CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Major Project Report entitled **" ANALYSIS AND PREDICTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING CLASSIFICATION APPROACHES"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Information Technology** of **Maharaja Surajmal Institute of Technology, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **March 2021 to June 2021** under the guidance of **Ms. Meena Siwach, Assistant Professor.**

The matter presented in the B. Tech Major Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Kevin Singh Bagga	**Manvendra Gupta**	**Shivam Kakkar**	**Shubham Kukreti**
01015003117	**01315003117**	**02115003117**	**02415003117**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination

Ms. Meena Siwach **Dr. Anupama Kaushik**

Assistant Professor **Head, IT**

The B. Tech Major Project Viva-Voce Examination of **Kevin Singh Bagga (01015003117), Manvendra Gupta (01315003117), Shivam Kakkar (02115003317), Shubham Kukreti (02415603317)** has been held on**……**

Dr. Tripti Sharma

(Signature of External Examiner) **Project Coordinator**

ABSTRACT

Chronic Kidney Disease (CKD) is one of the deadliest diseases that slowly damages human kidney. The disease remains undetected in its early stage and the patients can only realize the severity of the disease when it gets advanced. Hence, detecting such disease at earlier stage is a key challenge now. Machine Learning is one of the emerging fields used in the health sectors for the diagnosis of different diseases. In this paper, we compute, analyze and compare between Machine Learning classification approaches to determine which classification approach is the optimal for the prediction of CKD. K- Nearest Neighbor Classifier, Decision Tree Classifier, Gaussian NB, Random Forest classifier and Ada Boost Classifier are some renowned machine learning methods which were selected to train the model and based on these results, we can compare and determine which among the following Machine Learning Methods can predict the possibility of CKD at the most accurate level. From this comparative analysis, Logical Regression is found to be the best approach to predict CKD. Methods can predict the possibility of CKD at the most accurate level. From this comparative analysis, Logical Regression is found to be the best approach to predict CKD.

Keywords: Chronic Kidney Disease, K-nearest Neighbor, Decision tree classifier, Random Forest, Ada Boost, Gaussian NB.

AKNOWLEDGEMENT

We express our deep gratitude to **Ms. Meena Siwach**, Assistant Professor, Department of Information Technology for her valuable guidance and suggestion throughout my project work. We are thankful to **Dr. Tripti Sharma,** Project Coordinator for their valuable guidance.

We would like to extend my sincere thanks to **Dr. Anupama Kaushik, Head of Department,** for her time-to-time suggestions to complete our project work. We are also thankful to **Prof. (Dr.) Sanjay Kumar, Director** for providing us the facilities to carry out our project work.

Sign.	**Sign.**	**Sign.**	**Sign.**
Kevin Singh Bagga	**Manvendra Gupta**	**Shivam Kakkar**	**Shubham Kukreti**
01015003117	**01315003117**	**02115003117**	**02415003117**

Table of Content

List of Figures

List of Tables

CHAPTER 1

1.1 Introduction

Two bean-shaped organs, named kidney, are two important parts in human body. Kidney removes waste from blood by filtering. If this filtering system is hampered, protein can seep to urine and waste elements can remain in blood. And gradually, kidney loses its ability to filter. This failure of kidney is called chronic kidney disease (CKD), also known as Chronic Renal Disease. Whole body is affected by kidney failure. Generally, people suffer with this disease with their age, but recently from 5 years children and youth are also suffering from CKD disease. There are some symptoms which shows kidneys are beginning to fail like muscle cramps, nausea and vomiting, appetite losses, swelling in your feet and ankles, too much urine or not enough urine, trouble catching your breath, trouble sleeping, fever and vomiting. Risk factors of CKD are diabetes, smoking, lack of sleeping, hyper- tension, improper diet, etc. Among them diabetes is the more dangerous factor. At the last stage, the patient must take dialysis or do kidney transplantation. One of the best ways to reduce this death rate is early treatment. Therefore, early prediction and proper treatments can possibly stop, or slow the progression of this chronic disease.

1.2 Existing System

To predict diseases, data mining or machine learning models are playing a vital role. By making some mathematical approaches, data mining models extract patterns from data and later these patterns are used for the survival of patients. Multilayer Perceptron (MP), Support Vector Machine (SVM), K- Nearest Neighbor (KNN), Logistic Regression (LR), Naïve Bayes (NB), Random Forest (RF), etc. are some renowned machine learning methods which were successfully implemented to examine and classify the kidney disease. IN recent times, some researchers have been working on CKD by applying different computational techniques for the prediction and diagnosis of this disease.

1.3 Drawbacks

Machine learning algorithms can build complex models and make accurate decisions when given relevant data. When there is an adequate amount of data, the performance of machine learning algorithms is expected to be sufficiently satisfactory. However, in specific applications, the data are often insufficient. Therefore, it is important to analyse these algorithms and obtain good results with a relatively small sample size.

1.4 Proposed System

We are using mean, mode and median based pre-processing techniques for the missing values. Further, we have used K-Nearest Neighbor Classifier, Decision Tree Classifier, Gaussian Naïve Bayes, Logical Regression and Artificial Neural Network to train the model. Then, based on the results of each of these Machine Learning Methods, we can compare and determine which among the following methods can predict the possibility of CKD most accurately.

1.5 Plan of Implementation

The steps involved in this system implementation are:

1.5.1 Dataset selection and Data Preprocessing – Selection of the correct dataset and preprocessing the data to remove any noise, fill in empty data etc.

1.5.2 Execution of Algorithms – Implement the algorithms for the given dataset. KNN, Decision Tree, ANN, Logistic regression is used to compare the results and come up with the best method.

1.5.3 Classification of control metrics and predicting the most effective algorithm – Compare the performance of the algorithms based on the several control metrics. Plot the graphs depicting actual and predicted values and determine the most effective algorithm.

1.6 Problem Statement

From last 15 years data it has been noticed that increase in number of patient which is suffering from CKD disease and more than 60% patients are not receiving medical attention. CKD ranks number 27 and 18 in 1990 and 2010 respectively as world's prime reason of death. 956,000 people died in 2013 because of CKD. At the last stage, the patient must take dialysis or do kidney transplantation. One of the best ways to reduce this death rate is early treatment. But in developing countries, patients take treatment when they reached in serious state. An automated system can be built in order to detect CKD affected patients before reaching in last stage. Our aim is to compute, analyze and compare between Machine Learning classification approaches to determine which classification approach is the optimal for prediction of chronic kidney disease.

1.7 Objective of the Project

- CKD remains undetected in its early stage and the patients can only realize the severity of the disease when it gets advanced.
- Hence, detecting such disease at earlier stage is a key challenge now.
- This project helps to raise awareness among people and to promote early diagnosis.
- Early prediction and proper treatments can possibly stop, or slow the progression of this chronic disease.

Chapter 2: Theoretical Background

2.1 Overview on Machine Learning

Machine learning is an application of artificial intelligence (AI) that gives systems the ability to automatically learn and evolve from experience without being specially programmed by the programmer. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The main aim of machine learning is to allow computers to learn automatically and adjust their actions to improve the accuracy and usefulness of the program, without any human intervention or assistance. Traditional writing of programs for a computer can be defined as automating the procedures to be performed on input data in order to create output artifacts. Almost always, they are linear, procedural and logical. A traditional program is written in a programming language to some specification, and it has properties like:

- We know or can control the inputs to the program.
- We can specify how the program will achieve its goal.
- We can map out what decisions the program will make and under what conditions it makes them.
- Since we know the inputs as well as the expected outputs, we can be con dent that the program will achieve its goal

Traditional programming works on the premise that, as long as we can define what a program needs to do, we are confident we can define how a program can achieve that goal. This is not always the case as sometimes, however, there are problems that you can represent in a computer that you cannot write a traditional program to solve. Such problems resist a procedural and logical solution. They have properties such as:

- The scope of all possible inputs is not known beforehand.
- You cannot specify how to achieve the goal of the program, only what that goal is.
- You cannot map out all the decisions the program will need to make to achieve its goal.
- You can collect only sample input data but not all possible input data for the program.

2.1.1 Supervised and Unsupervised Learning

Machine learning techniques can be broadly categorized into the following types:

Supervised learning takes a set of feature/label pairs, called the training set. From this training set the system creates a generalized model of the relationship between the set of descriptive features and the target features in the form of a program that contains a set of rules. The objective is to use the output program produced to predict the label for a previously unseen, unlabeled input set of features, i.e., to predict the outcome for some new data. Data with known labels, which have not been included in the training set, are classified by the generated model and the results are compared to the known labels. This dataset is called the test set. The accuracy of the predictive model can then be calculated as the proportion of the correct predictions the model labeled out of the total number of instances in the test set. Unsupervised learning takes a dataset of descriptive features without labels as a training set. In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data. The goal now is to create a model that finds some hidden structure in the dataset, such as natural clusters or associations. The system does not figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data. Unsupervised learning can be used for clustering, which is used to discover any inherent grouping that are already present in the data. It can also be used for association problems, by creating rules based on the data and finding relationships or associations between them. Semi-supervised machine learning falls somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring labeled data generally does not require additional resources.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Machine learning algorithms are tools to automatically make decisions from data in order to achieve some over-arching goal or requirement.

The promise of machine learning is that it can solve complex problems automatically, faster and more accurately than a manually specified solution, and at a larger scale. Over the past few decades, many machine learning algorithms have been developed by researchers, and new ones continue to emerge and old ones modified.

2.1.2 Machine Learning Tools

There are many different software tools available to build machine learning models and to apply these models to new, unseen data. There are also a large number of well-defined machine learning algorithms available. These tools typically contain libraries implementing some of the most popular machine learning algorithms. They can be categorized as follows:

2.1.3 Pre-built application-based solutions.

2.1.4 Programming languages which have specialized libraries for machine learning Using programming languages to develop and implement models is more flexible and gave us better control of the parameters to the algorithms. It also allows us to have a better understanding of the output models produced. Some of the popular programming languages used in the field of machine learning are:

2.1.5 Python: Python is an extremely popular choice in the field of machine learning and AI development. Its short and simple syntax make it extremely easy to learn

2.1.3 SciKit-learn

SciKit learn is an open-source machine learning library built for python. Since its release in 2007, Scikit-learn has become one of the most popular open-source machine learning libraries. Scikit-learn (also called sklearn) provides algorithms for many machine learning tasks including classification, regression, dimensionality reduction and clustering. The documentation for scikit-learn is comprehensive, popular and well maintained. Sklearn is built on mature Python Libraries such as NumPy, SciPy, and matplotlib. While languages such as R and MATLAB are extremely popular and useful for machine learning, we decided to choose Python along with its SciKit-learn libraries as our programming language of choice. The reasons for this are:

- We already have some familiarity and exposure to Python, and thus have a smaller learning curve.
- Both Python and Scikit-learn have excellent documentation and tutorials available online
- The number of classic machine learning algorithms that come with Scikit-learn, and the consistent patterns for using the different models i.e., each model can be used with the same basic commands for setting up the data, training the model and using the model for prediction. This makes it easier to try a range of machine learning algorithms on the same data.
- The machine learning algorithms included with sklearn have modifiable parameters known as hyper-parameters that effect the performance of the model. These usually have sensible default values, so that we can run them without needing a detailed knowledge or understanding of their semantics.

- The Python notebook, which is an interactive computational environment for Python, in which a user can combine code execution, rich text, mathematics and plots in a web page. This functionality allows us to provide the notebooks we used to run our experiments almost as an audit and in a presentable.

2.2 Dataset

For the system, the dataset was collected from a variety of source but the main source was UCI Repository. It contains 400 samples of patients having 24 attributes

Datasets include: -

Attribute	**Representation**	**Information Attribute**	**Description**
1	Age	Age	Numerical
2	Blood Pressure	Bp	Numerical
3	Specific Gravity	Sg	Nominal
4	Albumin	Al	Nominal
5	Sugar	Su	Nominal
6	Red Blood Cell	Rbc	Nominal
7	Pus Cell	Pc	Nominal
8	PusCellClumps	Pcc	Nominal
9	Bacteria	Ba	Nominal
10	Blood Glucose Random	Bgr	Numerical
11	Blood Urea	Bu	Numerical
12	Serum Creatinium	Sc	Numerical
13	Sodium	So	Numerical
14	Potassium	Pot	Numerical
15	Haemoglobin	hemo	Numerical
16	Packed Cell Volume	Pcv	Numerical
17	White Blood cell Count	Wc	Numerical
18	Red Blood Cell count	Rc	Numerical
19	Hypertension	Htn	Nominal
20	Diabetes Mellitus	Dm	Nominal
21	Coronary Artery Disease	Cad	Nominal
22	Appetite	appet	Nominal
23	Pedal Edema	Pe	Nominal
24	Anemia	Ane	Nominal
25	Class	Class	Nominal

Table 2.1: List of data set

2.3 Data Preprocessing

The data preprocessing involves cleaning, extracting, filling missing values and transformation to suitable formats. The cleaning is done through some standard tools available like WEKA. Table 2 show how the missing values are filled. We are using the label encoder and min-max algorithm to convert the nominal value to numeric value and bring all the attributes to the same scale respectively for better training.

Type of Attribute	Filled missing value
Numeric	Average
Nominal	Mode

Table 2.2: Table 2 (Missing values)

2.4 Machine Learning Algorithms

Machine Learning algorithms used in the recommendation system are:

1. **Ada Boost Classifier:** It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Ada Boost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set.

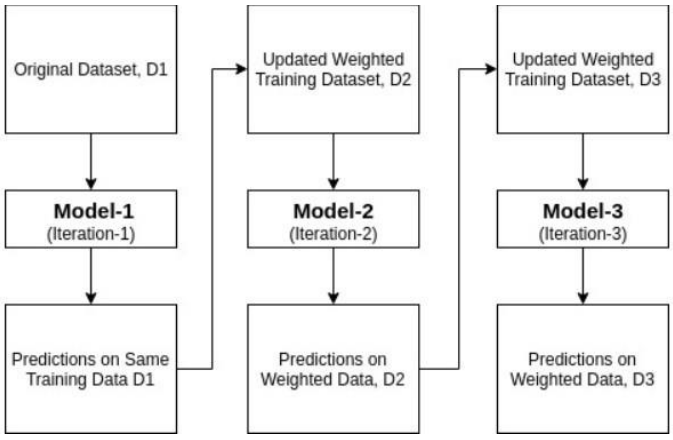


Figure 2.1: Ada Boost Classifier

2. **Random Forest**: Random Forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.
3. **Decision Tree**: Regression and classification problems. The general objective of using Decision Tree is to create a model that predicts classes or values of target variables by generating decision rules derived from training data sets. Decision tree algorithm follows a tree structure with roots, branches and leaves. The attributes of decision making are the internal nodes and class labels are represented as leaf nodes. Decision Tree algorithm is easy to understand compared with other classification algorithms.

4. **KNN (K-nearest Neighbor)**: In pattern recognition, the k-nearest neighbors' algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

   In K-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. K-NN is a type of instance based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation
5. **Gaussian NB**: In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.[1] But they could be coupled with Kernel density estimation and achieve higher accuracy levels. When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

Chapter 3: System Requirement Specification

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives the brief description of the services that the system should provide and also the constraints under which, the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

3.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements: -

1. All the data must be in the same format as a structured data.

2. The data collected will be vectorized and sent across to the classifier

3.2 Non-Functional Requirement

Nonfunctional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Nonfunctional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as: -

3.2.1 Product Requirements

3.2.2 Organizational Requirements

3.2.3 User Requirements

3.2.4 Basic Operational Requirements

3.3 Product Requirements

Platform Independency: Standalone executables for embedded systems can be created so the algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

Correctness: It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: Model Coder provides an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into many modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevancy and correctness Nonfunctional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

3.4 Organizational Requirements

Process Standards: The standards defined by DRDO are used to develop the application which is the standard used by the developers inside the defense organization.

Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

3.5 Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points: -

Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

3.5 System Configuration

Hardware System Configuration:

3.1 Processor: 2 gigahertz (GHz) or faster processor or SoC.

RAM: 8 gigabytes (GB) for 32-bit or 8 GB for 64-bit.

Hard disk space: =16GB. Software Configuration:

Operating System: Windows XP/7/8/8.1/10, Linux and Mac

Coding Language: Python

3.6 Tools Used:

1. Pandas
2. Numpy
3. Tensorflow
4. Keras
5. Sickitlearn

Chapter 4: System Analysis

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

4.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately, this condition does not prevail in practical world. So, it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

4.1.1 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Since the project is Machine learning based, the cost spent in executing this project would not demand cost for software and related products, as most of the products are open source and free to use. Hence the project would consume minimal cost and is economically feasible.

4.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Since machine learning algorithms is based on pure math there is very less requirement for any professional software. Also, most of the tools are open source. The best part is that we can run this software in any system without any software requirements which makes them highly portable. Most of the documentation and tutorials make easy to learn thetechnology.

4.1.3 Social Feasibility

The aspect of 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 main purpose of this project which is based on creating an early prediction system of chronic kidney diseases using basic blood test reports. Our aim is help patients get early treatment based on these predictions which would save many lives. Thus, this is a noble cause for the sake of the society, a small step taken to achieve a secure and healthy future.

4.2 Analysis

4.2.1 Performance Analysis

Most of the software we use is open source and free. The models which we use in this software only are trained once in the beginning and there is no need to fed again in for the training phase. One can directly predict for values. Hence time-complexity is very less. Therefore, this model is temporally sound.

4.2.2 Economic Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. The completion of this project can be considered free of cost in its entirety. As the software used in building the model is free of cost and all the data sets used are being downloaded from Kaggle and Govt. of India website.

4.2.3 Technical Analysis

As mentioned earlier, the tools used in building this software is open source. Each tool contains simple methods and the required methods are overridden to tackle the problem.

Chapter 5: System Design

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customer's requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical systemdesign.

5.1 System development methodology

System Development methodology is the development of a system or method for a unique situation. Having a proper methodology helps us in bridging the gap between the problem statement and turning it into a feasible solution. It is usually marked by converting the System Requirements Specifications (SRS) into a real-world solution. System design takes the following inputs:

- Statement of work.
- Requirement determination plan.
- Current situation analysis.
- Proposed system requirements including a conceptual data model and metadata (data about data).

5.2 Modal Phases

The waterfall model is a sequential software development process, in which progress is seen as owing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

• **Requirement Analysis**: This phase is concerned about collection of requirements of the system. This process involves generating document and requirement review.

• **System Design**: Keeping the requirements in mind the system specifications are translated into a software representation. In this phase the designer emphasizes on: - algorithm, data structure, software architecture etc.

• **Coding**: In this phase programmer starts his coding in order to give a full sketch of product. In other words, system specifications are only converted into machine.

• **Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation.

• **Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

•**Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

5.1.1 Advantages of the Waterfall Model

1. Clear project objectives.
2. Stable project requirements.
3. Progress of system is measurable.
4. Strict sign-off requirements.
5. Helps you to be perfect.
6. Logic of software development is clearly understood.
7. Production of a formal specification
8. Better resource allocation.

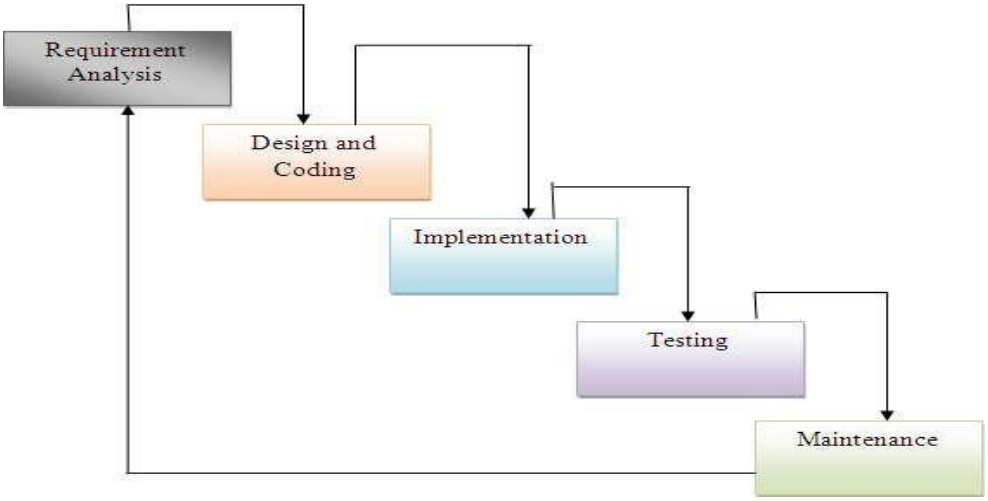


Fig 5.1: Waterfall model

5.3 System Architecture

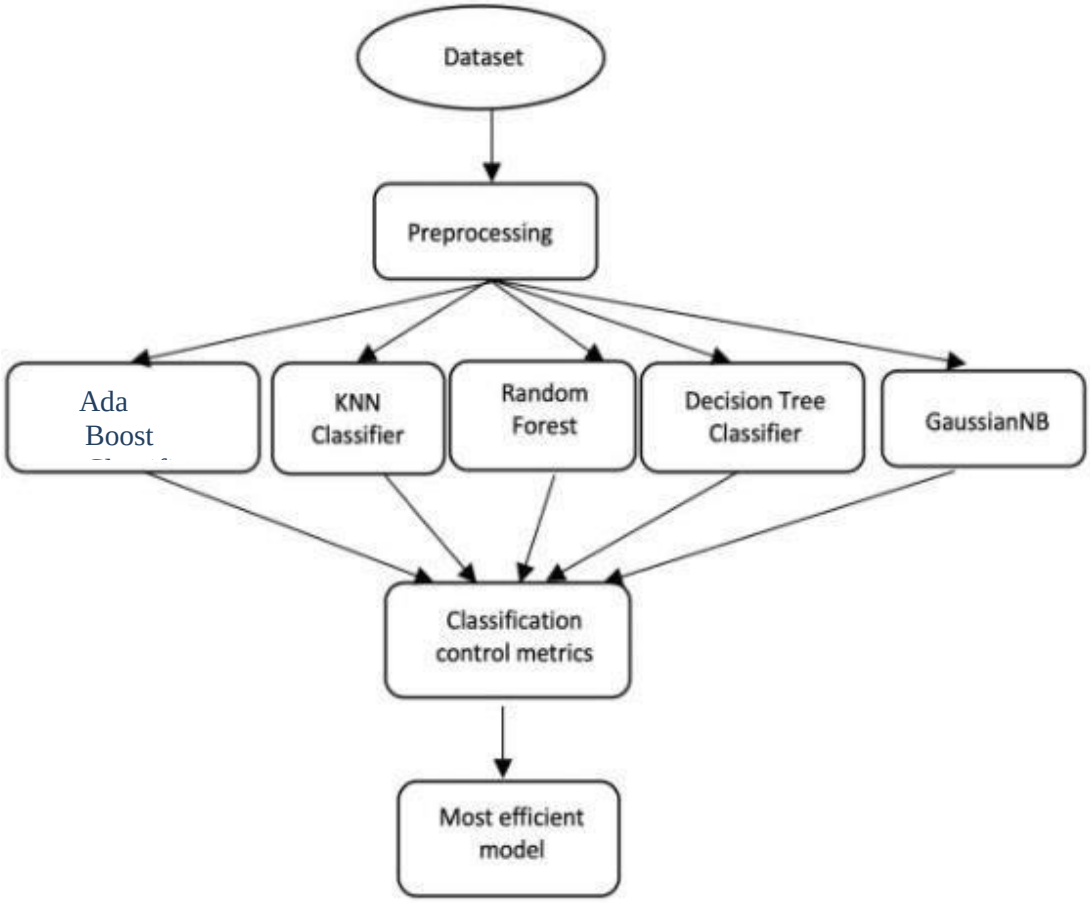


Figure 5.2 System Architecture

A system architecture is a conceptual model using which we can define the structure and behavior of that system. It is a formal representation of a system. Depending on the context, system architecture can be used to refer to either a model to describe the system or a method used to build the system. Building a proper system architecture helps in analysis of the project, especially in the early stages. Figure 6.2 depicts the system architecture and is explained in the following section.

5.4 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

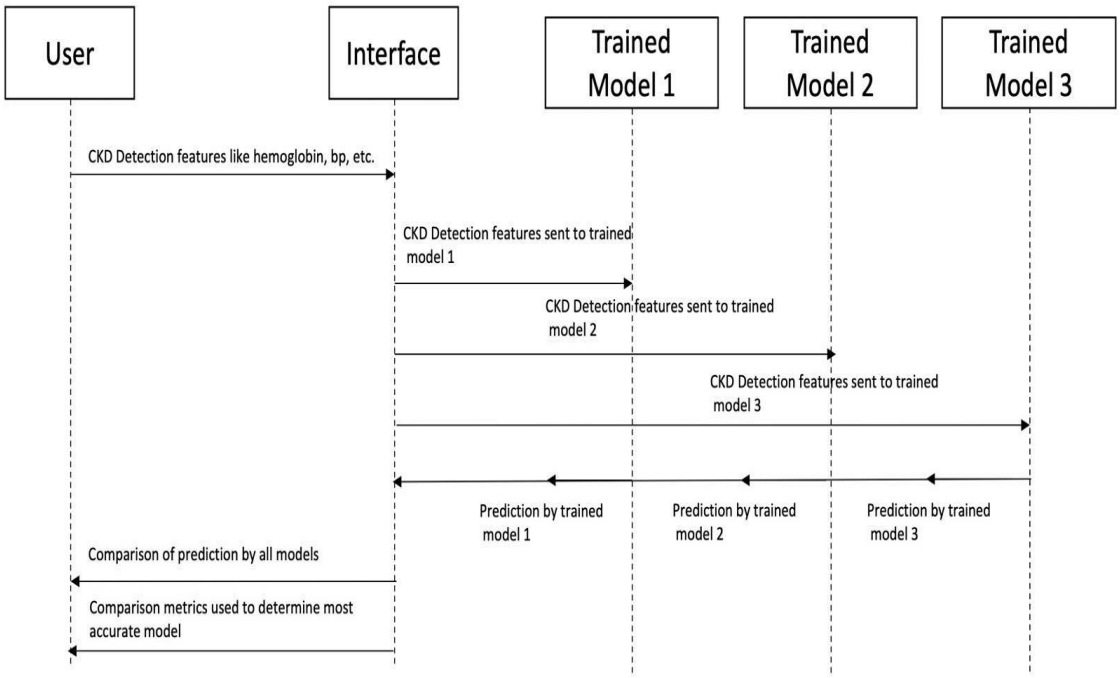


Figure 5.3 Sequence Diagram

Chapter 6: Implementation

6.1 Data Analysis

One of the first steps we perform during implementation is an analysis of the data. This was done by us in an attempt to find the presence of any relationships between the various attributes present in the dataset.

Acquisition of Training Dataset: The Training data set was acquired from the UCI Repository for kidney disease. The dataset was collected from a number of hospitals from Tamil Nadu and the values are actual test results values that were obtained. We have a total of 24 attributes which make up the dataset but on pre-processing it was found that only 6 of the 24 are important in determining that relationship. There are a total of 400 samples. This way the predictions can be used to correctly determine early detection of chronic kidney disease.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
		age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	rbc	pc	pcc	ba	ane
0		48	80	1.02	1	0	121	36	1.2			15.4	44	7800	5.2	yes	yes	no	good	no		normal	notpreser	notpreser	no
1		7	50	1.02	4	0		18	0.8			11.3	38	6000		no	no	no	good	no		normal	notpreser	notpreser	no
2		62	80	1.01	2	3	423	53	1.8			9.6	31	7500		no	yes	no	poor	no	normal	normal	notpreser	notpreser	yes
3		48	70	1.005	4	0	117	56	3.8	111	2.5	11.2	32	6700	3.9	yes	no	no	poor	yes	normal	abnormal	present	notpreser	yes
4		51	80	1.01	2	0	106	26	1.4			11.6	35	7300	4.6	no	no	no	good	no	normal	normal	notpreser	notpreser	no
5		60	90	1.015	3	0	74	25	1.1	142	3.2	12.2	39	7800	4.4	yes	yes	no	good	yes			notpreser	notpreser	no
6		68	70	1.01	0	0	100	54	24	104	4	12.4	36			no	no	no	good	no		normal	notpreser	notpreser	no
7		24		1.015	2	4	410	31	1.1			12.4	44	6900	5	no	yes	no	good	yes	normal	abnormal	notpreser	notpreser	no
8		52	100	1.015	3	0	138	60	1.9			10.8	33	9600	4	yes	yes	no	good	no	normal	abnormal	present	notpreser	yes
9		53	90	1.02	2	0	70	107	7.2	114	3.7	9.5	29	12100	3.7	yes	yes	no	poor	no	abnormal	abnormal	present	notpreser	yes
10		50	60	1.01	2	4	490	55	4			9.4	28			yes	yes	no	good	no		abnormal	present	notpreser	yes
11		63	70	1.01	3	0	380	60	2.7	131	4.2	10.8	32	4500	3.8	yes	yes	no	poor	yes	abnormal	abnormal	present	notpreser	no
12		68	70	1.015	3	1	208	72	2.1	138	5.8	9.7	28	12200	3.4	yes	yes	yes	poor	yes		normal	present	notpreser	no
13		68	70				98	86	4.6	135	3.4	9.8				yes	yes	yes	poor	yes			notpreser	notpreser	no
14		68	80	1.01	3	2	157	90	4.1	130	6.4	5.6	16	11000	2.6	yes	yes	yes	poor	yes	normal	abnormal	present	present	no
15		40	80	1.015	3	0	76	162	9.6	141	4.9	7.6	24	3800	2.8	yes	no	no	good	no		normal	notpreser	notpreser	yes
16		47	70	1.015	2	0	99	46	2.2	138	4.1	12.6				no	no	no	good	no		normal	notpreser	notpreser	no
17		47	80				114	87	5.2	139	3.7	12.1				yes	no	no	poor	no			notpreser	notpreser	no
18		60	100	1.025	0	3	263	27	1.3	135	4.3	12.7	37	11400	4.3	yes	yes	yes	good	no		normal	notpreser	notpreser	no
19		62	60	1.015	1	0	100	31	1.6			10.3	30	5300	3.7	yes	no	yes	good	no		abnormal	present	notpreser	no
20		61	80	1.015	2	0	173	148	3.9	135	5.2	7.7	24	9200	3.2	yes	yes	yes	poor	yes	abnormal	abnormal	notpreser	notpreser	yes
21		60	90					180	76	4.5		10.9	32	6200	3.6	yes	yes	yes	good	no			notpreser	notpreser	no
22		48	80	1.025	4	0	95	163	7.7	136	3.8	9.8	32	6900	3.4	yes	no	no	good	no	normal	abnormal	notpreser	notpreser	yes
23		21	70	1.01	0	0										no	no	no	poor	no		normal	notpreser	notpreser	yes
24		42	100	1.015	4	0		50	1.4	129	4	11.1	39	8300	4.6	yes	no	no	poor	no	normal	abnormal	notpreser	present	no
25		61	60	1.025	0	0	108	75	1.9	141	5.2	9.9	29	8400	3.7	yes	yes	no	good	no		normal	notpreser	notpreser	yes
26		75	80	1.015	0	0	156	45	2.4	140	3.4	11.6	35	10300	4	yes	yes	no	poor	no		normal	notpreser	notpreser	no
27		69	70	1.01	3	4	264	87	2.7	130	4	12.5	37	9600	4.1	yes	yes	yes	good	yes	normal	abnormal	notpreser	notpreser	no
28		75	70		1	3	123	31	1.4							no	yes	no	good	no			notpreser	notpreser	no
29		68	70	1.005	1	0		28	1.4			12.9	38			no	no	yes	good	no	abnormal	abnormal	present	notpreser	no
30			70				93	155	7.3	132	4.9					yes	yes	no	good	no			notpreser	notpreser	no
31		73	90	1.015	3	0	107	33	1.5	141	4.6	10.1	30	7800	4	no	no	no	poor	no		abnormal	present	notpreser	no
32		61	90	1.01	1	1	159	39	1.5	133	4.9	11.3	34	9600	4	yes	yes	no	poor	no		normal	notpreser	notpreser	no
33		60	100	1.02	2	0	140	55	2.5			10.1	29			yes	no	no	poor	no	abnormal	abnormal	notpreser	notpreser	no
34		70	70	1.01	1	0	171	153	5.2							no	yes	no	poor	no	normal		present	present	no

Figure 6.1: Dataset

D	E	F	G	H	I
sg	dm	htn	hemo	pcv	al
1.02	yes	yes	15.4	44	
1.02	no	no	11.3	38	
1.01	yes	no	9.6	31	
1.005	no	yes	11.2	32	
1.01	no	no	11.6	35	
1.015	yes	yes	12.2	39	
1.01	no	no	12.4	36	
1.015	yes	no	12.4	44	
1.015	yes	yes	10.8	33	
1.02	yes	yes	9.5	29	
1.01	yes	yes	9.4	28	
1.01	yes	yes	10.8	32	
1.015	yes	yes	9.7	28	
	yes	yes	9.8		
1.01	yes	yes	5.6	16	
1.015	no	yes	7.6	24	
1.015	no	no	12.6		
	no	yes	12.1		
1.025	yes	yes	12.7	37	
1.015	no	yes	10.3	30	
1.015	yes	yes	7.7	24	
	yes	yes	10.9	32	
1.025	no	yes	9.8	32	
1.01	no	no			

Figure 6.2: Model Training dataset

6.2 Data Preprocessing

After analyzing and visualizing the data, the next step is preprocessing. Data preprocessing is an important step as it helps in cleaning the data and making it suitable for use in machine learning algorithms. Most of the focus in preprocessing is to remove any outliers or erroneous data, as well as handling any missing values. Missing data can be dealt with in two ways. The first method is to simply remove the entire row which contains the missing or erroneous value. While this an easy to execute method, it is better to use only on large datasets. Using this method on small datasets can reduce the dataset size too much, especially if there are a lot of missing values. This can severely affect the accuracy of the result. Since ours is a relatively small dataset, we will not be using this method. Instead, we would be filling our missing values with average or mode of the column based on the type of attribute. If the attribute is nominal than we will use the average and if it is non-nominal, we would be using the mode. The dataset that we used had values that were in string format so we had to transform and encode the into integer valued so as to pass as an input to the neural network. First, we converted the data into pandas categorical data and create a separated data frame. The steps for the above are illustrated below.

```
#cleaning of data for garbage values
df = df.replace("?", np.nan)
df = df.replace("\t?", np.nan)
df['htn'] = df['htn'].replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes', '':np.nan})
df[['appet']] = df[['appet']].replace(to_replace={'no':np.nan})
df['dm'] = df['dm'].replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes', '':np.nan})
df['classification'] = df['classification'].replace(to_replace={'ckd\t':'ckd'})
df['cad'] = df['cad'].replace(to_replace='\tno',value='no')

#using techniques like mean and mode to fill the missing values
for column in df.columns:
    if df[column].dtype == np.number:
        #filled the missing values of numeric dataset with the mean of the column
        df[column].fillna(df[column].mean(), inplace=True)
    else:
        #filled the missing values of string dataset with the mode of the column
        df[column].fillna(df[column].value_counts().idxmax(), inplace=True)
        #converting the string values into categorical values for evaluation through different models
        if(column == "dm"):
            df[column] = LabelEncoder().fit_transform(df[column])
        else:
            df[column] = LabelEncoder().fit_transform(df[column])
#renaming column classification to class
df.rename(columns = {'classification':'class'},inplace = True)
#removing the column id as it has no actual use in my project and is an extra column
df.drop('id',axis=1,inplace=True)
print(df.head())
#furter cleaning of dataframe if any more NAN values are left
df = df.dropna(axis=0)
#gives the count of uniques values in the column class
df['htn'].value_counts()
```

Figure 6.3: Steps done for data pre-processing

```
#these lines of code are used for feature scaling using normalization. Faaiz will work on other similar methods and come up with
x_scaler = MinMaxScaler()
x_scaler.fit(X)
column_names = X.columns
X[column_names] = x_scaler.transform(X)
print(X.head())
```

Figure 6.4: Converting string values into categorical variable

As we all know out of all the 24 attributes not all will directly affect the result so in order to find the level of dependence, we used a tree method called Extra tree Classifier to find the level of dependency and came to the conclusion that only 6 attributes namely Al, Sg, HTN, pcv, hemo, dm are the ones which actually affect the final result the most.

```
#the below code is used to find the best attributes out of the 24.
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
#THIS LINE I HAVE STORED THE TOP 18 VALUES INTO x SO WHENEVR YOU WANT TO CHANGE
x = feat_importances.nlargest(6).keys()
feat_importances.nlargest(25).plot(kind='barh')
plt.show()
print (x)
```

```
 0.0050224  0.04236518 0.12539235 0.10879565 0.00383986 0.04859132
 0.02994217 0.01691402 0.03311996 0.00636771 0.00260624 0.01370255]
```

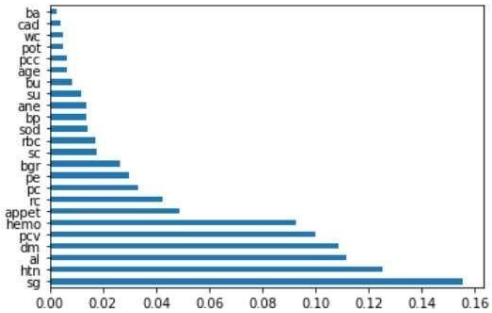


Figure 6.5: Finding the best attributes using Extra Tree Classifier

6.3 Machine Learning Models

6.3.1 **Decision Tree:** Regression and classification problems. The general objective of using Decision Tree is to create a model that predicts classes or values of target variables by generating decision rules derived from training data sets. Decision tree algorithm follows a tree structure with roots, branches and leaves. The attributes of decision making are the internal nodes and class labels are represented as leaf nodes. Decision Tree algorithm is easy to understand compared with other classification algorithms.

6.3.2 **KNN (K-nearest Neighbor):** In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In K-NN regression, the output is the property value for the object. This value is the

average of the values of k nearest neighbors. K-NN is a type of instance based learning, or lazy learning, where the function is only approximated locally and all computation is deferred untilfunction evaluation

6.3.3 **Gaussian NB:** In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.[1] But they could be coupled with Kernel density estimation and achieve higher accuracy levels. When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

6.3.4 **Random Forest:** Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

```
classifiers = [
    KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2),
    #SVC(C=.1, degree=1, kernel='poly', probability=True),
    #NuSVC(nu=.1, degree=1, kernel='poly', probability=True),
    DecisionTreeClassifier(),
    #AdaBoostClassifier(),
    #GradientBoostingClassifier(),
    GaussianNB(),
    #LinearDiscriminantAnalysis(),
    LogisticRegression()]

# Logging for Visual Comparison
log_cols = ["Classifier", "Accuracy", "f1score", "specificity", "sensitivity", 'Log Loss']
log = pd.DataFrame(columns=log_cols)

for clf in classifiers:
    clf.fit(X_train, y_train)
    name = clf.__class__.__name__

    print("="*30)
    print(name)
    try:
        print('****Results****')
        train_predictions = clf.predict(X_test)
        acc = accuracy_score(y_test, train_predictions)
        print("Accuracy: {:.4%}".format(acc))
        pre_score = precision_score(y_test, train_predictions)
        print("precision score: \n {:.4}".format(pre_score))
        f1score = f1_score(y_test, train_predictions)
        print("f1_score: \n {:.4}".format(f1score))
        # print("classification_report: \n",classification_report(y_test, train_predictions))
        conmat = confusion_matrix(y_test, train_predictions)
        print("confusion_matrix: \n",conmat)
        tn, fp, fn, tp = conmat.ravel()
        specificity = (tn)/(tn + fp)
        print("Specificity: \n {:.4}".format(specificity))
        sensitivity = (tp)/(tp + fn)
        print("Sensitivity: \n {:.4}".format(sensitivity))
        print("log_loss: \n",log_loss(y_test, train_predictions))
        log_entry = pd.DataFrame([[name, acc*100,f1score, specificity, sensitivity, log_loss(y_test, train_predictions)]], colum
        log = log.append(log_entry)
    except Exception as e:
        print (e)

print("="*30)
```

Figure 6.6: Different models used like decision tree, Random Forest etc.

6.3.5 Ada Boost-It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Ada Boost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set.

```
from keras.models import Sequential  #to initialize the neural network
from keras.layers import Dense  # to build the layers of ANN
from keras.layers import Dropout
from keras.optimizers import SGD
```

```
Using TensorFlow backend.
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Abhimanyu\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)ty
pe'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

```
f = int((len(X.columns)+1)/2)
import warnings
import keras as k
warnings.filterwarnings('ignore')
# Initialising the ANN
classifier = Sequential()
# Adding the input layer and the first hidden layer
classifier.add(Dense(units = f,  kernel_initializer = 'uniform', activation = 'relu', input_dim = len(X.columns)))
#classifier.add(Dropout(0.2))
# Adding the second hidden layer
classifier.add(Dense(units = f, kernel_initializer ='uniform', activation = 'relu'))
classifier.add(Dropout(0.2))
# Adding the output layer
classifier.add(Dense(units=1, kernel_initializer = 'uniform', activation = 'hard_sigmoid'))
# Compiling the ANN
#sgd = SGD(lr=0.0001, decay=0.01, momentum=0.9, nesterov=True)
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
# Fitting the ANN to the Training set
model = classifier.fit(X_train, y_train, validation_data = (X_test, y_test), batch_size = 5, epochs = 200)
```

Figure 6.7: Code snippet for Ada Boost

Chapter 7: TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve: -

1. To affirm the quality of the project.
2. To find and eliminate any residual errors from previous stages.
3. To validate the software as a solution to the original problem.
4. To provide operational reliability of the system.

7.1 Testing Methodologies

The program comprises of several algorithms which are tested individually for the accuracy. we check for the correctness of the program as a whole and how it performs.

7.2 Unit testing

Unit tests focus on ensuring that the correct changes to the world- state take place when a transaction is processed. The business logic in transaction processor functions should have unit tests, ideally with 100 percent code coverage. This will ensure that you do not have ty- pos or logic errors in the business logic. The various modules can be individually run from a command line and tested for correctness. The tester can pass various values, to check the answer returned and verify it with the values given to him/her. The other work around is to write a script and run all the tests using it and write the output to a log file and using that to verify the results. We tested each of the algorithms individually and made changes in preprocessing accordingly to increase the accuracy.

7.3 System Testing

Black System Testing is a level of software testing where a complete and in- tegrated software is tested. The purpose of this test is to evaluate the systems compliance with the specified requirements. System Testing is the testing of a complete and fully integrated software product. and White Box Testing. System test falls under the black box testing cate- gory of software testing. Different Types of System Testing:

- **Usability Testing** - Usability Testing mainly focuses on the users ease to use the application, flexibility in handling controls and ability of the system to meet its objectives.
- **Load Testing** - Load Testing is necessary to know that a software solution will perform under real-life loads.
- **Regression Testing**- Regression Testing involves testing done to make sure none of the changes made over the course of the development process have caused new bugs.
- **Recovery Testing** - Recovery testing is done to demonstrate a soft- ware solution is reliable, trustworthy and can successfully recoup from possible crashes.
- **Migration Testing** - Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

7.4 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process**.** Software quality assurance encompasses: -

- Analysis, design, coding and testing methods and tools
- Formal technical reviews that are applied during each software engineering

- Multi-tiered testing strategy
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanisms.

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factors that can be directly measured.
- Factors that can be indirectly measured
- These factors focus on three important aspects of a software product
- Its operational characteristics
- Its ability to undergo changes
- Its adaptability to a new environment.

7.5 Functional Testing

Functional Testing is also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: Interfacing systems or procedures must be invoked.

Chapter 8: Result and Performance Analysis

For the purposes of this project, we have used five popular algorithms: Logistic regression and Neural network, Decision Tree, Gaussian NB, KNN. All the algorithms are based on supervised learning. We are determining the best method considering 4 factors namely Specificity, Sensitivity, Log Loss, Accuracy. When plotted on a graph for all the algorithms it was found that Logistic Regression was the best method to use to find chronic kidney disease.

8.1 Accuracy

Accuracy is the number of correctly predicted data points out of all the data points. More formally, it is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives.

Accuracy = $(TP + TN)/(TP + TN + FP + FN)$

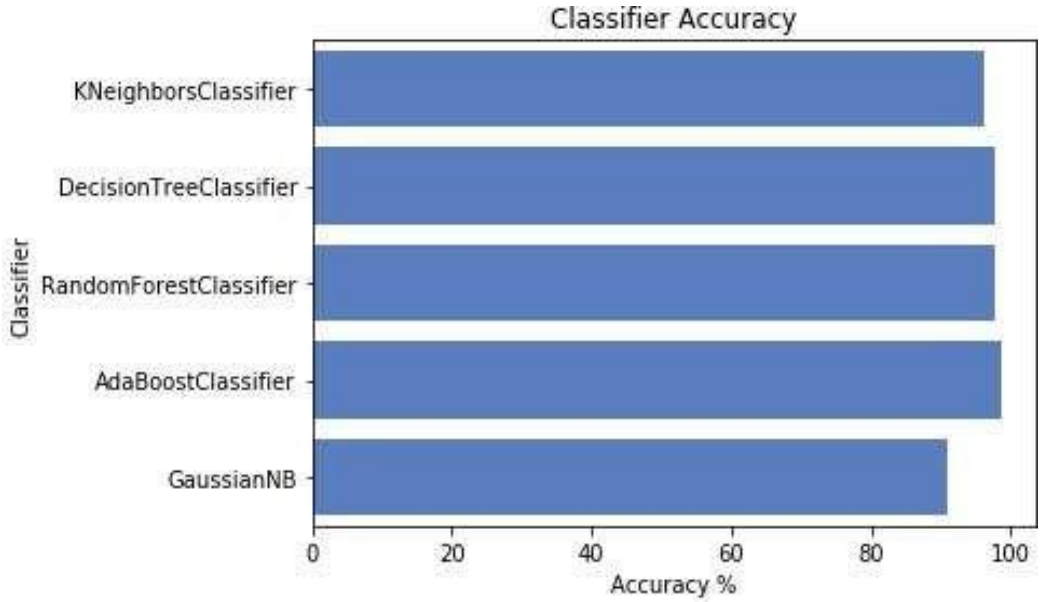


Figure 8.1: Accuracy comparison

8.2 Specificity

Specificity is defined as the proportion of actual negative, which got predicted as the negative (or true negative).

Specificity = $TN/\ (TN + FP)$

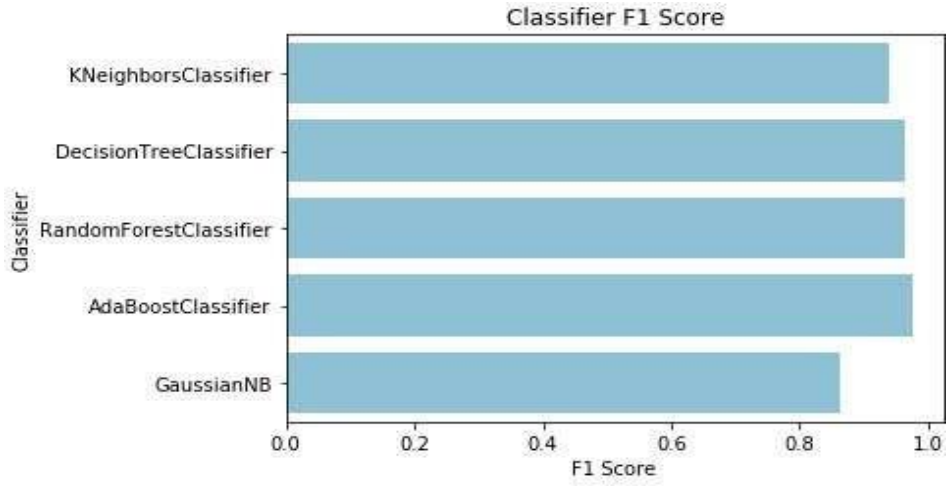


Figure 8.2: Specificity Comparison

8.3 Sensitivity

Sensitivity is a measure of the proportion of actual positive cases that got predicted as positive (or true positive).

Sensitivity = $TP\ /(TP + FN)$

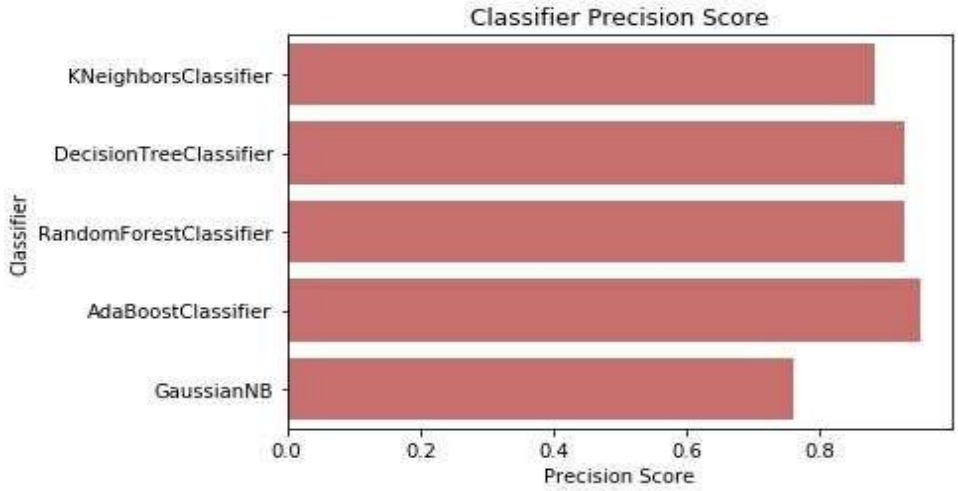


Figure 8.3: Sensitivity Comparison

8.4 Log Loss

Logarithmic loss measures the performance of a classification model where the prediction input is a probability value between 0 and 1. The goal of our machine learning models is to minimize thisvalue. Log Loss = -{(y\log(p) + (1 - y) \log (1 - p))}

- y - a binary indicator (0 or 1) of whether class label c is the correct classification for observation
- p - the model's predicted probability that observation o is of class c

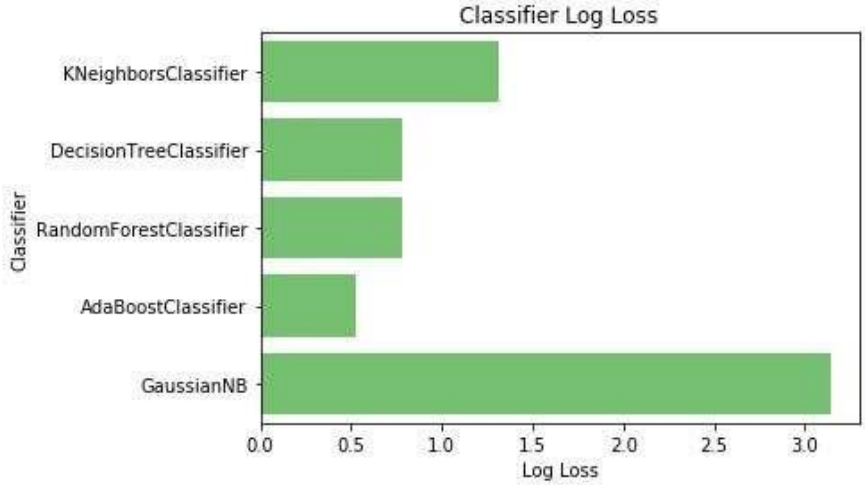


Figure 8.4: Log Loss Comparison

From the above comparison metrics, it can be easily deducted that Logistic regression is the best method to Achieve high prediction capabilities for Detecting Chronic Kidney Disease.

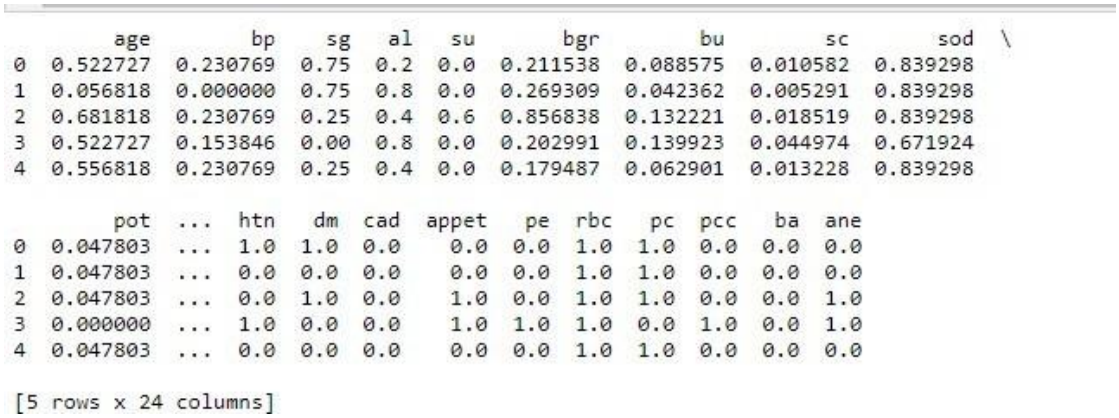
8.5 SNAPSHOTS OF OUTPUT

	id	age	bp	sg	al	su	bgr	bu	sc	sod	...	dm	cad	\
0	0	48.0	80.0	1.020	1.0	0.0	121.0	36.0	1.2	NaN	...	yes	no	
1	1	7.0	50.0	1.020	4.0	0.0	NaN	18.0	0.8	NaN	...	no	no	
2	2	62.0	80.0	1.010	2.0	3.0	423.0	53.0	1.8	NaN	...	yes	no	
3	3	48.0	70.0	1.005	4.0	0.0	117.0	56.0	3.8	111.0	...	no	no	
4	4	51.0	80.0	1.010	2.0	0.0	106.0	26.0	1.4	NaN	...	no	no	
..	...	...	...	...	...	...	...	...	...	...	...	...	...	
395	395	55.0	80.0	1.020	0.0	0.0	140.0	49.0	0.5	150.0	...	no	no	
396	396	42.0	70.0	1.025	0.0	0.0	75.0	31.0	1.2	141.0	...	no	no	
397	397	12.0	80.0	1.020	0.0	0.0	100.0	26.0	0.6	137.0	...	no	no	
398	398	17.0	60.0	1.025	0.0	0.0	114.0	50.0	1.0	135.0	...	no	no	
399	399	58.0	80.0	1.025	0.0	0.0	131.0	18.0	1.1	141.0	...	no	no	

	appet	pe	rbc	pc	pcc	ba	ane	classification
0	good	no	NaN	normal	notpresent	notpresent	no	ckd
1	good	no	NaN	normal	notpresent	notpresent	no	ckd
2	poor	no	normal	normal	notpresent	notpresent	yes	ckd
3	poor	yes	normal	abnormal	present	notpresent	yes	ckd
4	good	no	normal	normal	notpresent	notpresent	no	ckd
..	...	...	...	...	...	...	...	...
395	good	no	normal	normal	notpresent	notpresent	no	notckd
396	good	no	normal	normal	notpresent	notpresent	no	notckd
397	good	no	normal	normal	notpresent	notpresent	no	notckd
398	good	no	normal	normal	notpresent	notpresent	no	notckd
399	good	no	normal	normal	notpresent	notpresent	no	notckd

[400 rows x 26 columns]

Figure 8.5: Dataset before pre-processing



	age	bp	sg	al	su	bgr	bu	sc	sod	\
0	0.522727	0.230769	0.75	0.2	0.0	0.211538	0.088575	0.010582	0.839298	
1	0.056818	0.000000	0.75	0.8	0.0	0.269309	0.042362	0.005291	0.839298	
2	0.681818	0.230769	0.25	0.4	0.6	0.856838	0.132221	0.018519	0.839298	
3	0.522727	0.153846	0.00	0.8	0.0	0.202991	0.139923	0.044974	0.671924	
4	0.556818	0.230769	0.25	0.4	0.0	0.179487	0.062901	0.013228	0.839298	

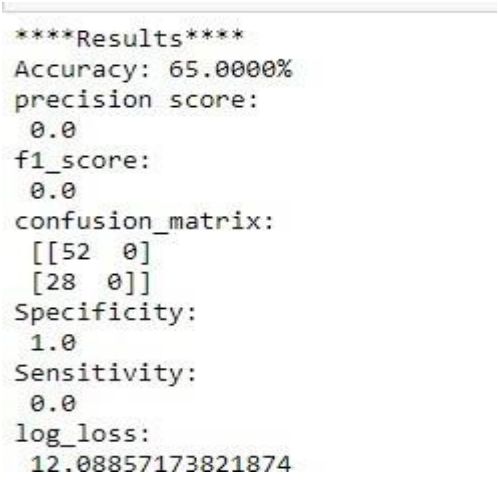
	pot	...	htn	dm	cad	appet	pe	rbc	pc	pcc	ba	ane
0	0.047803	...	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
1	0.047803	...	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
2	0.047803	...	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0
3	0.000000	...	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0
4	0.047803	...	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0

[5 rows x 24 columns]

Figure 8.6: Data set after pre-processing

```
Train on 320 samples, validate on 80 samples
Epoch 1/200
320/320 [==============================] - 1s 4ms/step - loss: 0.6917 - accuracy: 0.6156 - val_loss: 0.6892 - val_accuracy:
0.6500
Epoch 2/200
320/320 [==============================] - 0s 281us/step - loss: 0.6890 - accuracy: 0.6187 - val_loss: 0.6862 - val_accuracy:
0.6500
Epoch 3/200
320/320 [==============================] - 0s 259us/step - loss: 0.6865 - accuracy: 0.6187 - val_loss: 0.6830 - val_accuracy:
0.6500
Epoch 4/200
320/320 [==============================] - 0s 246us/step - loss: 0.6842 - accuracy: 0.6187 - val_loss: 0.6802 - val_accuracy:
0.6500
Epoch 5/200
320/320 [==============================] - 0s 246us/step - loss: 0.6821 - accuracy: 0.6187 - val_loss: 0.6775 - val_accuracy:
0.6500
Epoch 6/200
320/320 [==============================] - 0s 243us/step - loss: 0.6803 - accuracy: 0.6187 - val_loss: 0.6753 - val_accuracy:
0.6500
```

Figure 8.7: Training of neural net



```
****Results****
Accuracy: 65.0000%
precision score:
 0.0
f1_score:
 0.0
confusion_matrix:
 [[52  0]
 [28  0]]
Specificity:
 1.0
Sensitivity:
 0.0
log_loss:
 12.08857173821874
```

Figure 8.8: Neural net Result

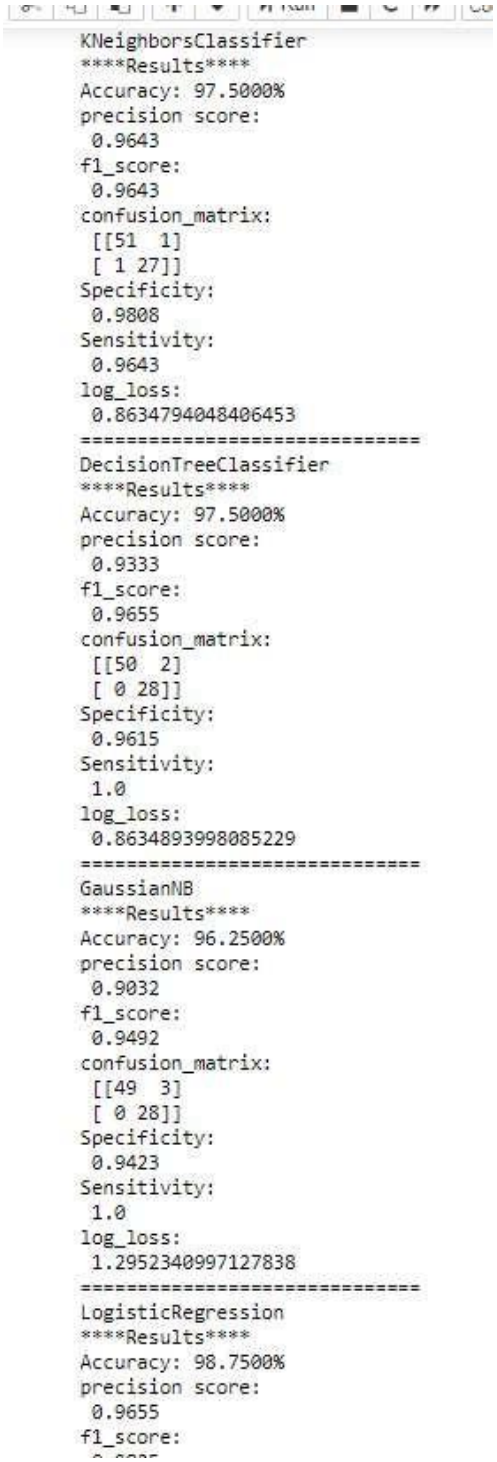


Figure 8.9: Results for all the models like decision tree etc.

Chapter 9: CONCLUSION AND FUTURE SCOPE

9.1 FUTURE SCOPE

- This would help detect the chances of a person having CKD further on in his life which would be really helpful and cost-effective people.
- This model could be integrated with normal blood report generation, which could automatically flag out if there is a person at risk.
- Patients would not have to go to a doctor unless they are flagged by the algorithms. This would make it cheaper and easier for the modern busy person.

9.2 CONCLUSION

This system presented the best prediction algorithm to predict CKD at an early stage. The dataset shows input parameters collected from the CKD patients and the models are trained and validated for the given input parameters. K-Nearest-Neighbors Classifier, Decision Tree Classifier, GaussianNB, Logical Regression and Artificial Neural Network learning models are constructed to carry out the diagnosis of CKD. The performance of the models is evaluated based on a variety of comparison metrics are being used, namely Accuracy, Specificity, Sensitivity and Log Loss. The results of the research showed that Logical Regression model better predicts CKD in comparison to the other models taking all the metrics under consideration. This system would help detect the chances of a person having CKD further on in his life which would be really helpful and cost-effective people. This model could be integrated with normal blood report generation, which could automatically flag out if there is a person at risk. Patients would not have to go to a doctor unless they are flagged by the algorithms. This would make it cheaper and easier for the modern busy person.

References

[1] M. P. N. M. Wickramasinghe, D. M. Perera and K. A. D. C. P. Kahandawaarachchi, "Dietary prediction for patients with Chronic Kidney Disease (CKD) by considering blood potassium level using machine learning algorithms," 2017 IEEE Life Sciences Conference (LSC), Sydney, NSW, 2017, pp. 300-303

[2] U. N. Dulhare and M. Ayesha, "Extraction of action rules for chronic kidney disease using Naïve bayes classifier," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016, pp.1-5.

[3] R. Devika, S. V. Avilala and V. Subramaniyaswamy, "Comparative Study of Classifier for Chronic Kidney Disease prediction using Naïve Bayes, KNN and Random Forest," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 679-684.

[4] G. Kaur and A. Sharma, "Predict chronic kidney disease using data mining algorithms in hadoop," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, 2017, pp. 973-979.

[5] Arif-Ul-Islam and S. H. Ripon, "Rule Induction and Prediction of Chronic Kidney Disease Using Boosting Classifiers, Ant-Miner and J48 Decision Tree," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 2019, pp. 1-6.

[6] S. Vijayarani, S. Dhayanand, "KIDNEY DISEASE PREDICTION USING SVM AND ANN ALGORITHMS", International Journal of Computing and Business Research (IJCBR), vol. 6, no. 2, 2015

[7] Deepti Sisodiaa , Dilip Singh Sisodia, Prediction of Diabetes using three classification algorithms, Computational International Conference on Data Intelligence and Science (ICCIDS 2018)

[8] Maryam Soltanpour Gharibdousti Kamran Azimi, Saraswathi Hathikal, Dae H Won, Prediction of Chronic Kidney Disease using Data mining techniques, 2017 Industrial and

Systems Engineering Conference.

[9] Krupa Joel Chabathula, Jaidhar C.D, Ajay Kumara M.A Comparative Study of Principal Component Analysis Based Intrusion Detection Approach Using Machine Learning Algorithms , 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN).

[10] Guneet Kaur , Ajay Sharma , Predict Chronic Kidney Disease Using Data Mining Algorithms in Hadoop , International Journal of Advanced Computational Engineering and Networking,
ISSN: 2320-2106, Volume-5, Issue- 6, Jun-2017

[11] "UCI Machine Learning Repository: Chronic_Kidney_DiseaseDataSet",Archive.ics.uci. edu,2015.Available:http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease

[12] www.kidneyfund.org/kidneydisease/chronic-kidney-diseaseckd/