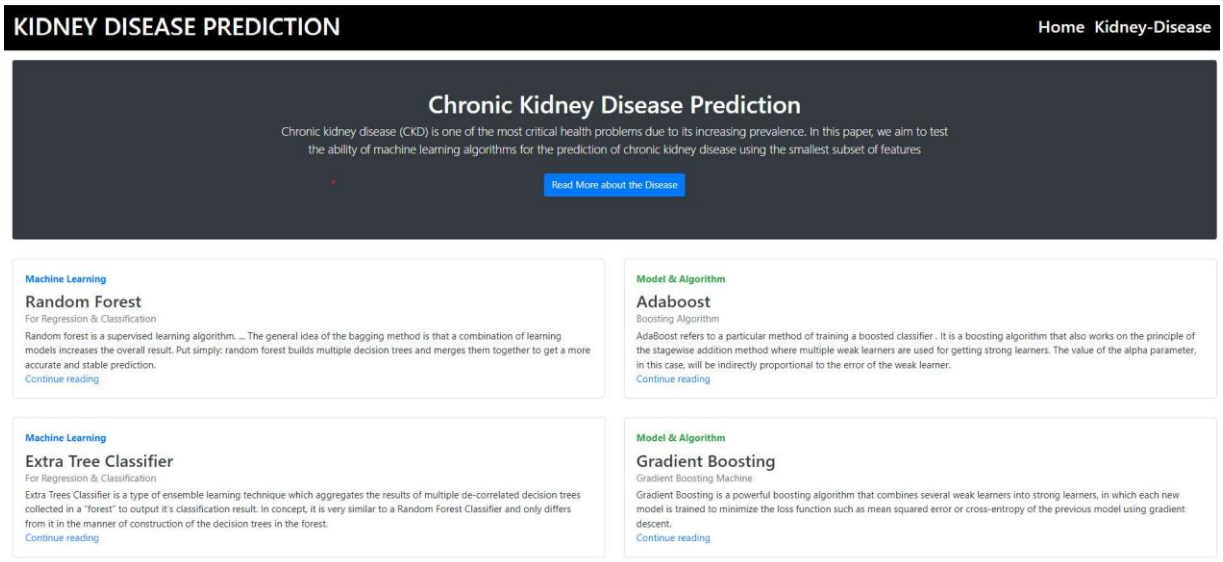


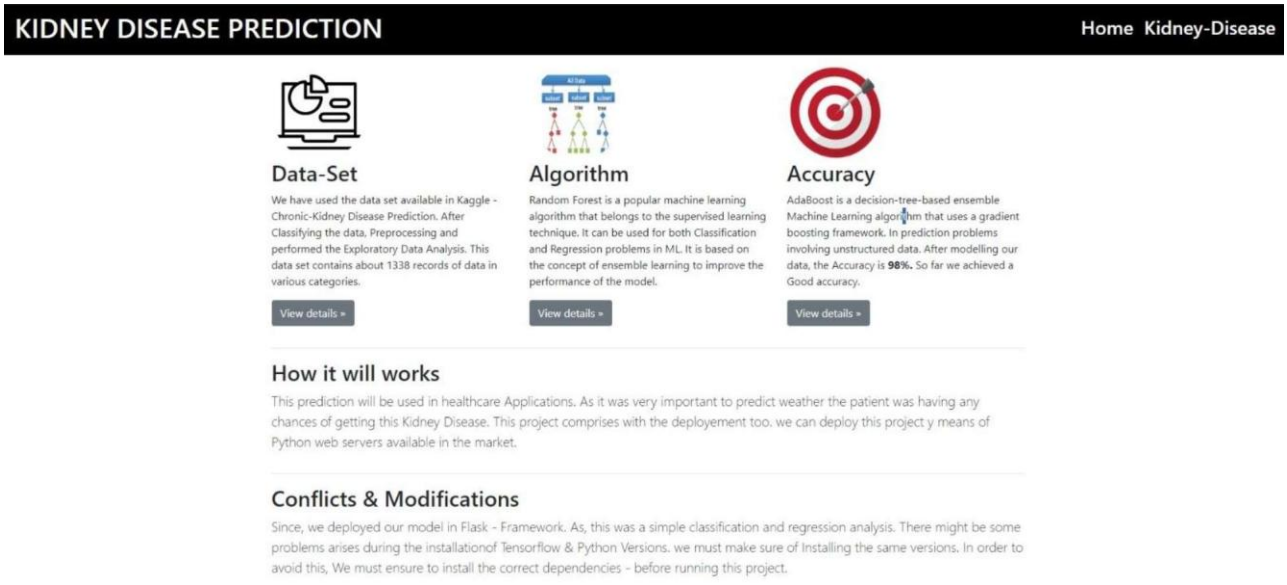
Sample-Inputs in the Data Set

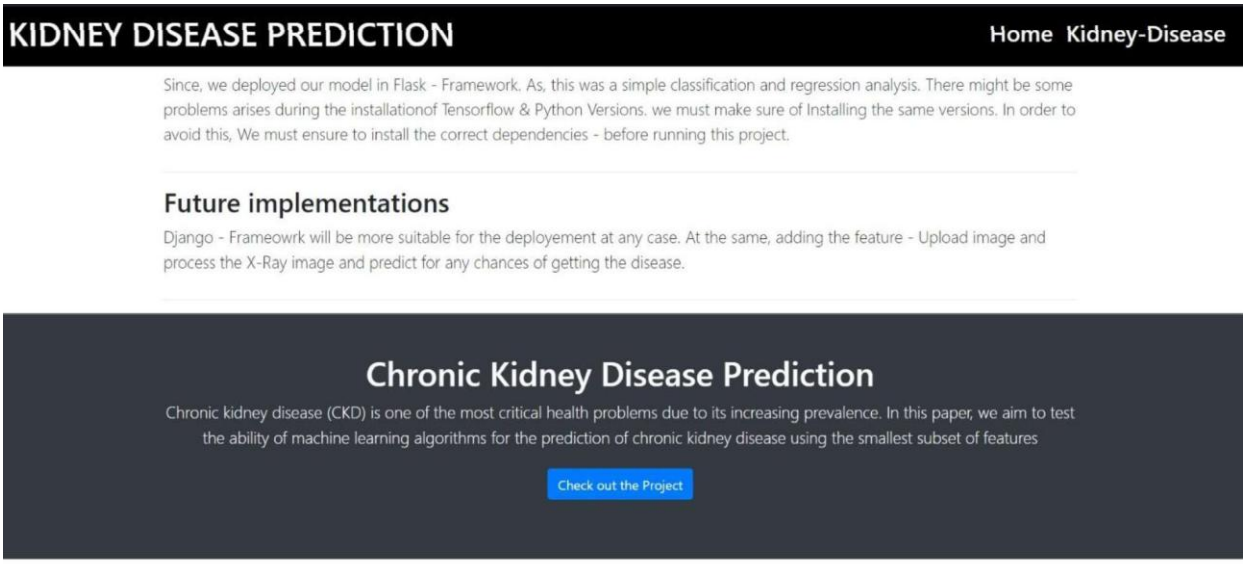
Age	Blood Pressure	Specific Gravity	Albumin	Sugar	Red Blood Cells	Pus Cell	Pus Cell Clumps	Bacteria	Blood Glucose Random	Blood Urea	Serum Creatinine	Sodium	Potassium	Haemoglobin	Packed Cell Volume	White Blood Cell Count	Red Blood Cell Count	Hypertension	Diabetes Mellitus	Coronary Artery Disease	Appetite	Pedal Edema	Anemia		
71.0	70.0	1.010	3.0	0.0	1	0	1	1	219.0	82.0	3.6	133.0	4.4	10.4	33.0	5600.0	3.6	1	1	1	0	0	0	0	0



Home.Html Page Output :-




KIDNEY DISEASE PREDICTION
Home Kidney-Disease
Data-Set
We have used the data set available in Kaggle - Chronic-Kidney Disease Prediction. After Classifying the data, Preprocessing and performed the Exploratory Data Analysis. This data set contains about 1338 records of data in various categories.
View details »
Algorithm
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning to improve the performance of the model.
View details »
Accuracy
AdaBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data. After modelling our data, the Accuracy is 98%. So far we achieved a Good accuracy.
View details »
How it will works
This prediction will be used in healthcare Applications. As it was very important to predict weather the patient was having any chances of getting this Kidney Disease. This project comprises with the deployement too. we can deploy this project y means of Python web servers available in the market.
Conflicts & Modifications
Since, we deployed our model in Flask - Framework. As, this was a simple classification and regression analysis. There might be some problems arises during the installationof Tensorflow & Python Versions. we must make sure of Installing the same versions. In order to avoid this, We must ensure to install the correct dependencies - before running this project.


KIDNEY DISEASE PREDICTION
Home Kidney-Disease
Since, we deployed our model in Flask - Framework. As, this was a simple classification and regression analysis. There might be some problems arises during the installationof Tensorflow & Python Versions. we must make sure of Installing the same versions. In order to avoid this, We must ensure to install the correct dependencies - before running this project.
Future implementations
Django - Frameowrk will be more suitable for the deployement at any case. At the same, adding the feature - Upload image and process the X-Ray image and predict for any chances of getting the disease.
Chronic Kidney Disease Prediction
Chronic kidney disease (CKD) is one of the most critical health problems due to its increasing prevalence. In this paper, we aim to test the ability of machine learning algorithms for the prediction of chronic kidney disease using the smallest subset of features
Check out the Project

CHAPTER 7

CONCLUSION AND SCOPE FOR FUTURE WORK

Conclusion

The application of Machine Learning techniques for predictive analysis is very important in the health field because it gives us the power to chronic diseases earlier and therefore save people's lives through the anticipation of cure. In this application, Gradient Boosting, Extra Tree Classifier, AdaBoost, Random Forest Classifier, Decision Tree Classifier, Gaussian Naive Bayes, KNN, Support Vector Machine to predict patients with health care data, and patients who are healthy. Simulation results showed that Gradient Boosting proved its performance in predicting with best results in terms of accuracy.

Scope For Future work Health Recommendation

In future we can use a greater number of datasets and other parameters which are affecting chronic disease. We can use deep learning approach for better result, we can add the health recommendation module as a future enhancement to the application where user can get the health recommendation based on their disease status or health status.

REFERENCES

1. Chapter 1: Definition and classification of CKD. Kidney Int Suppl (2011). 2013 Jan;3(1):19-62. [PMC free article] [PubMed]
2. Inker LA, Astor BC, Fox CH, Isakova T, Lash JP, Peralta CA, Kurella Tamura M, Feldman HI. KDOQI US commentary on the 2012 KDIGO clinical practice guideline for the evaluation and management of CKD. Am J Kidney Dis. 2014 May;63(5):713-35. [PubMed]
3. Webster AC, Nagler EV, Morton RL, Masson P. Chronic Kidney Disease. Lancet. 2017 Mar 25;389(10075):1238-1252. [PubMed]
4. Aeddula NR, Bardhan M, Baradhi KM. StatPearls [Internet]. StatPearls Publishing; Treasure Island (FL): Sep 4, 2023. Sickle Cell Nephropathy. [PubMed]
5. Textor SC. Ischemic nephropathy: where are we now? J Am Soc Nephrol. 2004 Aug;15(8):1974-82. [PubMed]
6. Kitamoto Y, Tomita M, Akamine M, Inoue T, Itoh J, Takamori H, Sato T. Differentiation of hematuria using a uniquely shaped red cell. Nephron. 1993;64(1):32-6. [PubMed]
7. Khanna R. Clinical presentation & management of glomerular diseases: hematuria, nephritic & nephrotic syndrome. Mo Med. 2011 Jan-Feb;108(1):33-6. [PMC free article] [PubMed]
8. Aeddula NR, Baradhi KM. StatPearls [Internet]. StatPearls Publishing; Treasure Island (FL): May 22, 2023. Reflux Nephropathy. [PubMed]
9. Madero M, García-Arroyo FE, Sánchez-Lozada LG. Pathophysiologic insight into MesoAmerican nephropathy. Curr Opin Nephrol Hypertens. 2017 Jul;26(4):296-302. [PubMed]
10. Coresh J, Astor BC, Greene T, Eknoyan G, Levey AS. Prevalence of chronic kidney disease and decreased kidney function in the adult US population: Third National Health and Nutrition Examination Survey. Am J Kidney Dis. 2003 Jan;41(1):1-12. [PubMed]
11. National Kidney Foundation. K/DOQI clinical practice guidelines for chronic kidney disease: evaluation, classification, and stratification. Am J Kidney Dis. 2002 Feb;39(2 Suppl 1):S1-266. [PubMed]
12. Muntner P. Longitudinal measurements of renal function. Semin Nephrol. 2009 Nov;29(6):6507. [PubMed]
13. Kshirsagar AV, Bang H, Bomback AS, Vupputuri S, Shoham DA, Kern LM, Klemmer PJ, Mazumdar M, August PA. A simple algorithm to predict incident kidney disease. Arch Intern Med. 2008 Dec 08;168(22):2466-73. [PMC free article] [PubMed]
14. Luttropp K, Lindholm B, Carrero JJ, Glorieux G, Schepers E, Vanholder R, Schalling M, Stenvinkel P, Nordfors L. Genetics/Genomics in chronic kidney disease--towards personalized medicine? Semin Dial. 2009 Jul-Aug;22(4):417-22. [PubMed]

15. Levey AS, Coresh J. Chronic kidney disease. Lancet. 2012 Jan 14;379(9811):165-80. [PubMed]
16. Hyperfiltration in remnant nephrons: a potentially adverse response to renal ablation. J Am Soc Nephrol. 2001 Jun;12(6):1315-1325. [PubMed]
17. Jamerson K, Weber MA, Bakris GL, Dahlöf B, Pitt B, Shi V, Hester A, Gupte J, Gatlin M, Velazquez EJ., ACCOMPLISH Trial Investigators. Benazepril plus amlodipine or hydrochlorothiazide for hypertension in high-risk patients. N Engl J Med. 2008 Dec 04;359(23):2417-28. [PubMed]
18. Vidt DG. Telmisartan, ramipril, or both in patients at high risk for vascular events. Curr Hypertens Rep. 2008 Oct;10(5):343-4. [PubMed]
19. Moorhead JF, Chan MK, El-Nahas M, Varghese Z. Lipid nephrotoxicity in chronic progressive glomerular and tubulo-interstitial disease. Lancet. 1982 Dec 11;2(8311):1309-11. [PubMed]
20. Johnson RJ, Nakagawa T, Jalal D, Sánchez-Lozada LG, Kang DH, Ritz E. Uric acid and chronic kidney disease: which is chasing which? Nephrol Dial Transplant. 2013 Sep;28(9):2221-8. [PMC free article] [PubMed]
21. Anderson S, Rennke HG, Brenner BM. Antihypertensive therapy must control glomerular hypertension to limit glomerular injury. J Hypertens Suppl. 1986 Dec;4(5):S242-4. [PubMed]
22. Yu HT. Progression of chronic renal failure. Arch Intern Med. 2003 Jun 23;163(12):1417-29. [PubMed]
23. Methven S, Traynor JP, Hair MD, St J O'Reilly D, Deighan CJ, MacGregor MS. Stratifying risk in chronic kidney disease: an observational study of UK guidelines for measuring total proteinuria and albuminuria. QJM. 2011 Aug;104(8):663-70. [PubMed]
24. Hallan SI, Orth SR. Smoking is a risk factor in the progression to kidney failure. Kidney Int. 2011 Sep;80(5):516-23. [PubMed]
25. de Brito-Ashurst I, Varagunam M, Raftery MJ, Yaqoob MM. Bicarbonate supplementation slows progression of CKD and improves nutritional status. J Am Soc Nephrol. 2009 Sep;20(9):2075-84. [PMC free article] [PubMed]
26. Diabetes Control and Complications Trial Research Group. Nathan DM, Genuth S, Lachin J, Cleary P, Crofford O, Davis M, Rand L, Siebert C. The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. N Engl J Med. 1993 Sep 30;329(14):977-86. [PubMed]
27. Sachdeva B, Zulfiqar H, Aeddula NR. StatPearls [Internet]. StatPearls Publishing; Treasure Island (FL): Aug 8, 2023. Peritoneal Dialysis. [PubMed]