PREDICTION OF CHRONIC KIDNEY DISEASE (CKD) USING MACHINE LEARNING

ABDUL WAZED RIFAT

ABU SAKER SOBAIR HASAN

DIPOK SARKAR DIPU

A THESIS SUBMITTED

FOR THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

2020

SUPERVISOR'S APPROVAL

This thesis paper titled **"PREDICTION OF CHRONIC KIDNEY DISEASE (CKD) USING MACHINE LEARNING"**, submitted by Abdul Wazed Rifat, ID: 201714029, Abu Saked Zobair Hasan, ID:201714065 and Dipok Sarker Dipu, ID:201714021 has been ac-cepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in 2020.

Professor Syed Akhter Hossain
Head of Computer Science and Engineering department
University of Liberal Arts Bangladesh

DECLARATION

This is to certify that the work presented in this thesis paper, titled, "PREDICTION OF CHRONIC KIDNEY DISEASE (CKD) USING MACHINE LEARNING", is the outcome of the investigation and research carried out by the following students under the supervision of Professor Syed Akhter Hossain, Head of Computer Science and Engineering department, University of Liberal Arts Bangladesh.

It is also declared that neither this thesis paper nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Abdul Wazed Rifat
Roll: 201714029
14 March 2021

Abu Saker Zobair Hasan
Roll: 201714065
14 March 2021

Dipok Sarker Dipu
Roll: 201714021
14 March 2021

ABSTRACT

In this research, we tried to predict Chronic Kidney Disease(CKD) using Machine Learning algorithms.At first, we selected 7 algorithms for the majority rule method which was executed as an ensemble algorithm. The results of the 7 algorithms were fed into the ensemble algorithm and the final result was elected through hard voting.All the 7 algorithms were classification algorithms. As our dataset contains only two possible results CKD or NotCKD it is possible to deduce the result of the data set only by using classification algorithms. As our algorithms contain 2 classes, the 7 algorithms only had to determine one of the two classes and feed the determined result into the ensemble . Finally by using the voting classifier, in other word, due to ensemble algorithms it was possible to deduce the result with high accuracy.

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor, Professor Syed Akhter Hossain, Head of Computer Science and Engineering department, University of Liberal Arts Bangladesh, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis. We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

LIST OF ABBREVIATION

CKD : Chronic Kidney Disease
NotCKD : Not Chronic Kidney Disease
ACR : Albumin to Creatinine Ratio
GFR : Glomerular Filtration Rate
eGFR : Estimated Glomerular Filtration Rate
CVD : Cardiovascular disease
ML : Machine Learning
LR : Logistic Regression
DT : Decision Tree
SVM : Support Vector Machine
RF : Random Forest
GB : Gradient Boosting
CNN : Convolutional Neural Network
NB : Naive Bayes
EVC : Ensemble Voting Classifier

LIST OF SYMBOLS

$\mathbf{H}_\theta(x)$	: Hypothesis Value
y	: Actual Value
J(θ)	: Cost Function
m	: Number of features
α	: Step Size
$\theta_\mathbf{j}$	: Vector of Parameter
p	: Probability(Yes/No) value of an attribute
$\mathbf{X_{std}}$	: Standard Deviation of the value of an attribute
$\mathbf{X_{scaled}}$	: Scaled value of an attribute
$\mathbf{X_{min}}$	: Minimum value of an attribute
$\mathbf{X_{max}}$	: Maximum value of an attribute

CHAPTER 1
INTRODUCTION

1.1 Research Background

In this era of the modern age, humankind is more than partially dependent on machines for various kinds of tasks but while taking in count things that require handling of human life,people prefer doctors more than machines. But in today's age even that is changing gradually. In recent years many scientists and researchers have been trying to establish a generalized system that would be able to predict CKD perfectly depending on the data that was provided hence machine learning. As there have been many cases of CKD in the past it is possible to generate a set of data to train a machine to predict CKD and solve the problem much efficiently. Mainly classification techniques are used to predict if an individual has CKD or not. To use various features of human bodies and states of the parts of the body,a neural network is used. Even though its an age old technique but the machines were unable to implement this system because those machines at the time were simply not adequate enough.But as time passed with the use of transistors,computers became more powerful eventually and was able to implement neural system and other machine learning techniques which uses combinations of feature to come to a optimized conclusion in a classification problem.

1.2 Problem Statement

Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then excreted in your urine. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes and wastes can build up in your body.

In the early stages of chronic kidney disease, you may have few signs or symptoms.This is because the body is usually able to cope with a significant reduction in kidney function [1]. Chronic kidney disease may not become apparent until your kidney function is significantly impaired.Some of the symptoms are weight loss and poor appetite,tiredness,need to urinate more often etc [2].

To find out the kidney damage, mainly blood test and sometimes urine test is done to see the blood and ACR in the urine. From blood tests, doctors find out eGFR which is, how many millilitres of waste patient kidneys should filter in a minute. Result of eGFR fall into 5 categories [3] which are shown in Figure 1.1

Stage	Description	GFR, ml/min/1.73 m^2
-	At increased risk	≥ 60
1	Kidney damage with normal or increased GFR	≥ 90
2	Kidney damage with mild decreased GFR	60-89
3	Moderately decreased GFR	30-59
4	Severely decreased GFR	15-29
5	Kidney failure	< 15 (or dialysis)

Figure 1.1: GFR categories in CKD

In 2017, CKD resulted in 1.2 million deaths and was the 12th leading cause of death worldwide. In addition, 7.6% of all CVD deaths (1.4 million) could be attributed to impaired kidney function. Together, deaths due to CKD or to CKD-attributable CKD accounted for 4.6% of all-cause mortality [4]. According to the latest WHO data published in 2018 Kidney Disease Deaths in Bangladesh reached 16,948 or 2.18% of total deaths. The death rate of 14.83 per 100,000 of population ranks Bangladesh 94 in the world [5]. Treatment for chronic kidney disease focuses on slowing the progression of the kidney damage, usually by controlling the underlying cause. If chronic kidney disease can progress to end-stage kidney failure, then it will be fatal without artificial filtering (dialysis) or a kidney transplant.So,we need to detect the CKD at its early stages to reduce the treatment cost and provide better treatment to the patient.
As we are in the digital age, what is better than using machines to do this kind of detection.So, we will predict Chronic Kidney Disease (CKD) using machine learning.

1.3 Thesis Objectives

Based of previous discussion, the main objectives of our thesis are:

1. To predict chronic kidney disease.
2. To design prediction models using machine learning.
3. Compare different machine learning prediction models.
4. To deploy the model in the real-life application for predicting CKD correctly from data.

1.4 Methodological Overview

1.4.1 Research type

Our research type is quantitative .
Quantitative research is the process of collecting and analyzing numerical data which will be in csv format. It can be used to find patterns and averages, make predictions, test causal relationships, and generalize results to wider populations.
Our thesis is numerical, non-descriptive, applies statistics , mathematics and uses numbers. It uses an iterative process whereby a patient's medical history is evaluated. The results are presented in tables and graphs based on which we will be able to predict if a patient is to be or not to be diagnosed with CKD. It is conclusive. It investigates the what, where and when of decision making which will conclusively and with high accuracy be able to predict CKD.

1.4.2 Research design

Our research design is experimental design.
It enables us to maintain control over all factors that may affect the result of an experiment. In doing this, we will attempt to predict what may occur. This classic experimental design specifies an experimental group (test data) and a control group (train data) . The independent variable is administered to the experimental group and not to the control group, and both groups are measured on the same dependent variable(features). Subsequent experimental designs have used more groups and more measurements over longer periods. Our experiment has control, randomization, and manipulation.

1.4.3 Data collection

We have collected a benchmark dataset from the University of California Irvine Machine learning repository [6].
A total number of 400 instances were stored in this dataset which was collected from the Apollo Hospitals,Karaikudi, Tamilnadu, India.Every attribute contained missing values except the class attribute.

1.4.4 Sample

The data set includes a total of 400 samples.Among the 400 instances, there were 250 instances with "ckd" class which is 62.5% of the whole data and the rest 150 instances were labeled as "notckd" class which is 37.5% of the whole data.
Among this 400 samples 80% is used to train the model and 20% is used to test .

1.4.5 Sampling tool

The dataset is subjected to cross-validation. Some samples are provided with missing values in the original data. For those values, each missing value was filled with the average of the corresponding attribute within

the category to which it belongs.

1.4.6 Analysis of data

In this research, Machine learning techniques have been used for the analysis of data.We will use some Machine Learning algorithms here.
To compare the machine learning technique we have used Accuracy and Confusion Matrix.

1.5 Scope of the Thesis

The main focus of our thesis is to predict Chronic Kidney Disease(CKD) based on some feature. After reading related works regarding this field,we have picked a dataset containing important features.The features include 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, hemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus,coronary artery disease,appetite, pedal edema, anemia,class lebel-total 25 feature has been used to create the machine learning model that will predict CKD.

1.6 Organization of the Thesis

Our thesis is organized into five chapters.

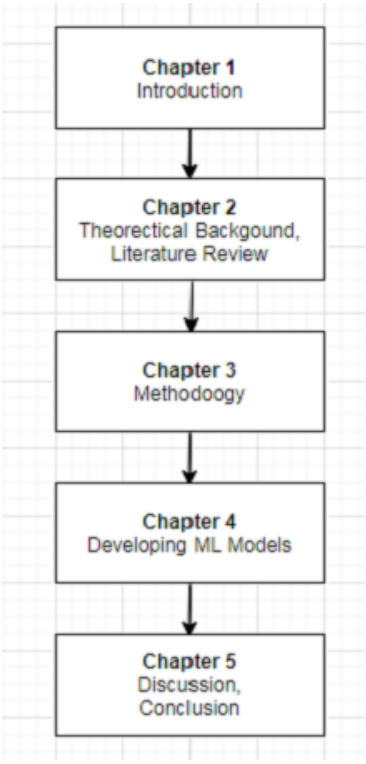

Figure 1.2: Organization of Thesis

In the first chapter, we talk about the motivation as to why we selected this topic as our thesis. We fixed our thesis objectives and its scope and also did an initial introduction of the methodology we are going to be using in this thesis.
In the next chapter, we talked about the techniques that we are going to be using to achieve our thesis objectives. We have also discussed and studied the related works,which has helped us in solving many of our problems.
Third chapter, we did our data related work like data collection, preprocessing, feature selection etc. in this chapter.
In the fourth chapter, we dive deep into our Machine Learning Models that we are going to be using to reach our thesis objectives. Using the theoretical knowledge from 'Chapter Two' we tried to develop and modify different ML models so that those models will be useful to predict CKD. Evaluation of different ML models also done in this chapter.
In the last chapter, we talk through, is our thesis able to meet its objectives, if it meets its objectives where can this thesis contribute, what kind of problems we faced when we were doing our thesis and what does future hold for this thesis.

CHAPTER 2
THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 Machine Learning

As we know, most of the scientific aspects are moving towards machine learning because it requires very low amount of interaction between human and computer in order to create an operational platform, based on collected data. We generally require two main components in this sector: a suitable algorithm and sufficient amount of data from real world experimentation or test. There are generally four types of machine learning algorithms: supervised, semi-supervised, unsupervised and reinforcement.

2.1.1 Supervised Learning

In supervised learning, the machine is taught by example, in other word datasets with given results for the particular data points. Here the given dataset contains both input and output . The algorithm is tasked to find the method to get the output from the input in the dataset. The algorithm identifies the pattern in the data, learns from the data and then makes predictions. This process continues until the algorithm reaches the peak of its performance based on the given dataset.

2.1.2 Semi-supervised Learning

It is like supervised learning but the difference is that it uses both labeled and unlabeled data.
Labeled data comes with a tag so any algorithm can understand the data but unlabeled data lacks the tag hence lacks the information.By using this combo, we can certainly label the unlabeled data.

2.1.3 Unsupervised Learning

This type of learning algorithm identifies the pattern in the dataset. As there is no answer or instruction provided, the answer is predicted by analyzing the given data. Here algorithm uses large data sets and addresses those data so that it can find patterns in the dataset. Finding the pattern enables the algorithm to organize the data in such a way that it can describe the structure of the data. This includes clustering the data.

2.1.4 Reinforcement Learning

It focuses on strictly organized learning processes, where the algorithm is provided with a label dataset. Here the algorithm tries out different options. It teaches the machine through the process of trial and

error. In other words, it learns from the past mistakes and experience and begins to adapt according to the situation.

2.2 Machine Learning Algorithms

Machine learning uses algorithms to transform a data set into a model. American computer scientist Arthur Samuel stated, the term 'machine learning' is defined as a "computer's ability to learn without being explicitly programmed".In a basic sense, machine learning uses programmed algorithms that primarily receive and analyse input data to predict output values within an acceptable range. As new data is fed to these algorithms,they learn and optimise their operations to improve performance, developing intelligence over time. We used an ensemble algorithm to establish our machine learning model to predict chronic kidney disease as accurately as possible. As it is a voting classifier, we used 7 algorithms to increase the accuracy of the model.In this 7 algorithms there consists both supervised and unsupervised learning. In this section,we will briefly discuss the algorithms used to train our model.

2.2.1 Logistic regression

Logistic regression generates a working model by using a logistic function.The main goal of the process is to model a binary or bool-type variable. In order to predict which class the data belongs to, a threshold can be set. Based on that threshold, we can set a data to either of the two classes. In other words, it returns the probability that the binary dependent variable may be predicted from the independent variables. [7]
To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes. Say, if predicted value ≥ 0.5, then classify email as spam else as not spam.
Decision boundaries can be linear or nonlinear. Polynomial order can be increased to get a complex decision boundary.

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathsf{Cost}(h_\theta(x), y)$$

$$\mathsf{Cost}(h_\theta(\mathbf{x}), y) = -ylog(h_\theta(\mathbf{x})) - (1-y)log(1-h_\theta(\mathbf{x}))$$

Gradient descent for logistic regression:

while not converged {

$$\theta_j^{\mathsf{new}} = \theta_j^{\mathsf{old}} - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \text{ for } j = 0, 1, \ldots, n$$

}

Here gradient descent reduces the cost function to minimize the number of errors and get a more accurate

results so logistic regression algorithm.

2.2.2 Decision tree

In the decision tree we prioritized the attributes according to their contents or based on their discriminating values. A decision tree is actually a flowchart like structure or more precisely a reverse tree like structure in which each internal node represents a test operated on an attribute, each link represents the decision taken into account for the test, and each leaf node represents the final outcome .
The decision tree algorithm generates the model by simply splitting the source set into subsets based on an attribute value set with some preset rules. This process is repeated on each subset in a recursive manner until the subset at a node contains all the same value of the target labels. This process is called recursive partitioning.

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

Decision tree works for both specific and continuous input and output variables.Moreover decision tree can be applied to both classification and regression problems. [8]

2.2.3 SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm. Even though it can be used in both classification and regression, we used it for classification only. It is mostly used in classification problems. In SVM, the data points are planted according to the value of the feature. For SVM, it is important to select the marginal line with the highest distance from the support vectors to get high accuracy. Chronic Kidney disease is predicted using classification techniques of data mining and SVM is used here. [9]

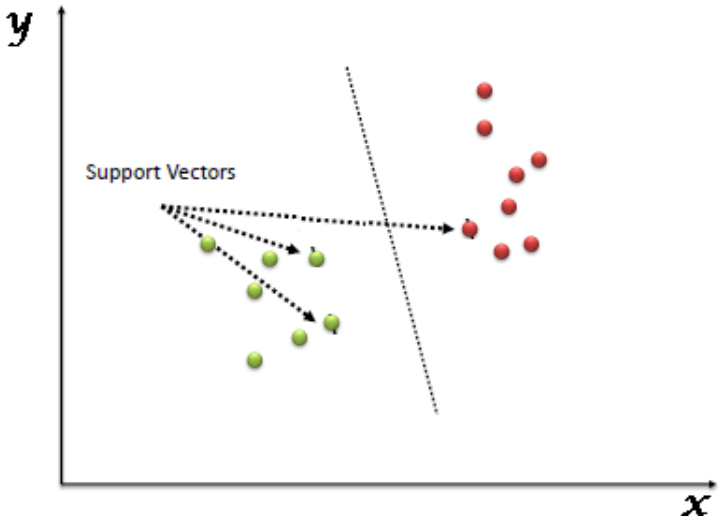

Figure 2.1: SVM

2.2.4 Random forest

Random forest is a supervised machine learning algorithm. The forest that is being built here is composed of decision trees. It is done through the bagging method. The theory behind random forest is that the more decision trees in the forest, the better the accuracy as it just combines multiple decision tree models. The fundamental concept behind random forest is a simple yet a powerful one which is "the wisdom of crowds". In data science theory, the reason that the random forest model works so well is "high number of correlated models working composedly will outperform any individual model". So naturally random forests require a large number of training samples to avoid over-fitting the training data. [10]

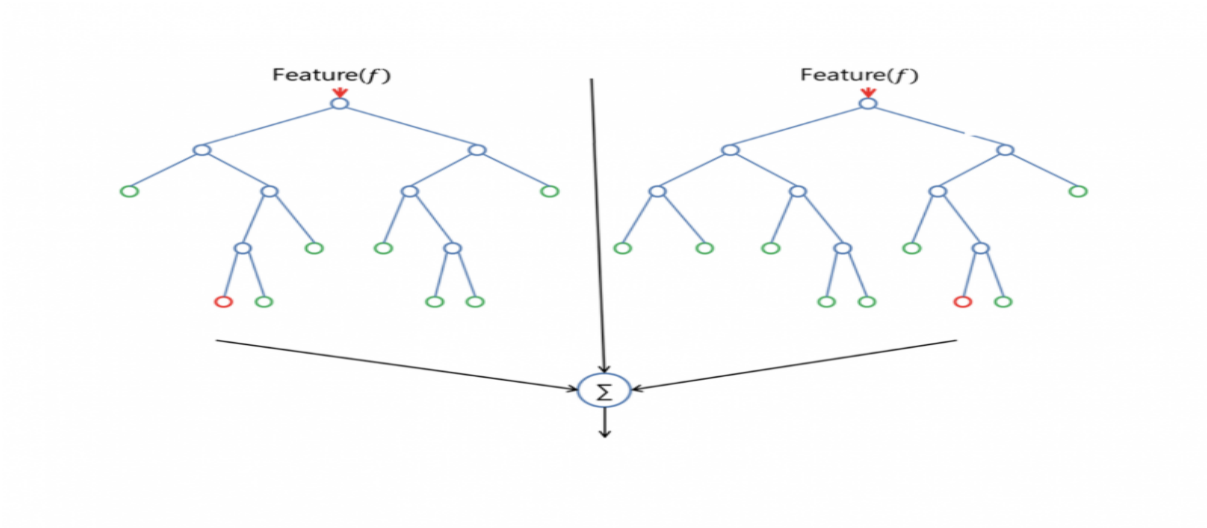

Figure 2.2: Random forest

Figure 2.2 indicates two trees based on a feature is adding as a part of the forest.

2.2.5 Gradient boosting

In gradient boosting algorithm boosting states a method of converting weak learners into strong. Here, while a new tree is generated, the tree represents a new modified version of the existing dataset. To understand gradient boost algorithm, knowledge of Adaboost algorithms is necessary. Adaboost starts by training a decision tree in which each observation is weighted equally. In the 2nd tree if an observation is harder to classify then the assigned weight is increased otherwise the weight is decreased. Subsequent trees help to classify observations that are not well classified by the previous trees.

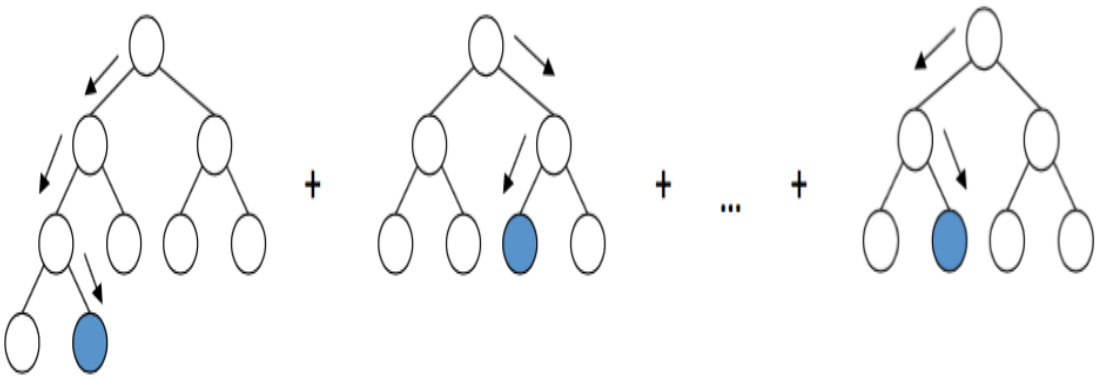

Figure 2.3: Gradient boosting

On the other hand, gradient boosting trains model in a gradual sequential and additive manner. The difference between these two algorithms are the method of identifying the shortcomings of weak learners. AdaBoost model identifies the shortcomings by using high weight data points and gradient boosting performs the same by using gradients in the loss function. The loss function is a measure indicating how good the model's coefficients are at fitting the underlying data.

2.2.6 Convolutional Neural network

Though CNN is often used in image processing it can also be used for natural language processing. First we have to look at the regular neural network. Regular neural network consists of an input layer, a hidden layer and an output layer. Each layer has its own neuron and each neuron is connected to the neuron from the previous layer in addition each layer has its own weight. In CNN each data is considered as spatial and instead of connecting to all the neurons of the previous layer the neurons are connected to the neurons that are closest to the in addition all the neurons have the same weight. CNN 's internal layer structure is similar to the traditional neural networks with the exception of having a pooling layer and convolution layer. There is also a 'Relu' layer for ensuring non-linearity as data is moved through each layer in CNN. 'Relu' is needed in order to maintain dimensionality. At the end there is a fully connected layer that allows us to classify the object. Convolutional layer is used for generating a convoluted feature map. In the pooling layer the feature map is down-sampled. It reduces the number of parameters.

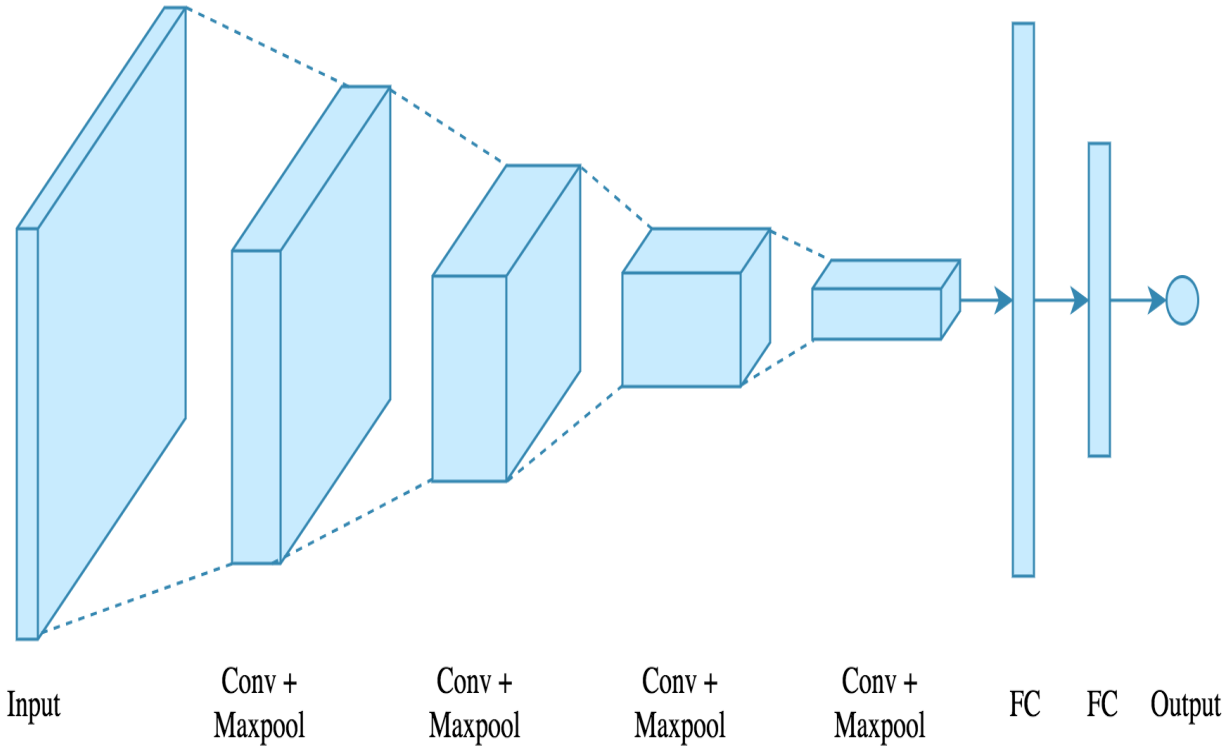

Figure 2.4: Convolutional Neural network

2.2.7 Naive Bayes

Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The crux of the classifier is based on the Bayes theorem.The bias theorem is-

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors are independent. That is, the presence of one particular feature does not affect the other. Therefore it is called naive.

2.2.8 Ensemble classifier

All the above algorithm models are fed into an ensemble classifier for the voting process. Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model.Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance ,bias, or improve predictions .

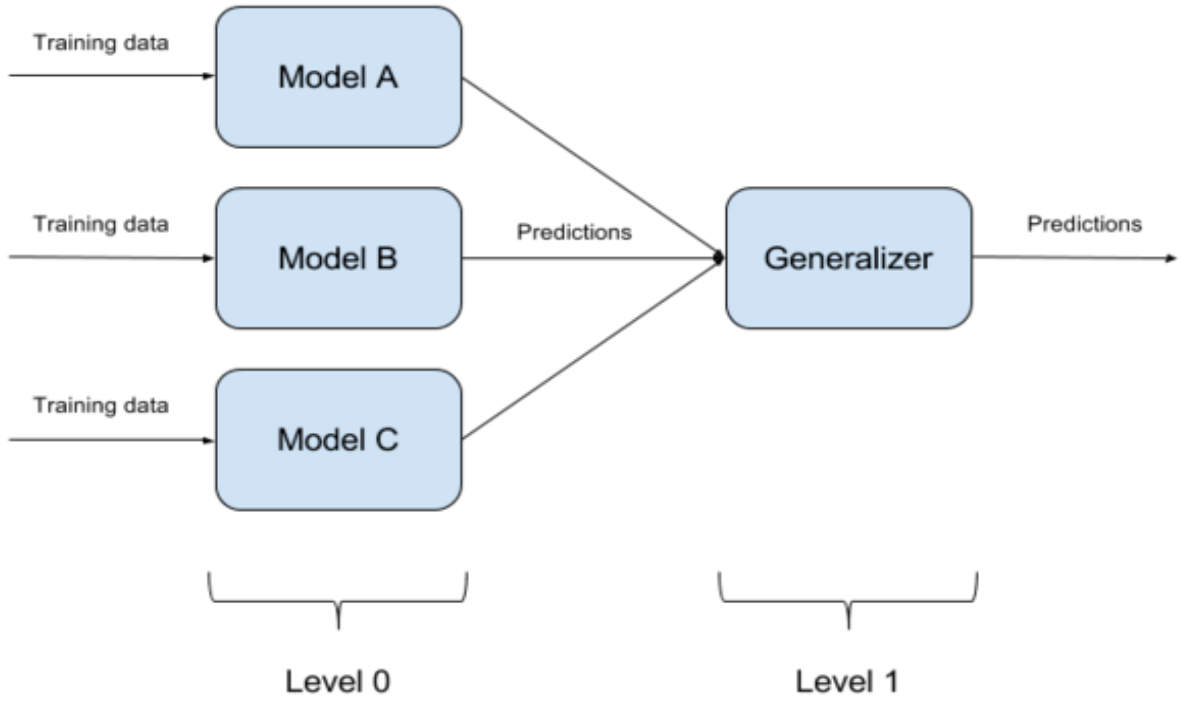

Figure 2.5: Ensemble classifier

2.3 Related Works

Sinha et al. [11] conducted a study to compare the performance of Support vector machine (SVM) and K-Nearest Neighbour (KNN) classifiers on the basis of its accuracy, precision and execution time for CKD prediction. It was observed that the performance of KNN classifiers is better than SVM. QIN et al. [12] conducted analyzing the misjudgments generated by the established models, they proposed an integrated model that combines logistic regression and random forest. Which was simulated for 10 times . After unsupervised imputation of missing values in the data set by using KNN imputation, It was observed that the integrated model could achieve a satisfactory accuracy.Sharma et al. [13] conducted research by using twelve classifiers to their dataset by using MATLAB 2016a and the results were obtained and analyzed in the terms of predictive accuracy, sensitivity, precision specificity. Linear Discriminant, Quadratic Discriminant, Linear SVM, Quadratic SVM, Fine KNN, Medium KNN, Cosine KNN, Cubic KNN, Weighted KNN, Feed Forward Back Propagation Neural Network using Gradient Descent performed better than the decision trees in terms of sensitivity. It was observed that the overall performance of decision trees was better than all the other classifiers for this study. K. M. Zubair Hasan and Md. Zahid Hasan [14] conducted extensive experiments on CKD datasets from the UCI machine learning repository showing that their ensemble-based model achieves state-of-the-art performance. In their paper, they propose an ensemble method based machine learning algorithm to improve the performance of the classifier for kidney disease. In contrast with quite a few classic prediction algorithms,It was observed that the classification accuracy of their proposed ensemble learning algorithm achieved high classification accuracy. BILAL KHAN et al. [15] conducted research and their paper is centered around the usage of ML techniques because of its procedures of value, distinguishing hidden patterns and providing novel information . Seven distinctive ML methods; NBTree, SVM, J48, MLP, LR, andNB, that are used in the early past along with

CHIRP, uti-lized in this study were used to figure out which system will give the better classification outcomes in terms of accuracy and error rate.It was observed that their experimental results concluded CHIRP as a promising technique for CKD prophecy.

2.4 Critical Summary

Here we used 7 algorithms to vote as we trained our model. By using this method we are not directly dependent on any particular algorithm in general. Here we use the common voting method “Majority rules” to determine the result of any tuple that is taken as the input. Ultimately as our resultant model has the involvement of 7 algorithms we can get the result more effectively and with more accuracy.

CHAPTER 3
METHODOLOGY

3.1 Feature Selection

We have to find out the features that are required to detect CKD disease, then we can feed those features into our machine learning model to accurately predict CKD patients. To select the required feature needed to precisely tell if a person has CKD or not, we have used literature survey which has been discussed before.
All of the features were selected from the literature review we have done in the previous chapter. These literature were selected based upon the problem they were addressing , which coincide with our problem statement. These literature were found on the web,published by different publishers/websites. Some of them were researchgate, SpringerLink, ScienceDirect, IEEEXplore, Google Scholar etc. To find the literature of related works some search terms like “CKD using ML”, “Prevention of CKD”, Chronic Kidney Disease”,”Detection of CKD”, “Early detection of CKD using machine learning” etc were used. Among hundreds of related works, we have chosen 28 based on similarities of our problem statement.
In almost every literature review,they used a total of 24 same features to train the ML model to make a model that will be able to accurately detect CKD. These literature used a benchmark CKD dataset from UCI repository. Removing any of the features will produce error which is harmful for our study.

3.2 Data Collection

We collected data from the University of California Irvine Machine learning repository [6].
A total number of 400 instances were stored in this dataset which was collected from the Apollo Hospitals ,Karaikudi, Tamilnadu, India. The features include age, blood pressure(BP), gravity, albumin, sugar, red blood cells(RBC) ,pus cell(PC), pus cell clumps(PCC), bacteria, blood glucose random(BGR), blood area(BA), serum creatinine(SC), sodium, potassium, hemoglobin, packed cell volume(PCV), white blood cell count(WBCc), red blood cell count(RBCC), hypertension(HTN), diabetes mellitus(DM), coronary artery disease(CAD), pedal edema(PE), anemia(ANE), appetite (APPET). The output variable was named as class which contained only two values, ckd and notckd. Every attribute contained missing values except the class attribute. Among the 400 instances, there were 250 instances with “ckd” class which is 62.5% of the whole data and the rest 150 instances were labeled as “notckd” class which is 37.5% of the whole data.

3.3 Data Preprocessing

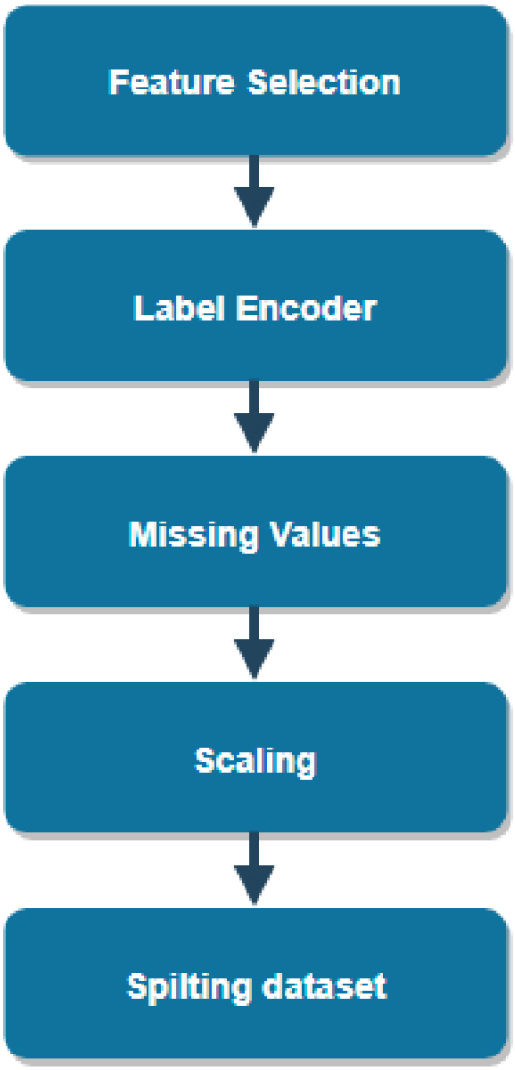

Figure 3.1: Data Preprocessing

3.3.1 Feature Selection

As we have got only one dataset to train our model and as every little attribute in the dataset plays a role in detecting CKD, we did not remove any of the features from our dataset.

3.3.2 Label Encoder

When we explored the dataset, we saw categorical value in many attributes. As our model is not capable of handling categorical value, we encode those attributes into binary values such as Table 3.1

Attributes	1	0
RBC	Normal	Abnormal
PC	Normal	Abnormal
PCC	Present	Notpresent
BA	Present	Notpresent
HTN	Yes	No
DM	Yes	No
CAD	Yes	No
PE	Yes	No
ANE	Yes	No
APPET	Good	Poor
Class	Notckd	ckd

Table 3.1: Encoding Table

3.3.3 Missing values

In our dataset we have missing values in some attributes. To fill out those missing values we used the mean value of that attribute (missing value = round(attribute. Mean) upto 2 decimal points).

3.3.4 Scaling

To make our ML model impartial to dataset values, we scale our dataset. We used min-max scaling to scale our data. We used this formula :

X_{std} = (X - X_{min}(axis=0)) / (X_{max}(axis=0) - X_{min}(axis=0))

X_{scaled} = X_{std} * (max - min) + min

and these data :

min = [0, 0, 1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2000, 0, 0, 0, 0, 0, 0, 0]

max = [120, 360, 1.1, 5.5, 5.5, 1, 1, 1, 1, 700, 500, 100, 200, 100, 50, 100, 30000, 20, 1, 1, 1, 1, 1, 1]

to scale the whole dataset between 0 and 1

3.3.5 Splitting Dataset

The UCI dataset consists of 400 samples. To train and test our ML model ,we split the dataset 8:2 ratio, where 80% data is used for training our ML model and 20% data for testing our model. We used the scikit-learn Python machine learning library train_test_split() function.

CHAPTER 4
DEVELOPING AND EVALUATING ML MODELS

4.1 Developing ML Models

In this phase we have chosen 7 different machine learning algorithms based on accuracy and low amuont of false negative errors. The algorithms that were used are Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Gradient Boosting (GB), Convolutional Neural Network (CNN), Naive Bayes (NB). We then used Ensemble voting classifier to get output from these 7 algorithms & return output that is supported by maximum algorithms. The models are developed using scikit-learn, which is a python library containing a large volume of machine learning algorithms & functions required for building machine learning models. To evaluate the performance of these algorithms a cross-validation train-test split was performed, which is also a built in function of scikit-learn. Here we have split our dataset & considered 80% data for training purposes & rest 20% is used for testing purposes. While using these machine learning models, we mostly used default parameters of machine learning models defined in scikit-learn. The parameters that are different from default values are explained in next

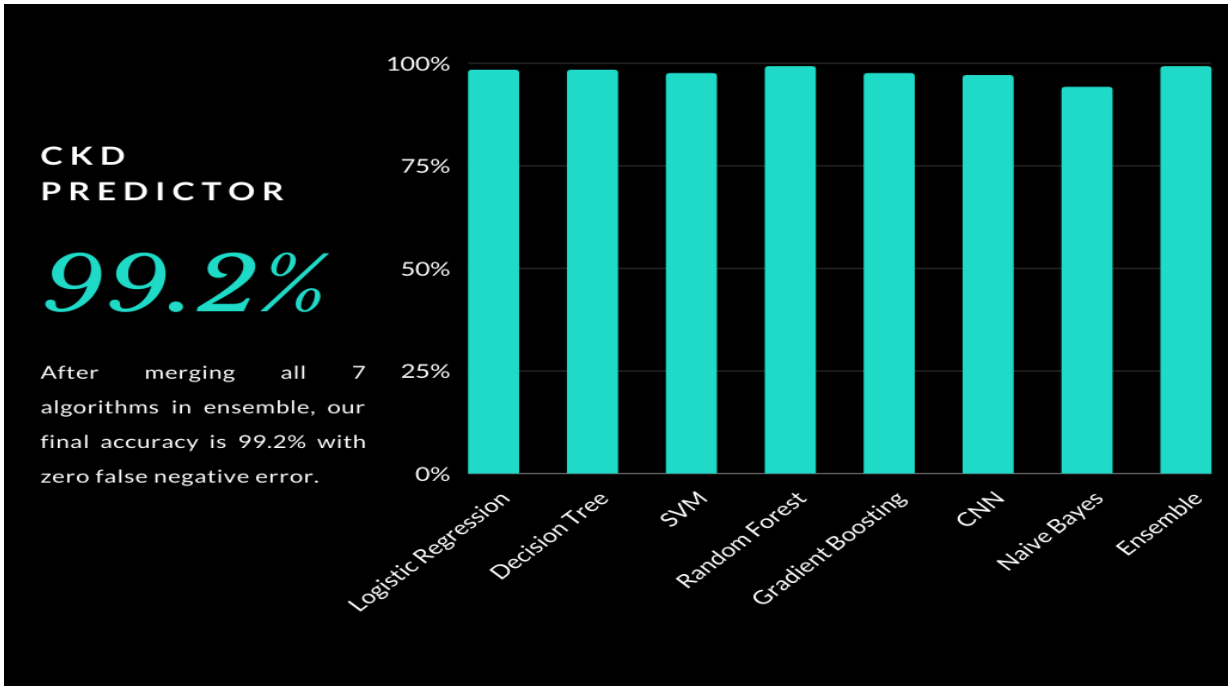

Figure 4.1: Performance Graph

section for each algorithm.

We have evaluated these algorithms based on accuracy. Then we have prioritized algorithms that have low false negative error. As we know false negative is more critical error than false positive error. If you tell someone that he/she is safe even though he has CKD, his life may be in danger. This is a false negative

error. But if you tell him/her that he/she has CKD even though he/she is healthy, their life won't be in danger.

4.1.1 Logistic Regression

For logistic regression, in order to avoid convergence error we have increased max iteration to 1000. After training by fitting our datasets into LR model & testing we have measured 1.7% false negative error & no false positive error. So, this algorithm has an accuracy of 98.3%.

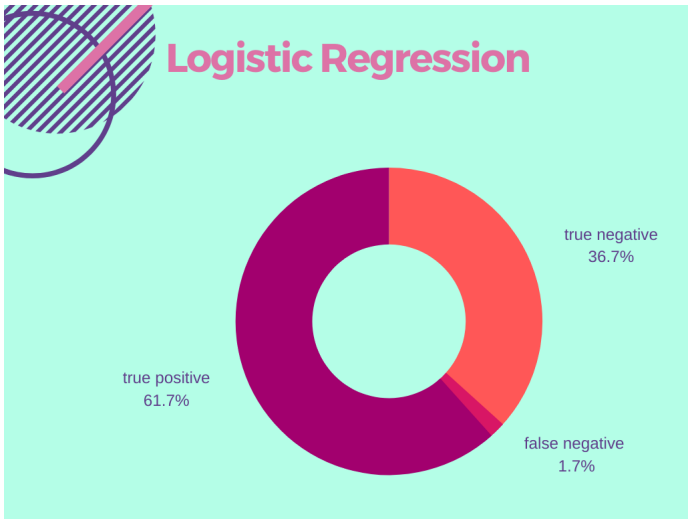

Figure 4.2: confusing matrix for Logistic Regression

4.1.2 Decision Tree

For the decision tree, we have used default parameters. After fitting our datasets & testing we have measured 0.85% false positive error & 1.85% false negative error. So, this algorithm has an accuracy of 98.33%.

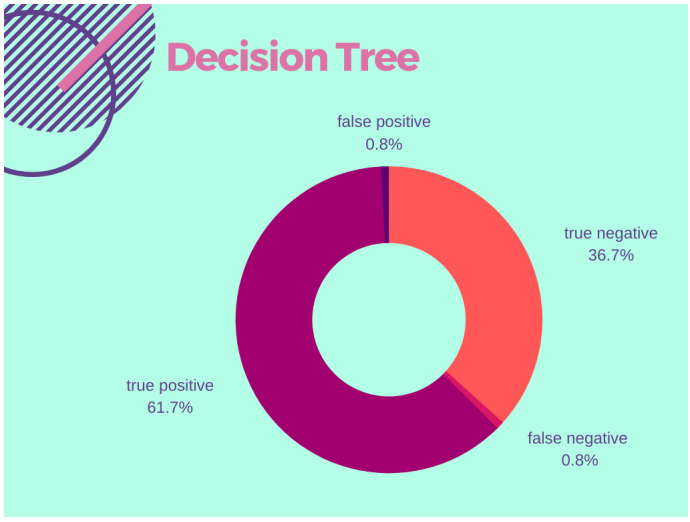

Figure 4.3: confusing matrix for Decision Tree

4.1.3 Support Vector Machine

Here we have used a ploynomial kernel. It represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. Here the degree of this polynomial kernel was 2.

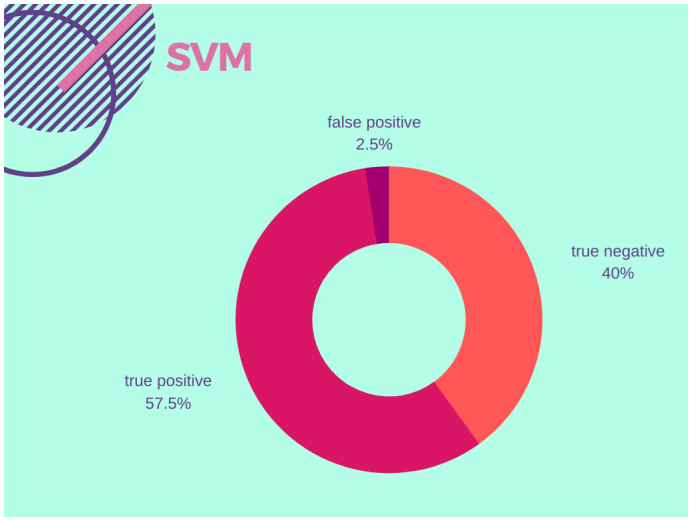

Figure 4.4: confusing matrix for Support Vector Machine

SVM has an accuracy of 97.5% while testing. But a very big advantage of svm was that it has no critical error in other words it has zero false negative error. So 2.5% error was false positive error.

4.1.4 Random Forest

In random forest we increased n_estimator(number of trees in forest) from 100 (default) to 1000 for better accuracy. As for other parameters, we have used default values. This resulted in a very good accuracy of 99.16%. It contains only 0.84% false negative error. No false positive error was found.

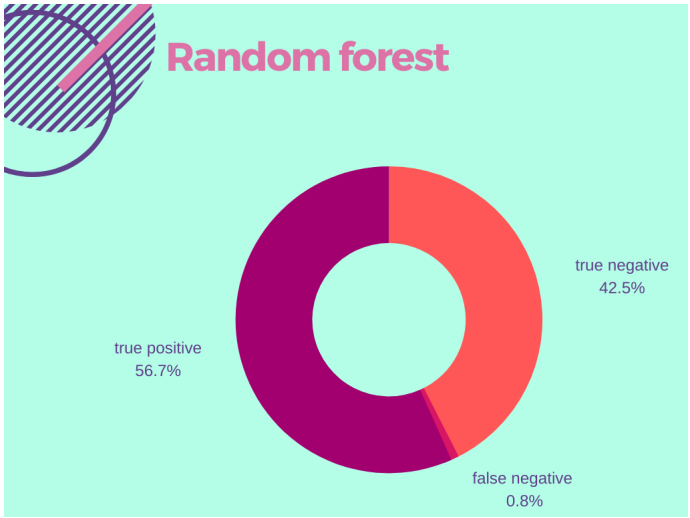

Figure 4.5: confusing matrix for Random Forest

4.1.5 Gradient Boosting

Here, The number of boosting stages to perform is set to 100. Gradient boosting is fairly robust to overfitting so a large number usually results in better performance. Learning rate shrinks the contribution of each tree by learning_rate. There is a trade-off between learning_rate and n_estimators. We used a learning rate of 0.7.The number of features to consider when looking for the best split is 24 as we have a total 24 features in our datasets. The maximum depth of the individual regression estimators is 1000. The maximum depth limits the number of nodes in the tree. The random seed given to each Tree estimator at each boosting iteration is fixed by setting value to zero. Finally we got an accuracy of 97.5% which contains only false negative error of 2.5%.

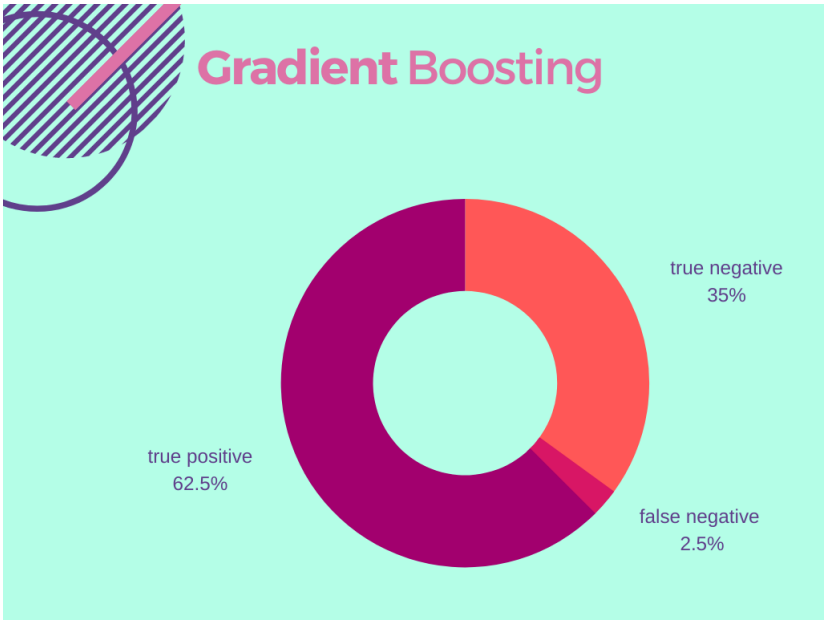

Figure 4.6: confusing matrix for Gradient Boosting

4.1.6 Convolutional Neural Network

We also used a neural network Multi-layer Perceptron classifier classifier known as CNN for classification. This model optimizes the log-loss function using LBFGS or stochastic gradient descent. We build this neural network using 2 hidden layers with 24 neurons for both hidden layers.The solver used for weight optimization is 'lbfgs'. 'lbfgs' is an optimizer in the family of quasi-Newton methods. L2 penalty (regularization term) parameter is set to 10-5. Max iteration increased to 1000. This algorithm had an accuracy of 97%. With 3% false positive error.

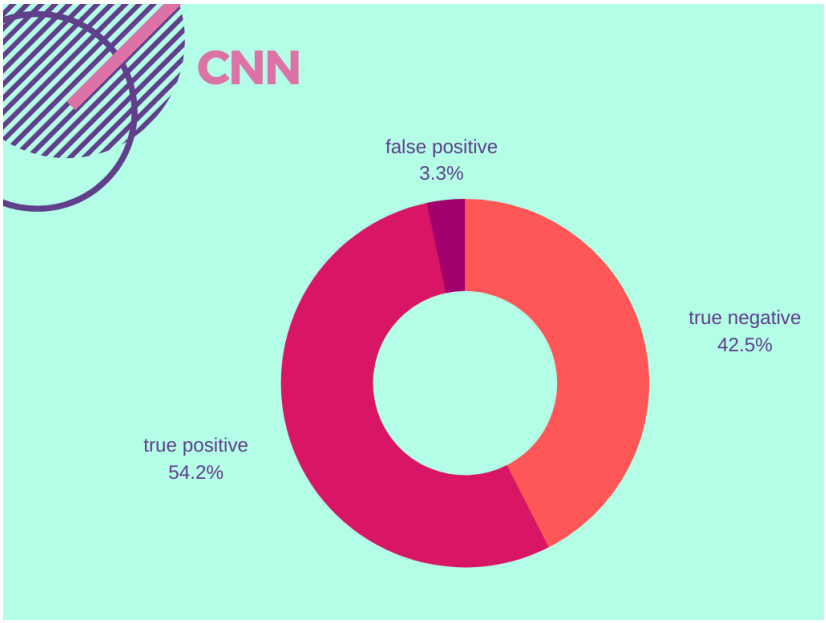

Figure 4.7: confusing matrix for Convolutional Neural Network

4.1.7 Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Default parameters are used here. It had a low accuracy of 94.17%. With 5.83% false negative error.

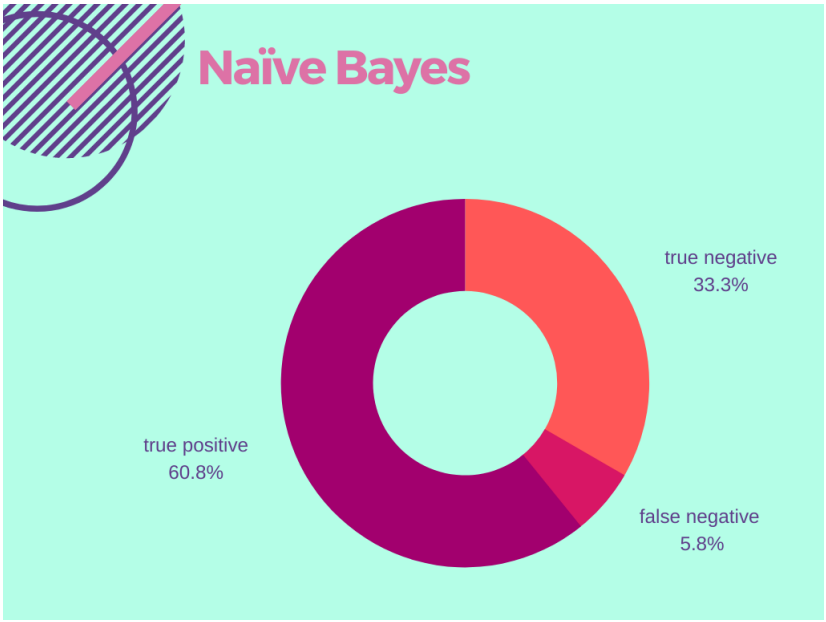

Figure 4.8: confusing matrix for Naive Bayes

4.1.8 Ensemble Voting Classifier

Finally, we had many good algorithm performances above. But to increase accuracy, we did not just choose the best algorithm. Rather we choose all this algorithm & use a voting classifier to get the output that is supported by maximum algorithms. That way, we have achieved more reliability. To avoid ties, we have used an odd number of algorithms. Thus We predicted class labels for majority rule voting. Finally we got the best accuracy of 99.2%. We were able to reduce critical error to zero. So we only had an 0.8% false positive error.

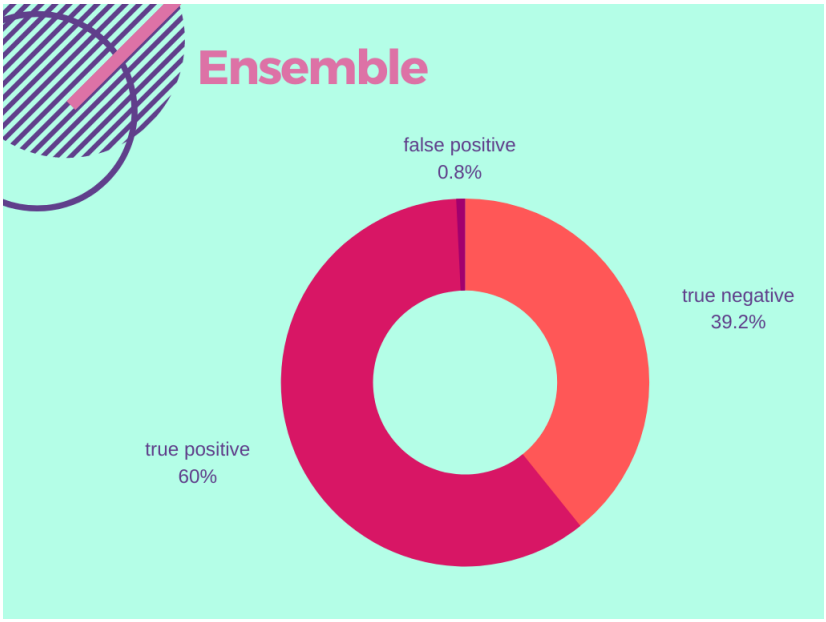

Figure 4.9: confusing matrix for Ensemble Voting Classifier

4.2 Evaluating ML Models

We evaluated these algorithms based on accuracy. Then we prioritized algorithms that have low false negative error. As we know false negative is more critical error than false positive error. If you tell someone that he/she is safe even though he has CKD, his life may be in danger. This is a false negative error. But if you tell him/her that he/she has CKD even though he/she is healthy, there is no life danger. So, we choose the best 7 algorithms depending on accuracy & critical values. Then we used a hard voting classifier to always get the best of all these algorithms. Our final output was supported by the majority of these 7 algorithms. So, along with accuracy, we also have increased reliability. For measuring critical error we have used confusion matrix.

		Prediction	
		No	Yes
CKD	No	True Negative	False Positive
	Yes	False Negative	True Positive

Table 4.1: Confusion Matrix

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be

in group i and predicted to be in group j.Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

4.2.1 Highest Accuracy by different algorithms

So Here's our independent accuracy & confusion matrix for each algorithms :

Algorithms	Confusion Matrix		Accuracy
Logistic regression	44	0	98.33%
	2	74	
Decision Tree	44	1	98.33%
	1	74	
Support Vector Machine	16	1	97.5%
	0	23	
Random Forest	51	0	99.16%
	1	68	
Gradient Boosting	14	0	97.5%
	1	25	
Convolutional Neural Network	51	4	97%
	0	65	
Naive Bayes	40	0	94.17%
	7	73	
Ensemble Voting Classifier	51	1	99.2%
	0	78	

Table 4.2: performance graph

4.2.2 Highest Accuracy by all features

As we can see, we had very good accuracy with each algorithm individually (above 94%). But after using ensemble voting classifiers, these algorithms were used together to get a better result of 99.2% with no false negative error. Here we have used all our 24 features from our datasets.

4.2.3 Minimum number of features with Highest Accuracy

Although we had a very good output accuracy, with very low severe error, we had some limitations. The data could collect & had to work with had a lot of null values. Now for some algorithms , we had to fill this null value with the average value of that column.
in Figure 4.10 every yellow part for each column stands for null value. So we can see it, some features have a lot of null values & some features have very little null values. So, features with little null values are the minimum requirements to get the highest accuracy possible.

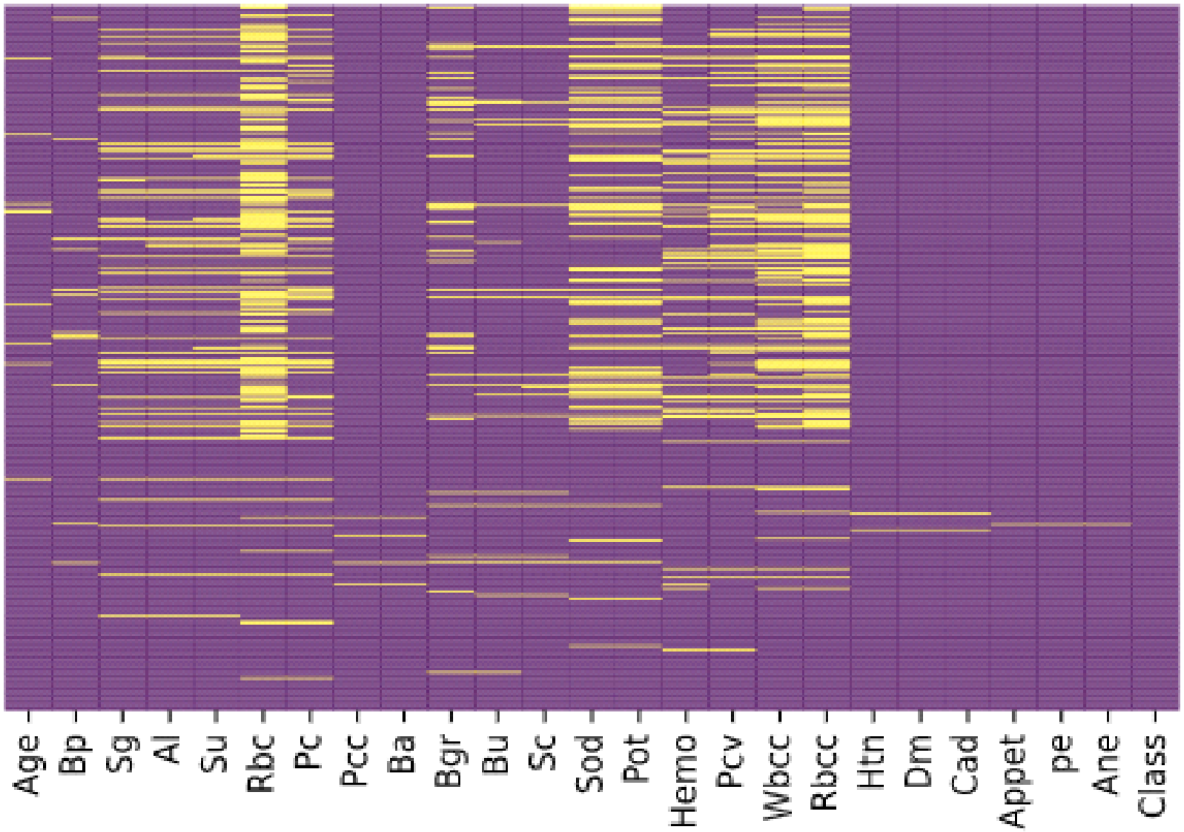

Figure 4.10: Missing values in dataset

CHAPTER 5
DISCUSSION AND CONCLUSION

5.1 Thesis Outcome

As for the outcome of our thesis project which has been previously stated “Chronic kidney disease prediction using machine learning“ has produced acceptable results according to the dataset that was provided by the UCI repository. The accuracy of predicting CKD is 99.2%. As a result of our thesis we were able to extract the trained model and deploy it as a fully functional web application which can be interacted by the both patient and doctors through an user friendly interface to get an prediction about the condition of one’s health by using the application.

To develop the front end we have used html, css, javascript, bootstrap, jquery. And to develop the backend, we have used python & flask. In our website, first page is a form where we collect data from user for all attributes : 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, Hemoglobin, Packed Cell Volume, White Blood Cell Count, Red Blood Cell Count, Hypertension, Diabetes Mellitus, Coronary Artery Disease, Appetite, Pedal Edema, Anemia. After clicking the submit button, these input datas are fit into our model, & CKD is predicted.

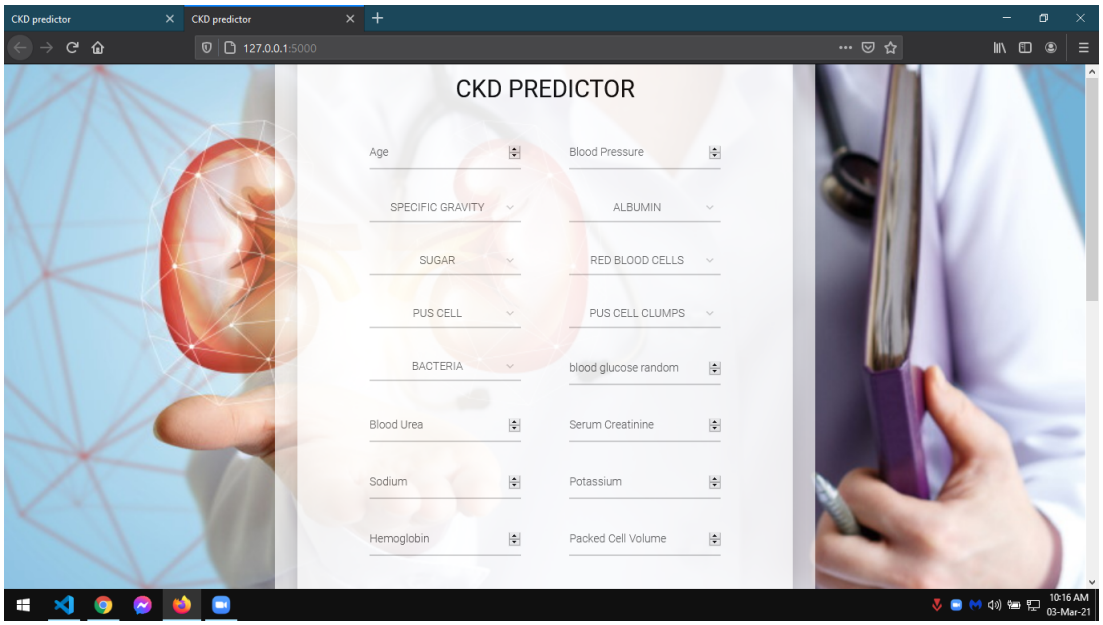

Figure 5.1: website front page first half

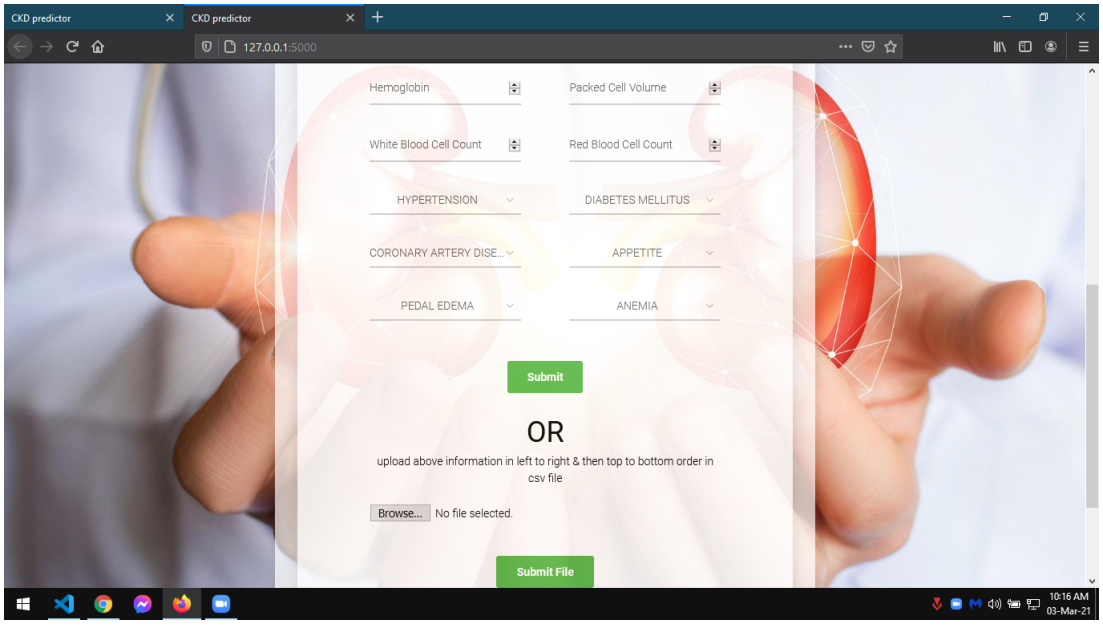

Figure 5.2: website front page second half

In Case, user wants to input data from a file, our website also accepts input datas in CSV format in correct order as explained in the website.

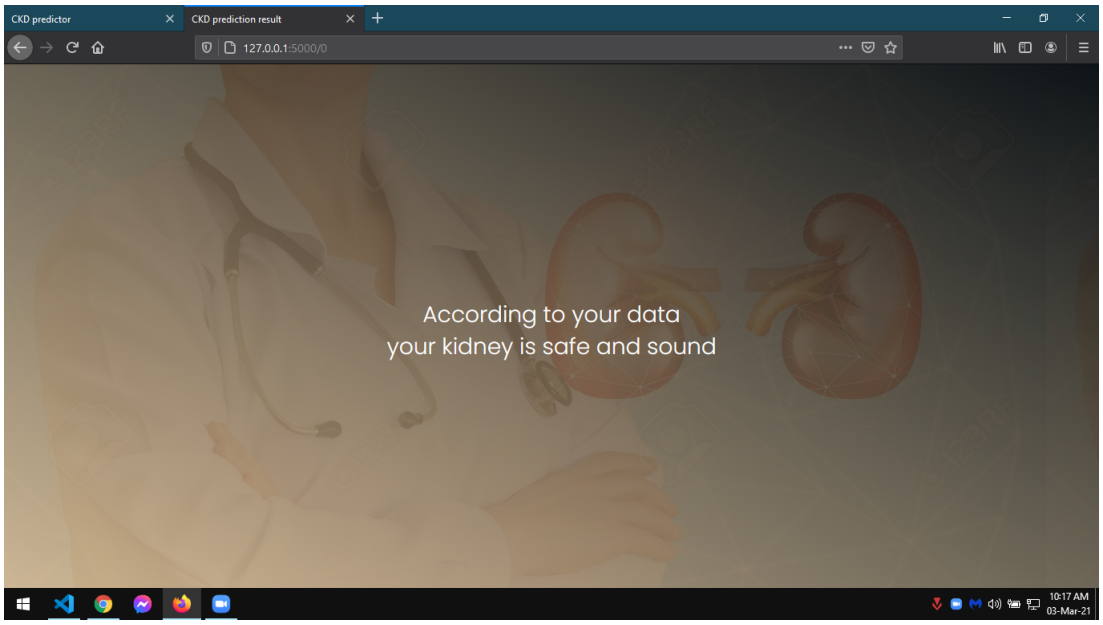

Figure 5.3: website final page positive

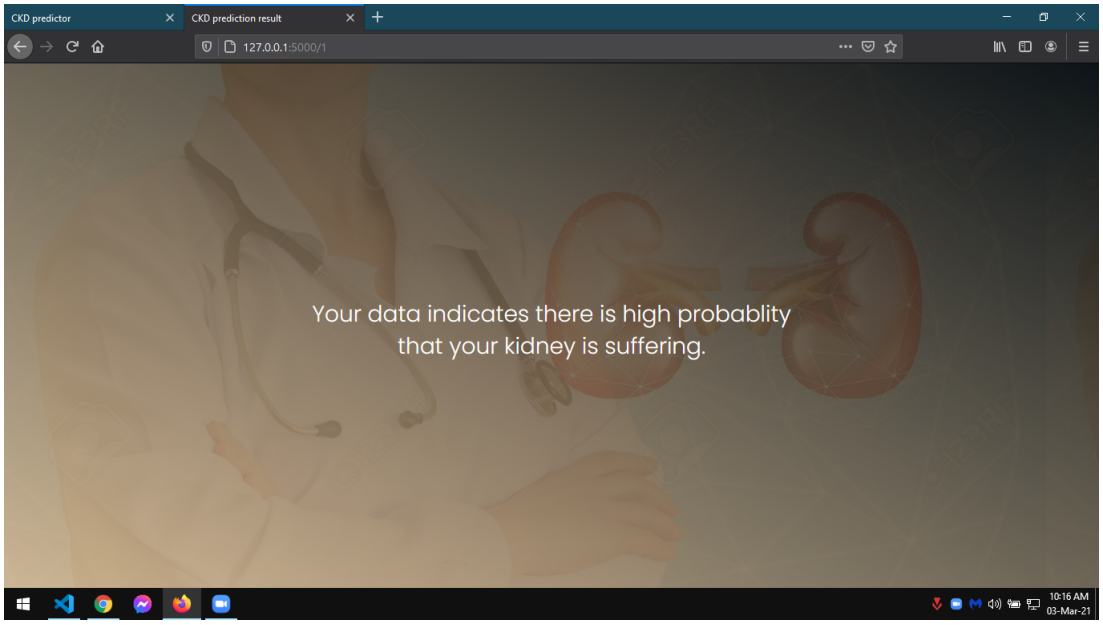

Figure 5.4: website final page negative

After prediction, we also give appropriate suggestions according to output.

5.2 Thesis Implication and Contribution

We had run about 15 algorithms for the purpose of predicting CKD as accurately as possible . While doing so we manage to find something interesting which contributes as equally as the accuracy of the model in predicting the CKD based on real time human data. That is if someone has the CKD and our model predicts that one does not have CKD bears far greater risk compared to the vice versa case. So based on this scenario we selected 7 classification algorithms not only based on accuracy but also by analyzing the critical errors which in term gave us a far accurate and a real time usable model which bears a really low amount of risk.

5.3 Thesis Limitation

First of all we think our thesis largely depends on the dataset we collected from UCI which only contains the people of India . But as we know probability also differs from region to region . Like in underdeveloped countries more people are to be affected due to lack of hygiene and also lack of proper medical care. So there parameters values might differ due to these cases while predicting if someone is actually affected with CKD or not.

5.4 Future Work

Our project is mainly for kidney related disease .But if someone were to train our model for detecting other complications or diseases it is possible to do so by using our model. One simply has to change the

input dataset and number of parameters according to provided in the documentation for the code. In other words one can use our model for detecting other diseases also. Taking into consideration the ability to branch in any other aspect of medical science we think our model can achieve greater and satisfactory results with better and more refined dataset.

REFERENCES

[1] "Chronic kidney disease," https://www.nhs.uk/conditions/kidney-disease, last accessed: 11 August, 2020.

[2] "Chronic kidney disease symptoms," https://www.mayoclinic.org/diseases-conditions/chronic-kidney-disease/symptoms-causes/syc-20354521, last accessed: 27 September, 2020.

[3] "Nkf kdoqi guidelines," https://kidneyfoundation.cachefly.net/professionals/KDOQI/guidelines_ckd/p4_class_g1.htm, last accessed: 21 June, 2020.

[4] E. F. Carney, "The impact of chronic kidney disease on global health," *Nature Reviews Nephrology*, vol. 16, no. 5, pp. 251–251, 2020.

[5] "Chronic kidney disease stats," https://www.worldlifeexpectancy.com/bangladesh-kidney-disease#:~:text=According%20to%20the%20latest%20WHO,Bangladesh%20%2394%20in%20the%20world., last accessed: 11 August, 2020.

[6] "Chronic kidney disease dataset," https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease, last accessed: 23 May, 2020.

[7] L. J. Rubini and P. Eswaran, "Generating comparative analysis of early stage prediction of chronic kidney disease," *International Journal of Modern Engineering Research (IJMER)*, vol. 5, no. 7, pp. 49–55, 2015.

[8] V. Chaurasia, S. Pal, and B. Tiwari, "Chronic kidney disease: a predictive model using decision tree," *International Journal of Engineering Research and Technology*, 2018.

[9] S. Vijayarani, S. Dhayanand, and M. Phil, "Kidney disease prediction using svm and ann algorithms," *International Journal of Computing and Business Research (IJCBR)*, vol. 6, no. 2, pp. 1–12, 2015.

[10] R. Cuingnet, R. Prevost, D. Lesage, L. D. Cohen, B. Mory, and R. Ardon, "Automatic detection and segmentation of kidneys in 3d ct images using random forests," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2012, pp. 66–74.

[11] P. Sinha and P. Sinha, "Comparative study of chronic kidney disease prediction using knn and svm," *International Journal of Engineering Research and Technology*, vol. 4, no. 12, pp. 608–12, 2015.

[12] J. Qin, L. Chen, Y. Liu, C. Liu, C. Feng, and B. Chen, "A machine learning methodology for diagnosing chronic kidney disease," *IEEE Access*, vol. 8, pp. 20 991–21 002, 2019.

[13] S. Sharma, V. Sharma, and A. Sharma, "Performance based evaluation of various machine learning classification techniques for chronic kidney disease diagnosis," *arXiv preprint arXiv:1606.09581*, 2016.

[14] N. Shetty, N. Prasad, and N. Nalini, *Emerging research in computing, information, communication and applications*. Springer, 2016.

[15] B. Khan, R. Naseem, F. Muhammad, G. Abbas, and S. Kim, “An empirical evaluation of machine learning techniques for chronic kidney disease prophecy,” *IEEE Access*, vol. 8, pp. 55 012–55 022, 2020.

CHAPTER 6
DATASET PREPROCESSING

```
import numpy as np
import pandas as pd

# Read dataset file ckd.csv
dataset = pd.read_csv("CKD.csv",header=0, na_values="?")

# Replace null values "?" by numpy.NaN
dataset.replace("?", np.NaN)

# Convert nominal values to binary values
cleanup = {"Rbc":     {"normal": 1, "abnormal": 0},
           "Pc": {"normal": 1, "abnormal": 0},
           "Pcc": {"present": 1, "notpresent": 0},
           "Ba": {"present": 1, "notpresent": 0},
           "Htn": {"yes": 1, "no": 0},
           "Dm": {"yes": 1, "no": 0},
           "Cad": {"yes": 1, "no": 0},
           "Appet": {"good": 1, "poor": 0},
           "pe": {"yes": 1, "no": 0},
           "Ane": {"yes": 1, "no": 0}}

# Replace binary values into dataset
dataset.replace(cleanup, inplace=True)

# Fill null values with mean value of the respective column

dataset.fillna(round(dataset.mean(),2), inplace=True)

# Save this dataset as final.csv for further prediction
dataset.to_csv("final.csv", sep=',', index=False)
```

CHAPTER 7
MODEL TRAINING

```
from sklearn.neural_network import MLPClassifier
import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn. ensemble import RandomForestClassifier, VotingClassifier
from sklearn. ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import f1_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import mean_absolute_error, precision_recall_fscore_support
from sklearn.metrics import confusion_matrix, accuracy_score

data = pd.read_csv("final.csv")

x = data.drop('Class', axis = 1)
y = data['Class']

min = [0, 0, 1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2000, 0, 0,
0, 0, 0, 0, 0]
max = [120, 360, 1.1, 5.5, 5.5, 1, 1, 1, 1, 700, 500, 100, 200,
100, 50, 100, 30000, 20, 1, 1, 1, 1, 1, 1]

i = 0
for col in x.columns:
    x[col] = np.array(x[col].apply(lambda x:(x - min[i]) / (max[i] - min[i])),
    dtype=np.float32)
    i += 1


```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

lr = LogisticRegression(max_iter=1000)
dt = DecisionTreeClassifier()
svm = SVC(kernel = 'poly', degree = 24)
RF = RandomForestClassifier(n_estimators=1000)
XB = GradientBoostingClassifier(n_estimators=1000, learning_rate=0.7,
max_features=24, max_depth=1000)
CNN = MLPClassifier(max_iter=1000, solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(24, 24))
NB = GaussianNB()


evc = VotingClassifier(estimators= [('lr',lr),('dt',dt),('svm',svm),('RF',RF),
('XB',XB),('CNN',CNN),('NB',NB)],voting = 'hard')


evc.fit(x_train,y_train)
evc_pred = evc.predict(x_test)

acc=evc.score(x_test, y_test)
print(acc)

with open("evc.pickle","wb") as f:
        pickle.dump(evc,f)
```

CHAPTER 8
WEBSITE FLASK BACKEND

```
from flask import Flask, render_template, request, url_for, redirect
import pickle
import io
import csv

app = Flask(__name__)

model = pickle.load(open('evc.pickle', 'rb'))

names = ["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",
"Hemoglobin", "Packed Cell Volume", "White Blood Cell Count",
"Red Blood Cell Count", "Hypertension", "Diabetes Mellitus",
"Coronary Artery Disease", "Appetite", "Pedal Edema", "Anemia"]

min = [0, 0, 1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2000,
0, 0, 0, 0, 0, 0, 0]
max = [120, 360, 1.1, 5.5, 5.5, 1, 1, 1, 1, 700, 500, 100, 200,
100, 50, 100, 30000, 20, 1, 1, 1, 1, 1, 1]

@app.route("/upload", methods=["POST", "GET"])
def file():
        if request.method == "POST":
                if request.files:
                        f = request.files['data_file']

                        # print(f)
                        if (f.filename[-3:] != "csv"):
                                return render_template("index.html",
                                text=["file should be in a CSV format"])

                        if f:
                                stream = io.StringIO(f.stream.read().
                                decode("UTF8"), newline=None)
```

```
                                data = list(csv.reader(stream))

                                if (len(data[0]) != 24):
                                        return render_template("index.html",
                                        text=["file should have above 24
                                        information in left to right & then top
                                        to bottom order"])

                                for i in range(24):
                                        data[0][i] = float((float(data[0][i]) -
                                        min[i]) / (max[i] - min[i]))

                                res = model.predict(data)[0]
                                return redirect(url_for("result", res=res))
                else:
                        return "no files added"
        else:
                return render_template("index.html", text=[""])

@app.route("/", methods=["POST", "GET"])
def home():
        data = []
        if request.method == "POST":
                for n in names:
                        data.append(float(request.form.get(n)))

                for i in range(24):
                        data[i] = (data[i] - min[i]) / (max[i] - min[i])

                res = model.predict([data])[0]
                return redirect(url_for("result", res=res))
        else:
                return render_template("index.html", text=[""])

@app.route("/<res>")
def result(res):
        print("res : " + res)
        if res == '1':
                return render_template("result.html",
                text=["Your data indicates  there is high probablity",
                " that your kidney is suffering."])
        else:
                return render_template("result.html",
```

```
79                 text=["According␣to␣your␣data␣",
80                 "your␣kidney␣is␣safe␣and␣sound"])
81
82 if __name__ == "__main__":
83         app.run(debug=True)
```