Faculty of Computers
&Artificial Intelligence

Benha University

Predicting a disease based on the diagnosis of another disease using machine learning

A senior project submitted in partial fulfillment of the requirements for the degree of Bachelor of Computers and Artificial Intelligence.

Scientific Computing Departement,

Project Team

1- Yasmine Emad Hamissa
2- Doaa Maher Shahat Ahmed
3- Fadia alazab Mohamed elgendi
4- Ahmed Mohamed Mohamed salah Mohamed
5- Ahmed Mahmoud Ahmed Labib
6- Mahmoud Saeed Mustafa Lashin
7- Ahmed Mohie Abd el-Azem Younis

Under Supervision of

Dr. Ahmed Hagag

Benha, July 2021

ACKNOWLEDGMENT

ALLAH is the first and the last to be thanked. We want to thank all those who helped us by their knowledge and experience. We will always appreciate their efforts. First and foremost, we would like to express a sincere gratitude to our mentor and advisor Dr. Ahmed Hagag for his advice and support while working on this project. We deeply appreciate his valuable advice. Next, we want also to acknowledge all the support we have received form our faculty “Faculty of Computers & Artificial Intelligence” while working on this project. Finally, it has been a pleasure knowing everyone we worked with during that period: all the people from our department and from the other departments with whom we shared so many good experiences.

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in *Scientific Computing* is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: ______________________________

Date: Tuesday, 13July2021.

ABSTRACT

Disease classification/detection is a crucial and challenging problem, because it helps in early diagnosis of disease by supporting the pathologists and doctors in their decision. Machine Learning technique is one of the emerging field can be used in the health sectors for the diagnosis of different diseases. This paper presents an effective approach for the diagnosis of chronic kidney disease (CKD) using artificial neural network (ANN) with back propagation algorithm, where first we fill the missing values of the dataset using mean, mode and median of attributes. Further, we have trained the NN classifier and evaluate the detection performances on separate test dataset. From the comparative analysis with other variants of classifiers like logistic regression, Naive Bayes, Decision tree, Random forest, K-NN and support vector machine (SVM), it is found that the recognition accuracy of Random forest is significantly encouraging.

Contents

LIST OF FIGURES

C h a p t e r   O n e

1 INTRODUCTION

1.1 PROJECT OVERVIEW

Disease classification/detection is a crucial and challenging problem, because it helps in early diagnosis of disease by supporting the pathologists and doctors in their decision. Machine Learning technique is one of the emerging field can be used in the health sectors for the diagnosis of different diseases.

Machine learning can be used as an informative tool to extract the useful information which helps pathologists and doctors of decisions making. Today's some researchers are working on CKD by applying different computational techniques for the prediction and diagnosis of this disease. As well as predicting other diseases, such as heart failure, and this is what this project try to achieve by finding the relations between chronic kidney disease and heart failure disease for early disease prediction.

1.2 PURPOSE

Heart related diseases and chronic kidney disease (CKD) are the main reason for a huge number of deaths in the world over the last few decades and has emerged as the most life-threatening disease. So, there is a need of reliable, accurate and feasible system to diagnose such diseases in time for proper treatment. Machine Learning algorithms and techniques have been applied to various medical datasets to automate the analysis of large and complex data. Many researchers, in recent times, have been using several machine learning techniques to help the health care industry and the professionals in the diagnosis of heart related diseases.

1.3 PROBLEM STATEMENT

Chronic kidney disease (CKD) is a risk factor for developing heart failure (HF). CKD and HF share common risk factors, but few data exist on the prevalence, signs and symptoms as well as correlates of HF in populations with CKD of moderate severity.

CKD is the lasting damage to kidney that can get worse over the time. If the kidney damage up to last stage than it may stop working. Generally people suffer with this disease with their age, but recently from 5 years children and youth also suffering from CKD disease. The main task of the kidney is to filter out waste products and excess fluid from body which is then passed out through urine. But in CKD kidney lose their functionality due to which some excess amount of urine mix with blood and also some protein mix with urine. 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. From last 15 years data it has been noticed that increase in number of patients which are suffering from CKD disease. More than 60% patients not receiving medical attention. Therefore, early diagnosis and detection of this disease can help the patients in recovery on the right time.

CVD is often under diagnosed and undertreated in patients with chronic kidney disease.

Heart failure (HF) is one of the prime cardiovascular conditions in patients with impaired renal function.

1.4 MOTIVATION & OBJECTIVE

The motivation that drove us to do this project is in order to estimate whether patients with moderate CKD already show signs and symptoms of HF found in its early stages.

1.5 PROJECT DESCRIPTION

1.5.1 Dataset description

1. Title: Early stage of Indians Chronic Kidney Disease (CKD)
2. Source Information:

(a) Source:

Dr.P.Soundarapandian.M.D.,D.M
(Senior Consultant Nephrologist),
Apollo Hospitals,
Managiri,
Madurai Main Road,
Karaikudi,
Tamilnadu,
India.

(b) Creator:

L.Jerlin Rubini (Research Scholar)
Alagappa University
EmailId :jel.jerlin@gmail.com
Contact No :+91-9597231281

(c) Guided by:

Dr.P.Eswaran Assistant Professor,
Department of Computer Science and Engineering,
Alagappa University,
Karaikudi,
Tamilnadu,
India.
Emailid:eswaranperumal@gmail.com

(d) Date: July 2015

But we modify it by adding three new attributes (ferritin, Iron, eGFR equation) to apply our project idea

3. Relevant Information:

age	-	age
bp	-	blood pressure
sg	-	specific gravity
al	-	albumin
su	-	sugar
rbc	-	red blood cells
pc	-	pus cell
pcc	-	pus cell clumps
ba	-	bacteria
bgr	-	blood glucose random
bu	-	blood urea
sc	-	serum creatinine
sod	-	sodium
pot	-	potassium
hemo	-	hemoglobin
pcv	-	packed cell volume
wc	-	white blood cell count
rc	-	red blood cell count
htn	-	hypertension
dm	-	diabetes mellitus
appet	-	appetite
pe	-	pedal edema
ane	-	anemia
eGFR	-	eGFR equation
fer	-	ferritin
ir	-	iron
class	-	class

And we make same modification on it as make it balanced by using SMOTE method and using and make duplication on it. And we apply classification on three Version of dataset.

1.5.2 Machine learning overview

Machine learning, a branch of intelligence, knowing information, identifying patterns, and accordingly making possible human intervention decisions, machine learning makes computers go into self-learning without the need for explicit programming, when fed with new data.

It allows software applications to become more accurate at predicting outcomes without explicitly programming them. The main focus of machine learning is building algorithms that can receive input data, and using statistical analysis; to predict outputs within an acceptable range.

The machine learns intelligent insights through the advanced use of learning algorithms. The machine is also trained to learn patterns from the data, and then it can move forward independently on new and changing data, and then create a dynamic feedback loop, allowing it to efficiently generate more models to obtain More ideas, even more accurately, without the need for additional resources or human interaction Machines are recovering increasingly self-organizing, self-engineering, resulting in greater value production for companies, and machine learning helps solve many problems in computer vision applications, Such as face recognition technology and speech recognition.

We use machine learning classifiers in our project.

Classifiers:

Logistic Regression: LR

Naive Bayes: NB

Decision Tree: DT

Random Forest Classifier: RFC

K-Nearest Neighbour: KNN

Support Vector Machine: SVM

1.6 Application Phases

Our web application has three phases:

- Phase one: the user fills the page of diagnosis
- Phase two: The application sends the diagnosis to be deal with by the model
- Phase three: The application receives the result and displays it on the result page.

1.7 Organization

The rest of this thesis is structured as follows. Chapter 2 illustrates the background. Chapter 3 illustrates the Methodology and the system analysis of the project. TheSystem design & Implementation is discussed in Chapter 4, including the description of both the hardware and software applications. The source codes are presented in Chapter 5. Finally, the conclusions are drawn in Chapter 6.

C h a p t e r T w o

2 BACKGROUND

2.1 Survey

Chronic kidney disease (CKD) is a risk factor for developing heart failure (HF). CKD and HF share common risk factors, but few data exist on the prevalence, signs and symptoms as well as correlates of HF in populations with CKD of moderate severity.

The large amount of data is a key resource to be processed and analyzed for knowledge extraction and enabling support for cost-savings and decision-making.

Coronary heart disease (CHD) is considered a deadly illness that causes death to over a million patients every year. Nearly half the patients diagnosed with CAD will eventually die from the disease.

335,000 of CHD patients will die of a heart attack in an emergency department or before they even reach the hospital. According to the American Heart Association, over 7 million Americans have suffered a heart attack in their lifetime.

When plaque is built inside the coronary arteries, they narrow these arteries making them unable to carry oxygenated blood to the heart muscle causing the well-known symptoms of CAD such as chest pain (angina) and shortness of breath.

Chronic kidney disease (CKD) is a main public health problem affecting more than 10% of the public population in many countries worldwide.

If the kidney damage up to last stage then it may stop working. Mostly people suffer with this disease with their age, but recently from 5 years children and youth also suffering from CKD disease.

The main task of the kidney is to filter out waste products and excess fluid from body which is then passed out through urine But in CKD kidney lose their functionality due to which some excess amount of urine mix with blood and also some protein mix with urine.

There are some symptoms which shows kidneys are beginning to fail like 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, appetite, anemia, iron, ferritin.

Trouble catching your breath, trouble sleeping, fever and vomiting. Early diagnosis and detection of this disease can help the patients in recovery on the right time.

It is associated with high cardiovascular disease (CVD) morbidity and mortality.

Table of survey:

Source	Publish year	Abstract	Conclusion
Detection of Chronic Kidney Disease Using Artificial Neural network	10/2019	**Abstract:** This paper presents an effective approach for the diagnosis of chronic kidney disease(CKD) using artificial neural network (ANN) Further, training the NN classifier and evaluate the detection performances on separate test dataset From the comparative analysis with other variants of classifiers like SVM, K-NN، Classification and Regression tree it is found that the recognition accuracy of ANN is significantly encouraging **Applied Techniques and Accuracy:**	After the experimental analysis it was found that the classificatio n and detection accuracy of mean, mode and median based pre- processing techniques with neural network was significantly encouraging than K-NN, SVM Regression Tree and Classificatio n Tree. Therefore، we can used this framework for the better prediction of chronic kidney disease.

		KNN (72.90) SVM (95.49) Classification Tree (99.09) Regression Tree (95.49) ANN (99.19) **Data selection:** training data (72%), Testing data (28%) **Features:** using only 18 important Attributes from 24 attributes	
Heart disease prediction using machine learning techniques : a survey	2/8/2018	Abstract: This paper presents a survey of various models based on such algorithms and techniques and analyze their performance. Models based on supervised learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbor (KNN), NaiveBayes, Decision Trees (DT), Random Forest (RF) and ensemble models are found very popular among the researchers. Algorithms and Techniques Used: Naive Bayes: has achieved an accuracy of	Each of the algorithms have performed extremely well in some cases but poorly in some other cases. Alternating decision trees when used with PCA, have performed extremely well but decision trees have performed very poorly in some other cases which could be due to overfitting. Random Forest and Ensemble models have performed very well because they

		(84.1584%) with the 10 most significant features, has achieved an accuracy of (83.49%) when all 13 attributes of the Cleveland dataset. Support Vector Machine: Achieve an accuracy of (98.9%) K – Nearest Neighbor: gives an accuracy of (83.16%) when the value of k is equal to 9 while using 10-cross validation technique, Ridhi Saini and others have obtained an efficiency of (87.5%) which is very good Decision Tree: Decision tree has the worst performance with an accuracy of (77.55%) but when decision tree is used with boosting technique it performs better with an accuracy of (82.17%) using alternating decision trees with principle component analysis to obtain	solve the problem of over fitting by employing multiple algorithms Models based on Naïve Bayes classifier were computation ally very fast and have also performed well. SVM performed extremely well for most of the cases. A lot of research can also be done on the correct ensemble of algorithms to use for a particular type of data.

		accuracy (92.2%) Random Forest: Has a significantly higher accuracy of (91.6%) than all the other methods. In People's Hospital dataset, it achieves an accuracy of (97%) random forest is used to predict coronary heart disease and it obtains an accuracy of (97.7%) Ensemble Model: Have used an ensemble of SVM, KNN and ANN to achieve an accuracy of (94.12%).	
Chronic Kidney Disease Prediction Using Machine Learning	April, 2018	Abstract: In this paper, some machine learning techniques for predicting the chronic kidney disease using clinical data. using three machine learning algorithms such as Decision Tree(DT) algorithm, Naive Bayesian (NB) algorithm. The performance of the above models are compared with each other in order to select	The performance of Decision tree method was found to be 99.25% accurate compared to naive Bayes method. Classification algorithm on chronic kidney disease dataset the performance was obtained as 99.33% Specificity and 99.20%

		the best classifier in Predicting the chronic kidney disease for given dataset. DATASET: Has 25 attributes, 11 numeric and 14 nominal. Total 400 instances of the dataset is used for the training to prediction algorithms, out of which 250 has label chronic kidney disease (CKD) and 150 has label non chronic kidney disease (NOTCKD). The attributes in the dataset are age, bp, sg, al, su, bc, pc, pcc, ba, bgr, bu, sc, sod, pot, hemo, pcr, wc, rc, htn, dm, cad, appet, pe, ane, classification. The dataset is divided into two groups, one for training and another for testing. The ratio of training and testing data is (70%) and (30%) respectively. Techniques used: Decision Tree: Accuracy (99.25%) Sensitivity (99.20%)	Sensitivity. We are also further working on enhancing the performance of prediction system accuracy in neural network and deep learning algorithm.

		Specificity (99.33%) Naive Bayes: Accuracy (98.75%) Sensitivity (98%) Specificity 98.75%)	
Applying Machine Learning Techniques for Predicting the Risk of Chronic Kidney Disease	August, 2016	Abstract: Foundation Heart Disease dataset, 600 clinical records collected from a leading Chennai based diabetes research center. Have tested the dataset for classification using Naïve Bayes and Decision tree method. Methods: Naives Bayes Method: Accuracy (86%) Decision Tree Accuracy (91%) features: 13 attributes Data selection: training data (90%), Testing data (10%)	The performance of Decision tree method was found to be 91% accurate compared to naive Bayes Method. Classification algorithm on diabetes dataset performance was obtained as 94% Specificity and 95% Sensitivity. We also found that mining helps to retrieve correlations from attributes which are not direct indicators of the class which we are trying to predict. also further working on enhancing the performance of

			prediction system accuracy in neural network and clustering algorithm data analysis.

2.2 MACHINE LEARNING ADVANTAGE

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-trailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

2.3 MACHINE LEARNING DISADVANTAGE

With all those advantages to its powerfulness and popularity, Machine Learning isn't perfect. The following factors serve to limit it:

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Chapter Three

3 METHODOLOGY

3.1 SYSTEM DEVELOPMENT REQUIREMENTS WE USE IN THE PROJECT:

✓ Software tools:

1) Anaconda platform
2) spyder software program
3) visual studio code
4) pycharm
5) python 3.8
6) flask
7) Django

✓ web Service Tools:

1) HTML
2) CSS
3) Java script

✓ Machine algorithms

1) Logistic Regression: LR
2) Naive Bayes: NB
3) Decision Tree: DT
4) Random Forest Classifier: RFC
5) K-Nearest Neighbor: KNN
6) Support Vector Machine: SVM

3.2 SYSTEM ANALYSIS

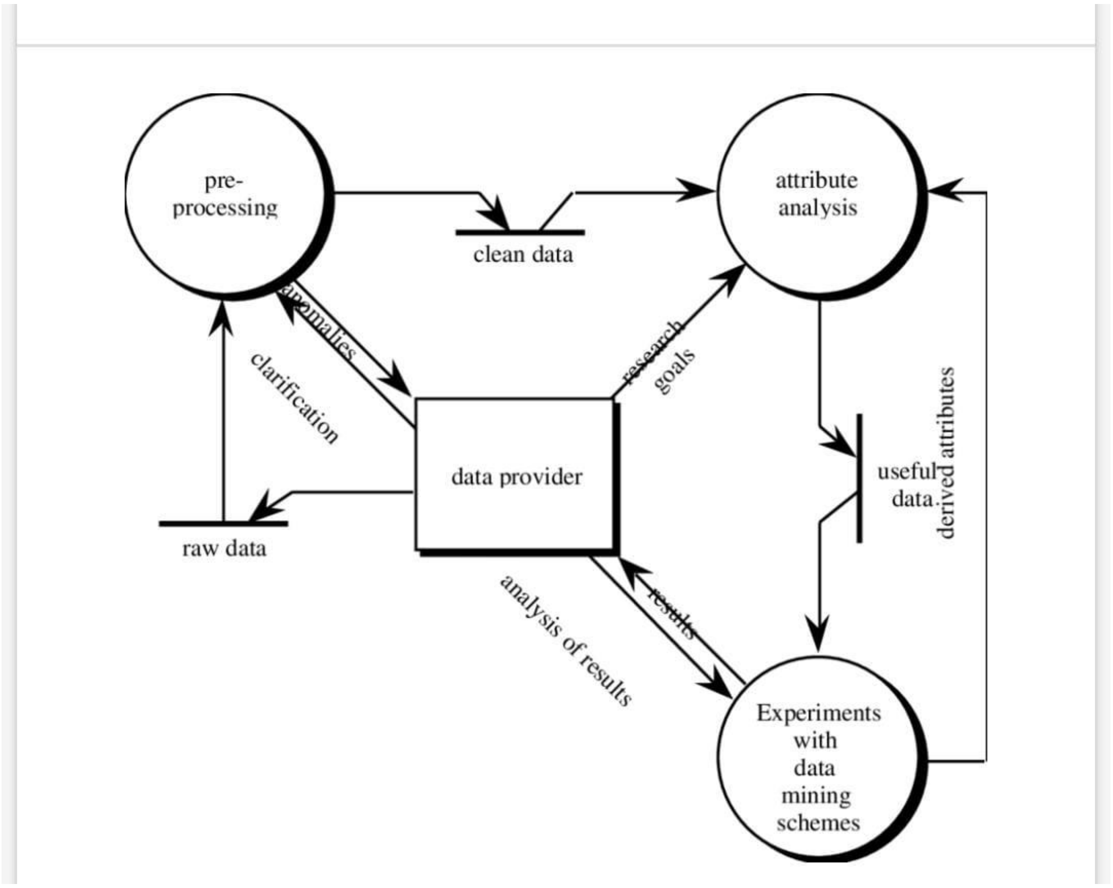


Figure 3-1: Data flow diagram.

3.3 Flow Chart Diagram

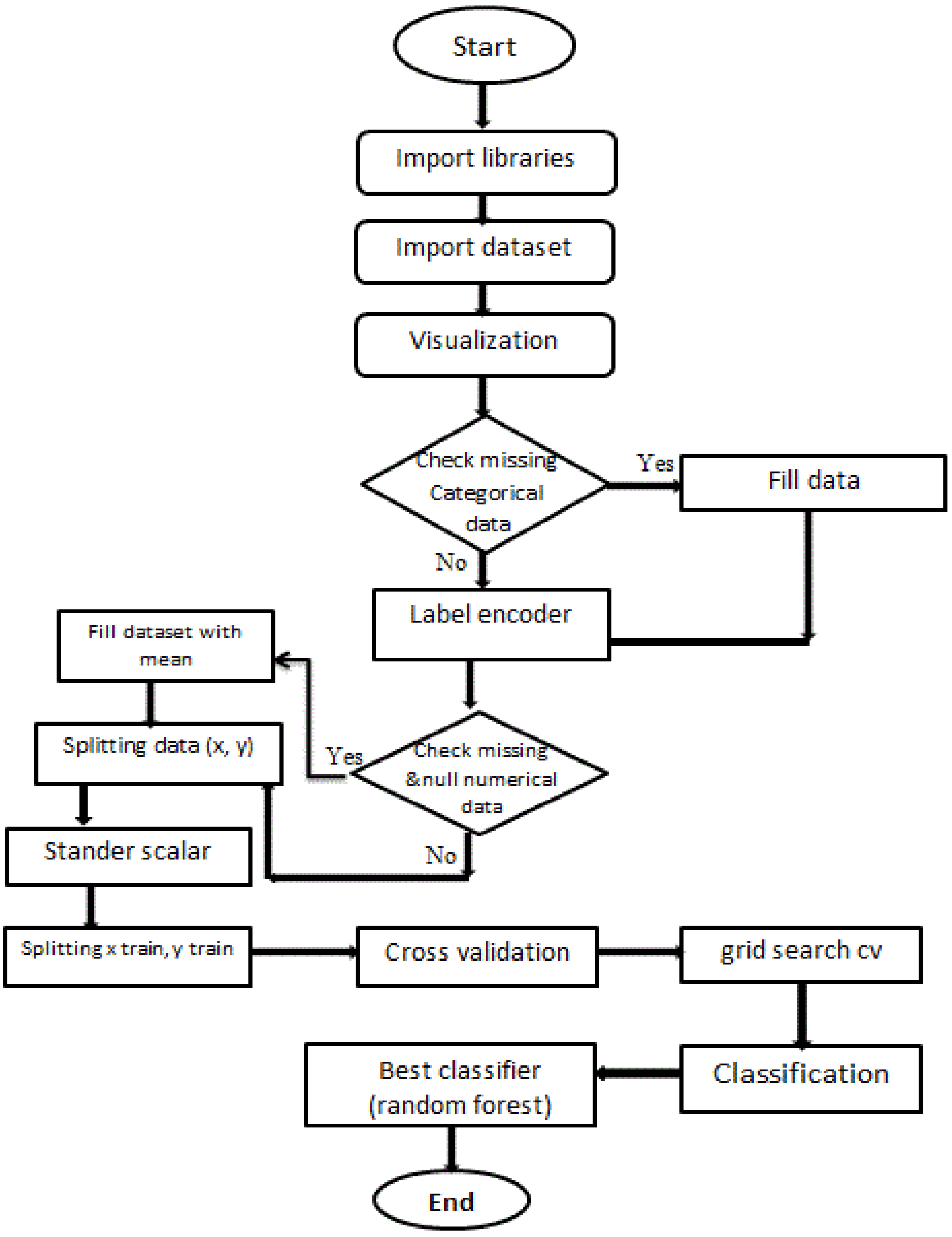


Figure 3-2: Flow chart diagram.

3.4 APPLICATION SYSTEM ANALYSIS

We have the web application program that help patients to record their Attributing, at first by the application we fill the attributes to the Application we have, then we analysis this attributes by machine model and reply by classifiers, after that the application will determine the result from machine model , and shows it in the web Application.

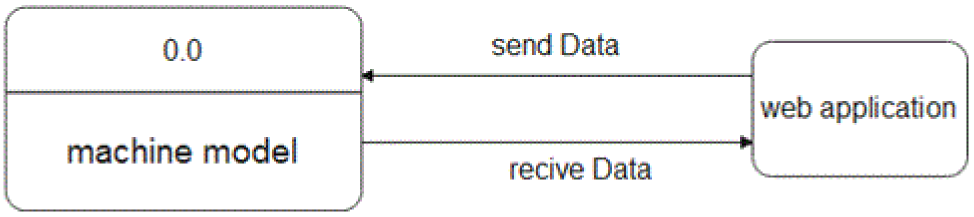


Figure 3-2: Application Analysis.

Chapter Four

4 SYSTEM DESIGN& IMPLEMENTATION

4.1 WEB SERVICE APPLICATION.

Our Application interface consist of three pages (Home, Diagnosis, Result, about us) as shown in Figure 4-1

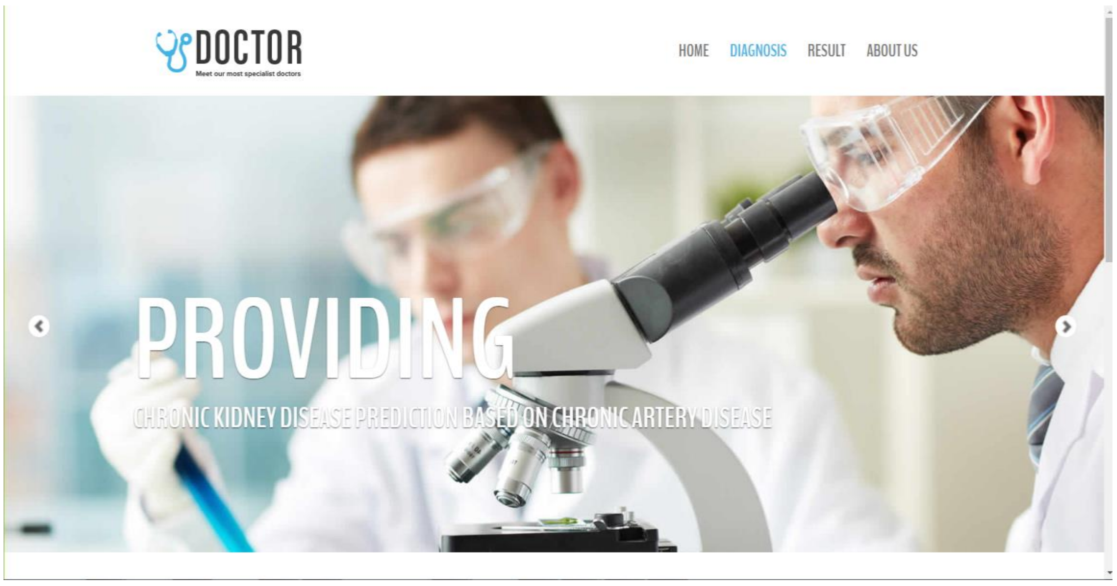
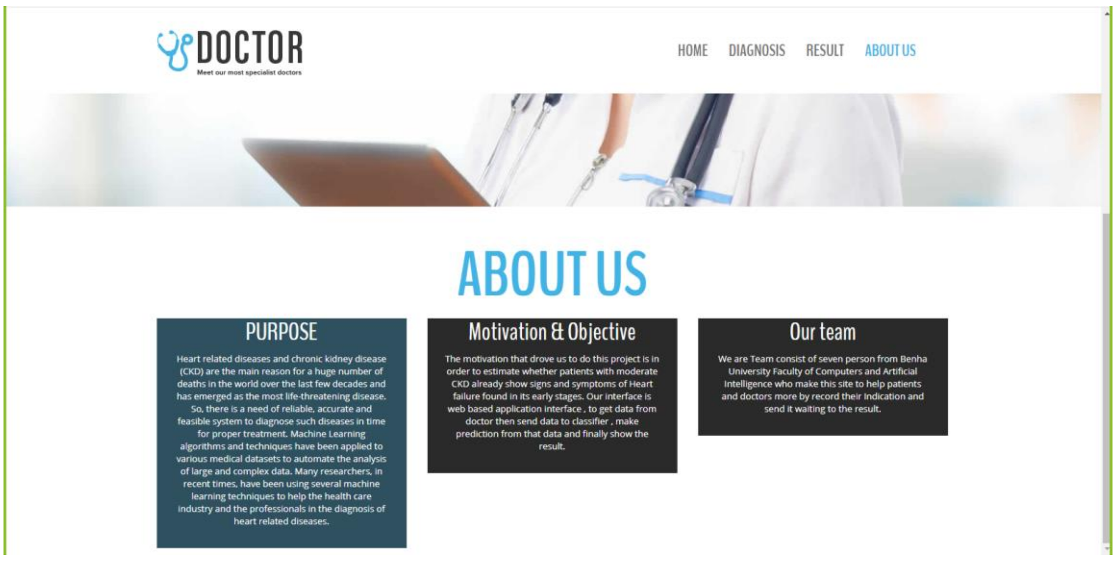

DOCTOR
Meet our most specialist doctors
HOME
DIAGNOSIS
RESULT
ABOUT US
PROVIDING
CHRONIC KIDNEY DISEASE PREDICTION BASED ON CHRONIC ARTERY DISEASE

Figure 4-1: Application interface.

We explain our project idea and our project Team in this page.



ABOUT US

PURPOSE

Heart related diseases and chronic kidney disease (CKD) are the main reason for a huge number of deaths in the world over the last few decades and has emerged as the most life-threatening disease. So, there is a need of reliable, accurate and feasible system to diagnose such diseases in time for proper treatment. Machine Learning algorithms and techniques have been applied to various medical datasets to automate the analysis of large and complex data. Many researchers, in recent times, have been using several machine learning techniques to help the health care industry and the professionals in the diagnosis of heart related diseases.

Motivation & Objective

The motivation that drove us to do this project is in order to estimate whether patients with moderate CKD already show signs and symptoms of Heart failure found in its early stages. Our interface is web based application interface , to get data from doctor then send data to classifier , make prediction from that data and finally show the result.

Our team

We are Team consist of seven person from Benha University Faculty of Computers and Artificial Intelligence who make this site to help patients and doctors more by record their Indication and send it waiting to the result.

Figure 4-2: page of about us.

This page help patients to record their Attributing to diagnosis if there have a chronic kidney disease or heart disease.

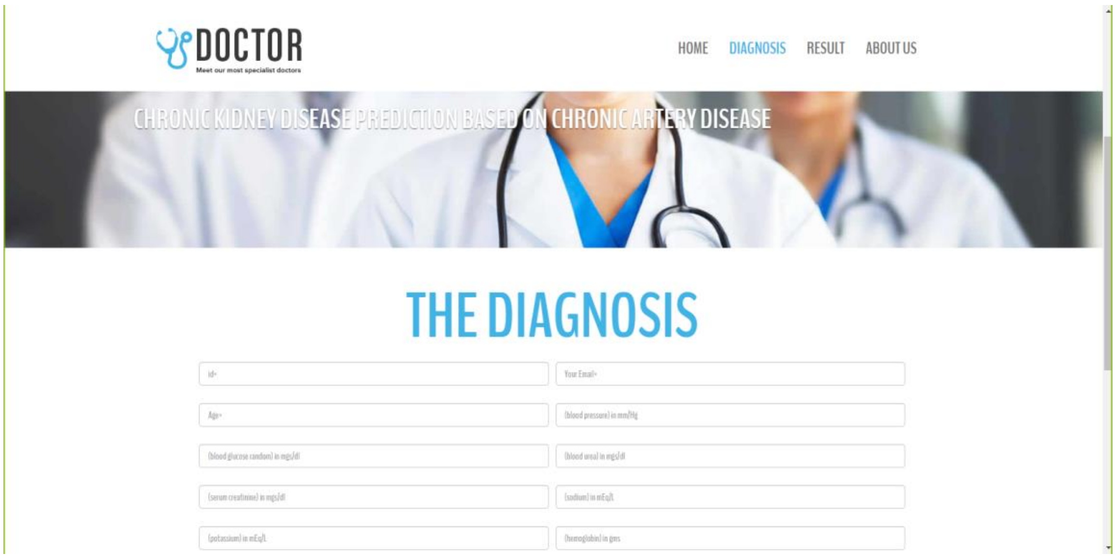

DOCTOR
Meet our most specialist doctors
HOME
DIAGNOSIS
RESULT
ABOUT US
CHRONIC KIDNEY DISEASE PREDICTION BASED ON CHRONIC ARTERY DISEASE
THE DIAGNOSIS
id*
Your Email*
Age*
(blood pressure) in mm/Hg
(blood glucose random) in mgs/dl
(blood urea) in mgs/dl
(serum creatinine) in mgs/dl
(sodium) in mEq/L
(potassium) in mEq/L
(hemoglobin) in gms

Figure 4-3: disease diagnosis 1.

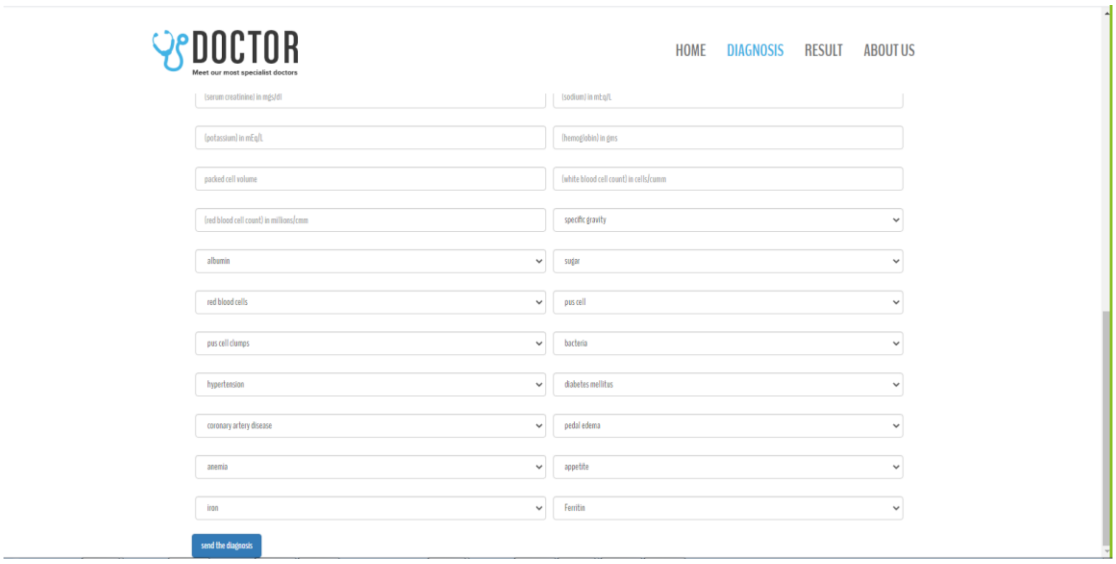


Figure 4-4: disease diagnosis 2.

In this page we get the result if patient have chronic kidney disease or heart disease.

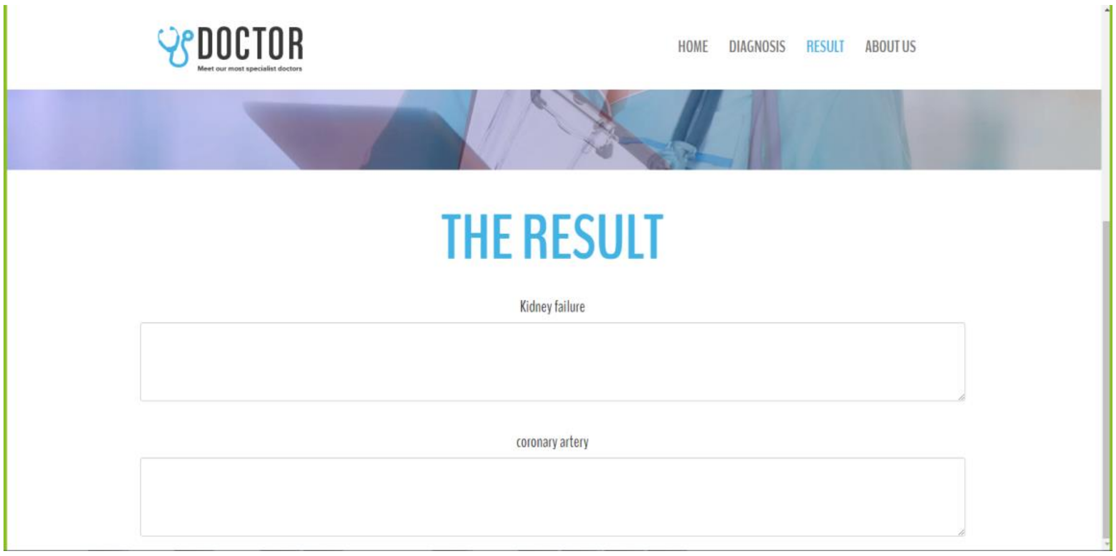


Figure 4-4:The page of the result.

Chapter Five

5 THE SOURCE CODES

5.1 SOURCE CODE OF WEB APPLICATION

5.1.1 Diagnostic part

```
  <!-- about section -->
  <section class="about text-center" id="about">
    <div class="container">
      <div class="row">
        <h2>The Diagnosis</h2>
        <div class="col-12 contact-us text-right">
          <div class="row">
           <div class="col-lg-6 col-sm-12 form-group name">
            <input type="number" placeholder="id*" class="for
m-control " required>
           </div>
           <div class="col-lg-6 col-sm-12 form-group email">
            <input type="text" placeholder="Your Email*" clas
s="form-control " required>
           </div>
          </div>
          <div class="row">
            <div class="col-lg-6 col-sm-12 form-group name">
             <input type="number" min="1" placeholder="Age*"
 class="form-control " required>
            </div>
            <div class="col-lg-6 col-sm-12 form-group name">
              <input type="number" placeholder="(blood pres
sure) in mm/Hg" class="form-control " required>
             </div>
             <div class="col-lg-6 col-sm-12 form-
group name">
              <input type="number" placeholder="(blood gluc
ose random) in mgs/dl" class="form-control " required>
             </div>
             <div class="col-lg-6 col-sm-12 form-
group name">
              <input type="number" placeholder="(blood urea
) in mgs/dl" class="form-control " required>
             </div>
             <div class="col-lg-6 col-sm-12 form-
group name">
```

```
                            <input type="number" placeholder="(serum crea
tinine) in mgs/dl" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="(sodium) in
 mEq/L" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="(potassium)
 in mEq/L" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="(hemoglobin
) in gms" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="packed cell
 volume" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="(white bloo
d cell count) in cells/cumm" class="form-control " required>
                          </div>
                          <div class="col-lg-6 col-sm-12 form-
group name">
                            <input type="number" placeholder="(red blood 
cell count) in millions/cmm" class="form-control " required>
                          </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>specific gravity</option>
                                <option value="">1.005</option>
                                <option value="">1.010</option>
                                <option value="">1.015</option>
                                <option value="">1.020</option>
                                <option value="">1.025</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>albumin</option>
```

```
                                <option value="">0</option>
                                <option value="">1</option>
                                <option value="">2</option>
                                <option value="">3</option>
                                <option value="">4</option>
                                <option value="">5</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>sugar</option>
                                <option value="">0</option>
                                <option value="">1</option>
                                <option value="">2</option>
                                <option value="">3</option>
                                <option value="">4</option>
                                <option value="">5</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>red blood cells</option>
                                <option value="">normal</option>
                                <option value="">abnormal</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>pus cell</option>
                                <option value="">normal</option>
                                <option value="">abnormal</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>pus cell clumps</option>
                                <option value="">present</option>
                                <option value="">notpresent</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>bacteria</option>
```

```
                                <option value="">present</option>
                                <option value="">notpresent</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>hypertension</option>
                                <option value="">yes</option>
                                <option value="">no</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>diabetes mellitus</option>
                                <option value="">yes</option>
                                <option value="">no</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>coronary artery disease</option>
                                <option value="">yes</option>
                                <option value="">no</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>pedal edema</option>
                                <option value="">yes</option>
                                <option value="">no</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>anemia</option>
                                <option value="">yes</option>
                                <option value="">no</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>appetite</option>
```

```
                                <option value="">good</option>
                                <option value="">poor</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>iron</option>
                                <option value="">low</option>
                                <option value="">normal</option>
                            </select>
                        </div>
                        <div class="col-lg-6 col-sm-12 form-group email">
                            <select name="" id="" class="form-control">
                                <option selected="selected" hidden disabl
ed>Ferritin</option>
                                <option value="">low</option>
                                <option value="">normal</option>
                            </select>
                        </div>

                        <div class="col-sm-1 maxm">
                            <button class="btn btn-
primary ">send the diagnosis</button>
                        </div>
                      </div>
                </div>

            </div>
        </div>
    </section><!-- end of about section -->
```

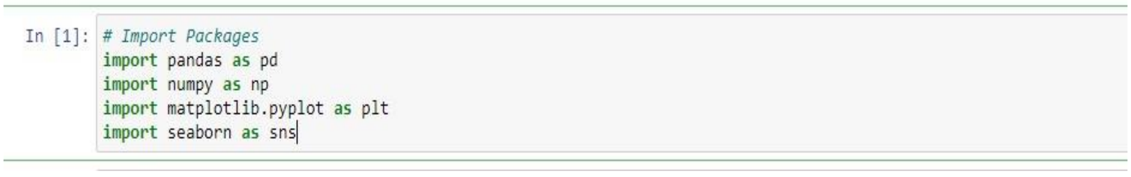
5.1.2 Result part

```
    <!-- about section -->
    <section class="about text-center" id="about">
        <div class="container">
            <div class="row">
                <h2>The Result</h2>
                <div class="row">
                    <div class="col-sm-12 form-group subject">
                        <h3>Kidney failure</h3>
                      <textarea  cols="30" rows="5" class="form-
control p-2"></textarea>
                    </div>
                </div>
                <div class="row">
                    <div class="col-sm-12 form-group subject">
                        <h3>coronary artery</h3>
                       <textarea  cols="30" rows="5" class="form-
control p-2"></textarea>
                    </div>
                </div>
            </div>
        </div>
    </section><!-- end of about section -->
```

5.2 SOURCE CODE OF MACHINE MODELS

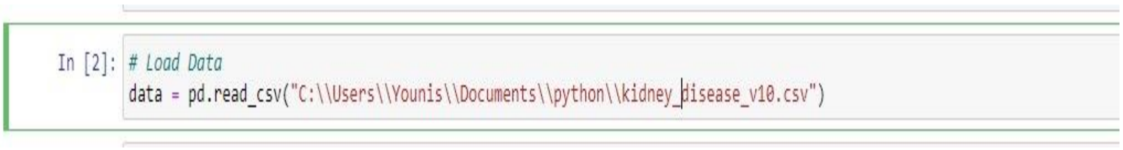
5.2.1 Model 1

Step 1: Importing data



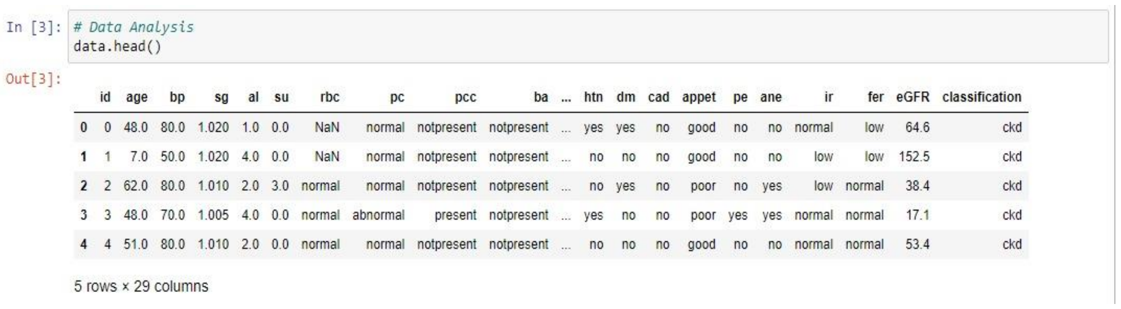
```
In [1]: # Import Packages
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Figure 5-1: Import packages.



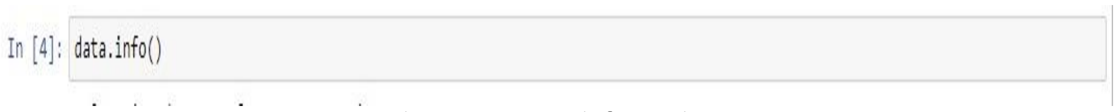
```
In [2]: # Load Data
        data = pd.read_csv("C:\\Users\\Younis\\Documents\\python\\kidney_disease_v10.csv")
```

Figure 5-2: Load data.



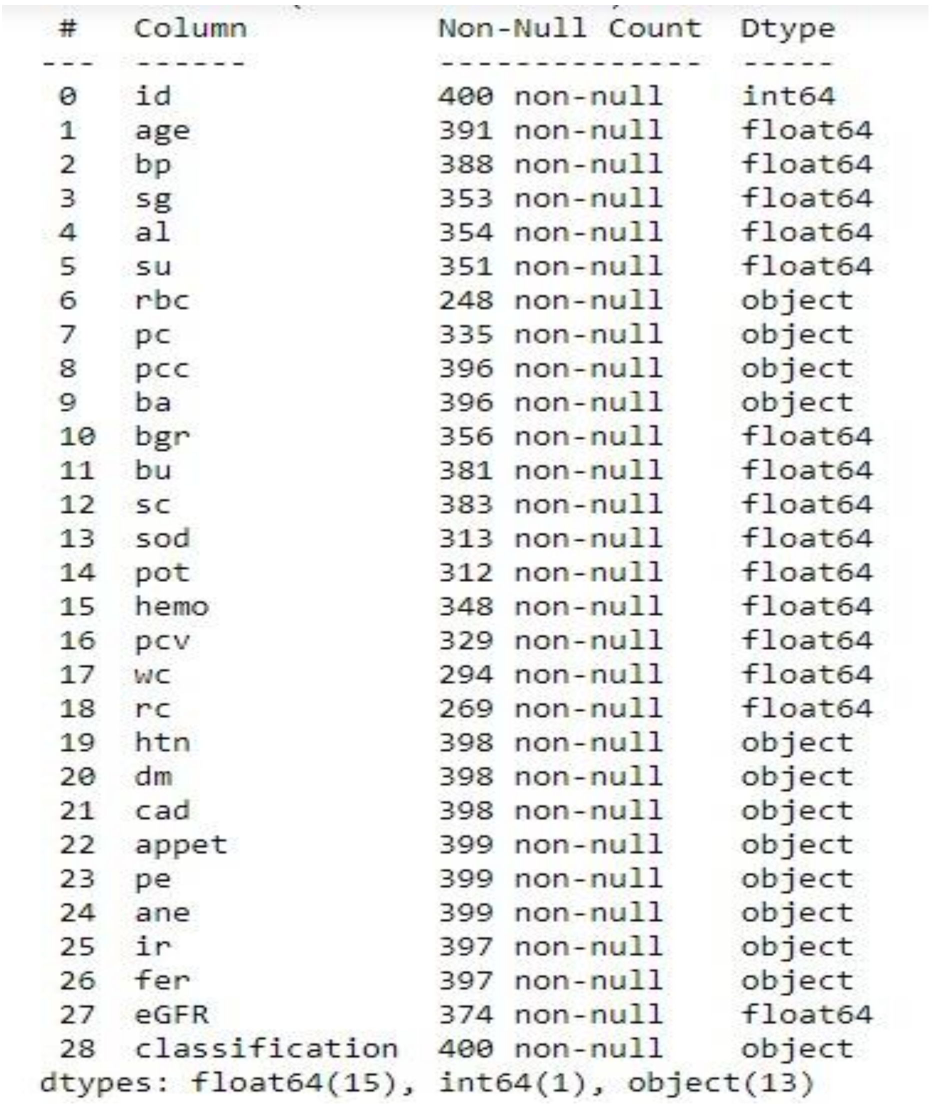
```
In [3]: # Data Analysis
        data.head()

Out[3]:
   id   age    bp     sg   al   su     rbc        pc         pcc          ba  ...  htn  dm  cad  appet   pe  ane      ir     fer  eGFR  classification
0   0  48.0  80.0  1.020  1.0  0.0     NaN    normal  notpresent  notpresent  ...  yes  yes  no   good   no   no  normal     low   64.6             ckd
1   1   7.0  50.0  1.020  4.0  0.0     NaN    normal  notpresent  notpresent  ...   no  no   no   good   no   no     low     low  152.5             ckd
2   2  62.0  80.0  1.010  2.0  3.0  normal    normal  notpresent  notpresent  ...   no  yes  no   poor   no  yes     low  normal   38.4             ckd
3   3  48.0  70.0  1.005  4.0  0.0  normal  abnormal     present  notpresent  ...  yes  no   no   poor  yes  yes  normal  normal   17.1             ckd
4   4  51.0  80.0  1.010  2.0  0.0  normal    normal  notpresent  notpresent  ...   no  no   no   good   no   no  normal  normal   53.4             ckd

5 rows × 29 columns
```

Figure 5-3: Data analysis.



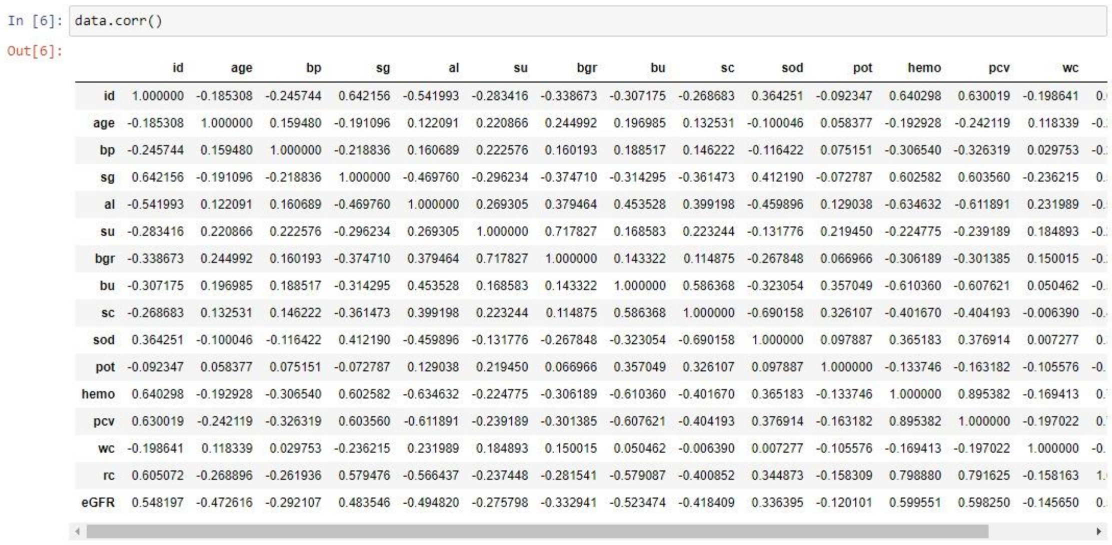
```
In [4]: data.info()
```

Figure 5-4: Data information.



```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    int64
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object
 7   pc              335 non-null    object
 8   pcc             396 non-null    object
 9   ba              396 non-null    object
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             329 non-null    float64
 17  wc              294 non-null    float64
 18  rc              269 non-null    float64
 19  htn             398 non-null    object
 20  dm              398 non-null    object
 21  cad             398 non-null    object
 22  appet           399 non-null    object
 23  pe              399 non-null    object
 24  ane             399 non-null    object
 25  ir              397 non-null    object
 26  fer             397 non-null    object
 27  eGFR            374 non-null    float64
 28  classification  400 non-null    object
dtypes: float64(15), int64(1), object(13)
```

Figure 5-5: Data information output.

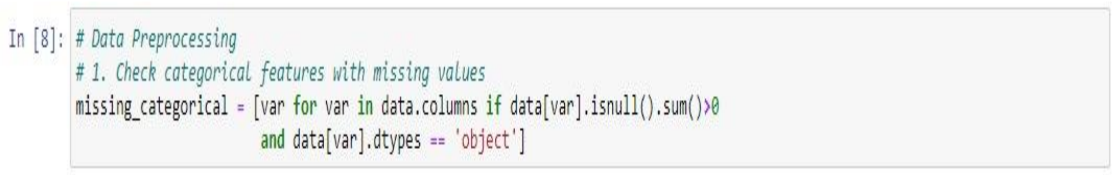


```
In [6]: data.corr()
Out[6]:
```

	id	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	wc	
id	1.000000	-0.185308	-0.245744	0.642156	-0.541993	-0.283416	-0.338673	-0.307175	-0.268683	0.364251	-0.092347	0.640298	0.630019	-0.198641	0.
age	-0.185308	1.000000	0.159480	-0.191096	0.122091	0.220866	0.244992	0.196985	0.132531	-0.100046	0.058377	-0.192928	-0.242119	0.118339	-0.
bp	-0.245744	0.159480	1.000000	-0.218836	0.160689	0.222576	0.160193	0.188517	0.146222	-0.116422	0.075151	-0.306540	-0.326319	0.029753	-0.
sg	0.642156	-0.191096	-0.218836	1.000000	-0.469760	-0.296234	-0.374710	-0.314295	-0.361473	0.412190	-0.072787	0.602582	0.603560	-0.236215	0.
al	-0.541993	0.122091	0.160689	-0.469760	1.000000	0.269305	0.379464	0.453528	0.399198	-0.459896	0.129038	-0.634632	-0.611891	0.231989	-0.
su	-0.283416	0.220866	0.222576	-0.296234	0.269305	1.000000	0.717827	0.168583	0.223244	-0.131776	0.219450	-0.224775	-0.239189	0.184893	-0.
bgr	-0.338673	0.244992	0.160193	-0.374710	0.379464	0.717827	1.000000	0.143322	0.114875	-0.267848	0.066966	-0.306189	-0.301385	0.150015	-0.
bu	-0.307175	0.196985	0.188517	-0.314295	0.453528	0.168583	0.143322	1.000000	0.586368	-0.323054	0.357049	-0.610360	-0.607621	0.050462	-0.
sc	-0.268683	0.132531	0.146222	-0.361473	0.399198	0.223244	0.114875	0.586368	1.000000	-0.690158	0.326107	-0.401670	-0.404193	-0.006390	-0.
sod	0.364251	-0.100046	-0.116422	0.412190	-0.459896	-0.131776	-0.267848	-0.323054	-0.690158	1.000000	0.097887	0.365183	0.376914	0.007277	0.
pot	-0.092347	0.058377	0.075151	-0.072787	0.129038	0.219450	0.066966	0.357049	0.326107	0.097887	1.000000	-0.133746	-0.163182	-0.105576	-0.
hemo	0.640298	-0.192928	-0.306540	0.602582	-0.634632	-0.224775	-0.306189	-0.610360	-0.401670	0.365183	-0.133746	1.000000	0.895382	-0.169413	0.
pcv	0.630019	-0.242119	-0.326319	0.603560	-0.611891	-0.239189	-0.301385	-0.607621	-0.404193	0.376914	-0.163182	0.895382	1.000000	-0.197022	0.
wc	-0.198641	0.118339	0.029753	-0.236215	0.231989	0.184893	0.150015	0.050462	-0.006390	0.007277	-0.105576	-0.169413	-0.197022	1.000000	-0.
rc	0.605072	-0.268896	-0.261936	0.579476	-0.566437	-0.237448	-0.281541	-0.579087	-0.400852	0.344873	-0.158309	0.798880	0.791625	-0.158163	1.
eGFR	0.548197	-0.472616	-0.292107	0.483546	-0.494820	-0.275798	-0.332941	-0.523474	-0.418409	0.336395	-0.120101	0.599551	0.598250	-0.145650	0.

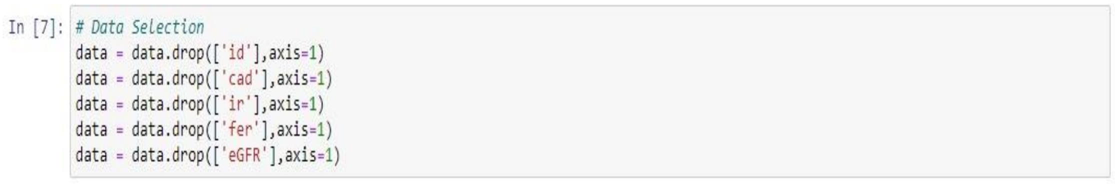
Figure 5-6:Numerical data.

Step 2: Data preprocessing



```
In [8]: # Data Preprocessing
        # 1. Check categorical features with missing values
        missing_categorical = [var for var in data.columns if data[var].isnull().sum()>0
                               and data[var].dtypes == 'object']
```

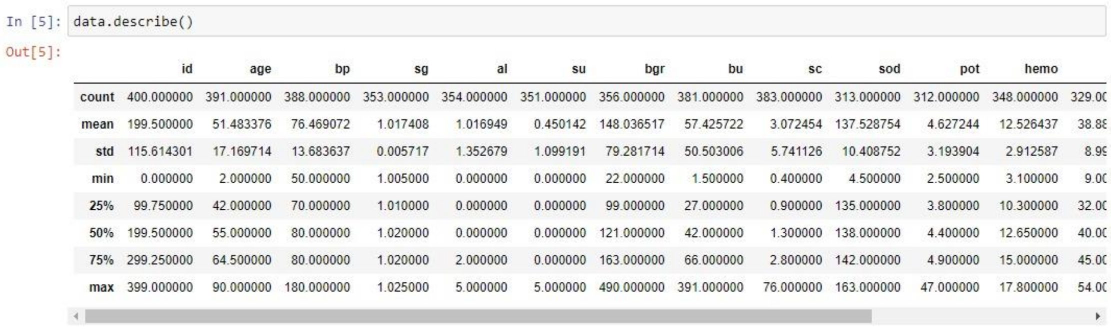
Figure 5-7: Data processing.



```
In [7]: # Data Selection
        data = data.drop(['id'],axis=1)
        data = data.drop(['cad'],axis=1)
        data = data.drop(['ir'],axis=1)
        data = data.drop(['fer'],axis=1)
        data = data.drop(['eGFR'],axis=1)
```

Figure 5-8: Data selection.

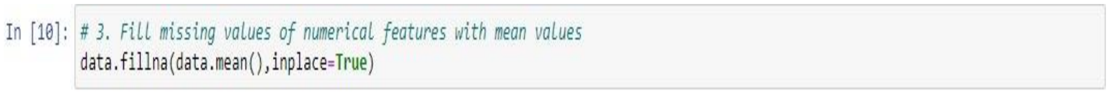
output of prepressing



```
In [5]: data.describe()
Out[5]:
```

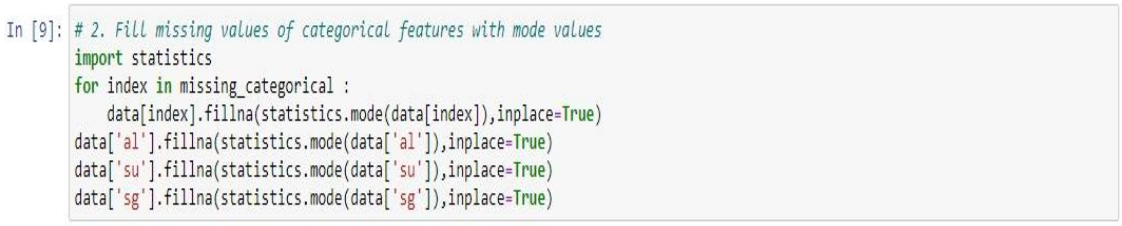
	id	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	
count	400.000000	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000	329.0
mean	199.500000	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437	38.8
std	115.614301	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587	8.9
min	0.000000	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000	9.0
25%	99.750000	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000	32.0
50%	199.500000	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000	40.0
75%	299.250000	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000	45.0
max	399.000000	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000	54.0

Figure 5-9: Output of processing.



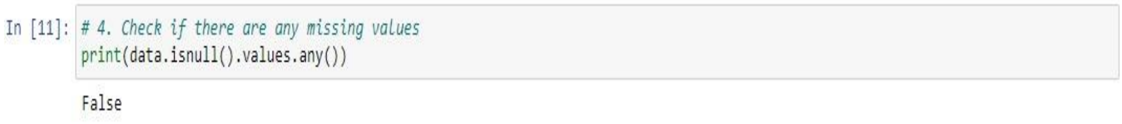
```
In [10]: # 3. Fill missing values of numerical features with mean values
         data.fillna(data.mean(),inplace=True)
```

Figure 5-10: Missing data 1



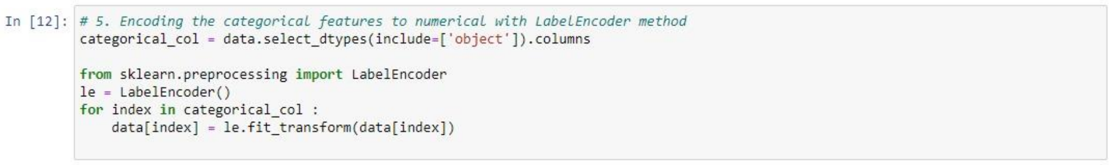
```
In [9]: # 2. Fill missing values of categorical features with mode values
        import statistics
        for index in missing_categorical :
            data[index].fillna(statistics.mode(data[index]),inplace=True)
        data['al'].fillna(statistics.mode(data['al']),inplace=True)
        data['su'].fillna(statistics.mode(data['su']),inplace=True)
        data['sg'].fillna(statistics.mode(data['sg']),inplace=True)
```

Figure 5-11: Missing data 2.



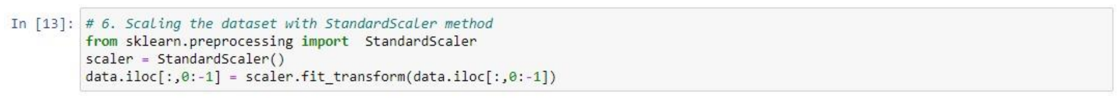
```
In [11]: # 4. Check if there are any missing values
         print(data.isnull().values.any())

         False
```

Figure 5-12: Missing data 3.



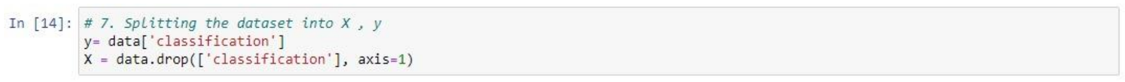
```
In [12]: # 5. Encoding the categorical features to numerical with LabelEncoder method
         categorical_col = data.select_dtypes(include=['object']).columns

         from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         for index in categorical_col :
             data[index] = le.fit_transform(data[index])
```

Figure 5-13: Encoding.



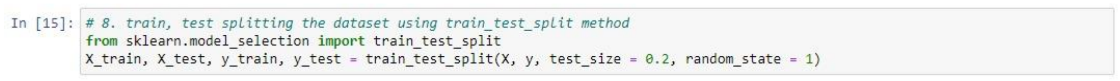
```
In [13]: # 6. Scaling the dataset with StandardScaler method
         from sklearn.preprocessing import  StandardScaler
         scaler = StandardScaler()
         data.iloc[:,0:-1] = scaler.fit_transform(data.iloc[:,0:-1])
```

Figure 5-14: Data scaling.



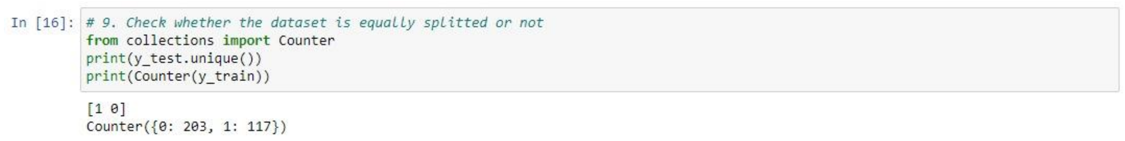
```
In [14]: # 7. Splitting the dataset into X , y
         y= data['classification']
         X = data.drop(['classification'], axis=1)
```

Figure 5-15: Splitting data.



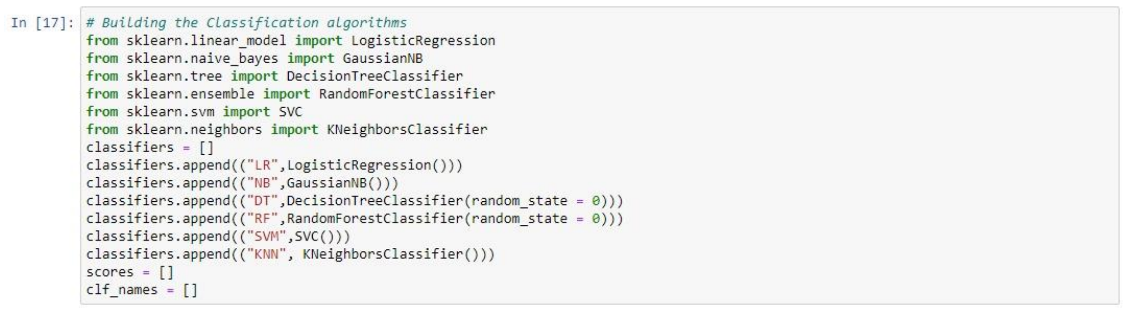
```
In [15]: # 8. train, test splitting the dataset using train_test_split method
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

Figure 5-16: Splitting data 2.



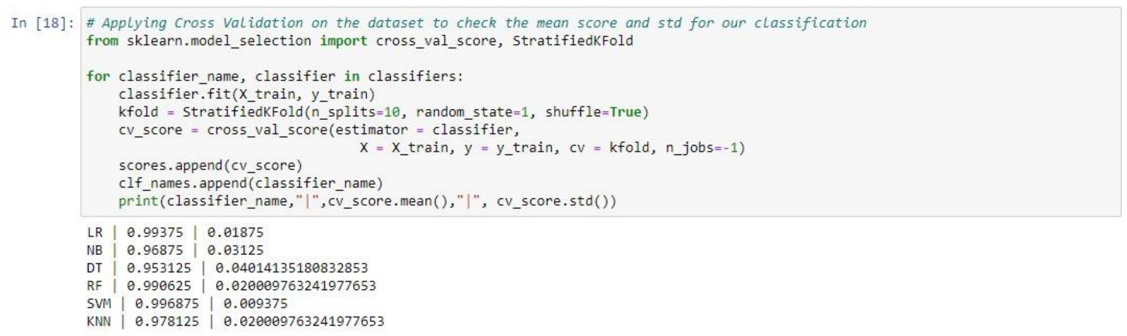
```
In [16]: # 9. Check whether the dataset is equally splitted or not
         from collections import Counter
         print(y_test.unique())
         print(Counter(y_train))

         [1 0]
         Counter({0: 203, 1: 117})
```

Figure 5-17: Splitting data 3.



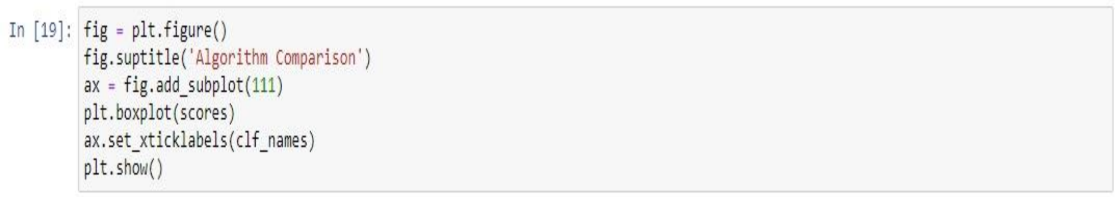
```
In [17]: # Building the Classification algorithms
         from sklearn.linear_model import LogisticRegression
         from sklearn.naive_bayes import GaussianNB
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.svm import SVC
         from sklearn.neighbors import KNeighborsClassifier
         classifiers = []
         classifiers.append(("LR",LogisticRegression()))
         classifiers.append(("NB",GaussianNB()))
         classifiers.append(("DT",DecisionTreeClassifier(random_state = 0)))
         classifiers.append(("RF",RandomForestClassifier(random_state = 0)))
         classifiers.append(("SVM",SVC()))
         classifiers.append(("KNN", KNeighborsClassifier()))
         scores = []
         clf_names = []
```

Figure 5-18: Building classification algorithm.



```
In [18]: # Applying Cross Validation on the dataset to check the mean score and std for our classification
         from sklearn.model_selection import cross_val_score, StratifiedKFold

         for classifier_name, classifier in classifiers:
             classifier.fit(X_train, y_train)
             kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
             cv_score = cross_val_score(estimator = classifier,
                                        X = X_train, y = y_train, cv = kfold, n_jobs=-1)
             scores.append(cv_score)
             clf_names.append(classifier_name)
             print(classifier_name,"|",cv_score.mean(),"|", cv_score.std())

         LR | 0.99375 | 0.01875
         NB | 0.96875 | 0.03125
         DT | 0.953125 | 0.04014135180832853
         RF | 0.990625 | 0.020009763241977653
         SVM | 0.996875 | 0.009375
         KNN | 0.978125 | 0.020009763241977653
```

Figure 5-29: Cross validation.



```
In [19]: fig = plt.figure()
         fig.suptitle('Algorithm Comparison')
         ax = fig.add_subplot(111)
         plt.boxplot(scores)
         ax.set_xticklabels(clf_names)
         plt.show()
```

Figure 5-20: Boxplot code.

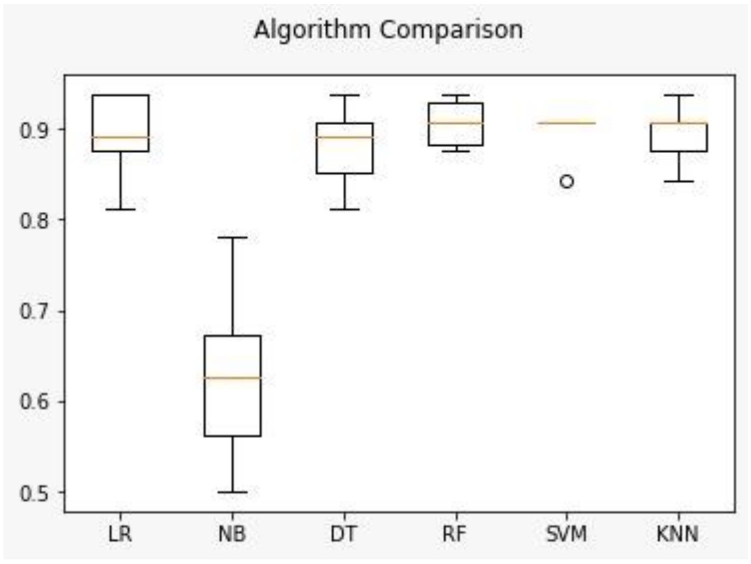
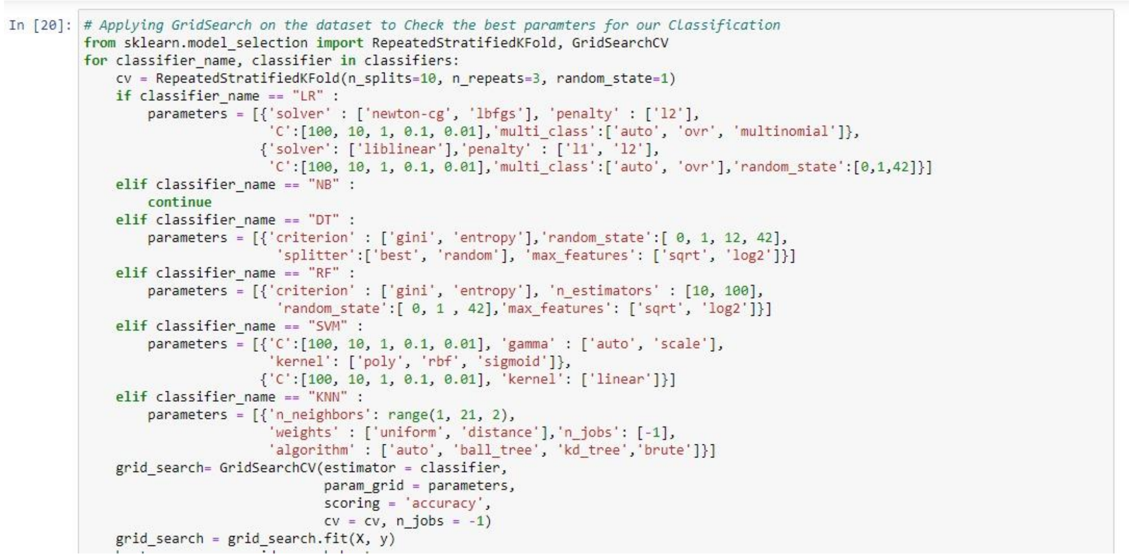
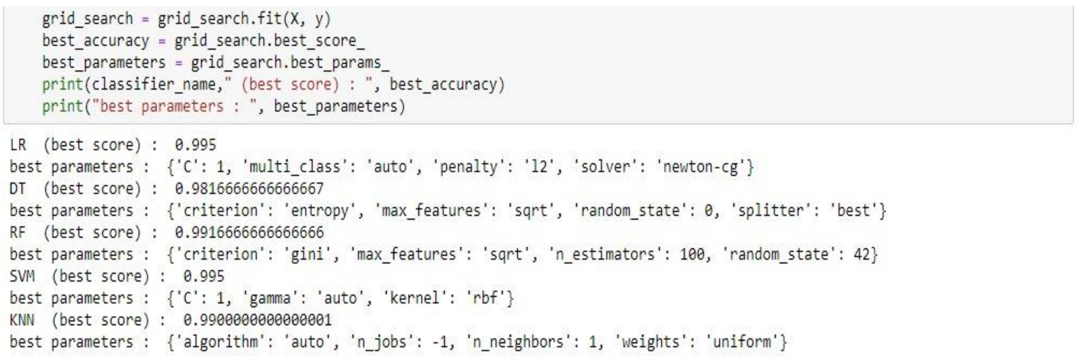


Figure 5-21: Algorithm comparison.



```
In [20]: # Applying GridSearch on the dataset to Check the best paramters for our Classification
         from sklearn.model_selection import RepeatedStratifiedKFold, GridSearchCV
         for classifier_name, classifier in classifiers:
             cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
             if classifier_name == "LR" :
                 parameters = [{'solver' : ['newton-cg', 'lbfgs'], 'penalty' : ['l2'],
                                'C':[100, 10, 1, 0.1, 0.01],'multi_class':['auto', 'ovr', 'multinomial']},
                               {'solver': ['liblinear'],'penalty' : ['l1', 'l2'],
                                'C':[100, 10, 1, 0.1, 0.01],'multi_class':['auto', 'ovr'],'random_state':[0,1,42]}]
             elif classifier_name == "NB" :
                 continue
             elif classifier_name == "DT" :
                 parameters = [{'criterion' : ['gini', 'entropy'],'random_state':[ 0, 1, 12, 42],
                                 'splitter':['best', 'random'], 'max_features': ['sqrt', 'log2']}]
             elif classifier_name == "RF" :
                 parameters = [{'criterion' : ['gini', 'entropy'], 'n_estimators' : [10, 100],
                                 'random_state':[ 0, 1 , 42],'max_features': ['sqrt', 'log2']}]
             elif classifier_name == "SVM" :
                 parameters = [{'C':[100, 10, 1, 0.1, 0.01], 'gamma' : ['auto', 'scale'],
                                'kernel': ['poly', 'rbf', 'sigmoid']},
                               {'C':[100, 10, 1, 0.1, 0.01], 'kernel': ['linear']}]
             elif classifier_name == "KNN" :
                 parameters = [{'n_neighbors': range(1, 21, 2),
                                'weights' : ['uniform', 'distance'],'n_jobs': [-1],
                                'algorithm' : ['auto', 'ball_tree', 'kd_tree','brute']}]
             grid_search= GridSearchCV(estimator = classifier,
                                       param_grid = parameters,
                                       scoring = 'accuracy',
                                       cv = cv, n_jobs = -1)
             grid_search = grid_search.fit(X, y)
```

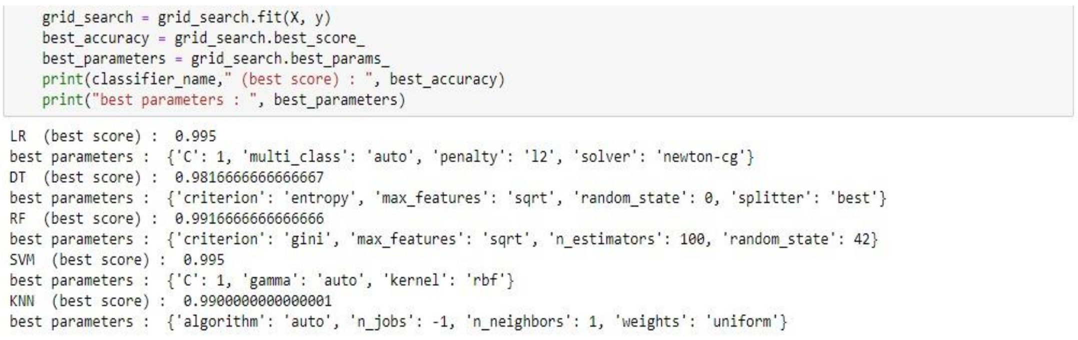
Figure 5-22: Grid search.



```
    grid_search = grid_search.fit(X, y)
    best_accuracy = grid_search.best_score_
    best_parameters = grid_search.best_params_
    print(classifier_name," (best score) : ", best_accuracy)
    print("best parameters : ", best_parameters)
```

```
LR  (best score) :  0.995
best parameters :  {'C': 1, 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'newton-cg'}
DT  (best score) :  0.9816666666666667
best parameters :  {'criterion': 'entropy', 'max_features': 'sqrt', 'random_state': 0, 'splitter': 'best'}
RF  (best score) :  0.9916666666666666
best parameters :  {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 100, 'random_state': 42}
SVM  (best score) :  0.995
best parameters :  {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
KNN  (best score) :  0.9900000000000001
best parameters :  {'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 1, 'weights': 'uniform'}
```

Figure 5-23: Best parameters.

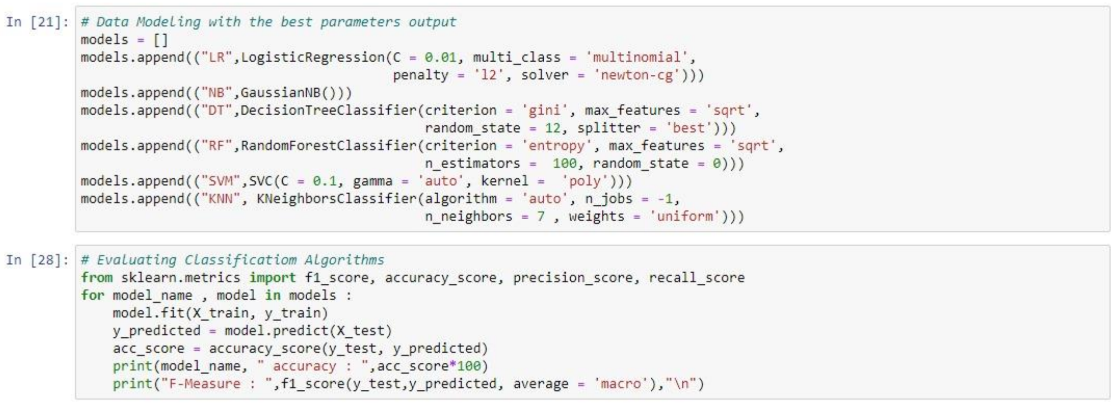


```
    grid_search = grid_search.fit(X, y)
    best_accuracy = grid_search.best_score_
    best_parameters = grid_search.best_params_
    print(classifier_name," (best score) : ", best_accuracy)
    print("best parameters : ", best_parameters)
```

```
LR  (best score) :  0.995
best parameters :  {'C': 1, 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'newton-cg'}
DT  (best score) :  0.9816666666666667
best parameters :  {'criterion': 'entropy', 'max_features': 'sqrt', 'random_state': 0, 'splitter': 'best'}
RF  (best score) :  0.9916666666666666
best parameters :  {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 100, 'random_state': 42}
SVM  (best score) :  0.995
best parameters :  {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
KNN  (best score) :  0.9900000000000001
best parameters :  {'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 1, 'weights': 'uniform'}
```

Figure 5-24: Grid search 2.

Step 3: Apply classifiers

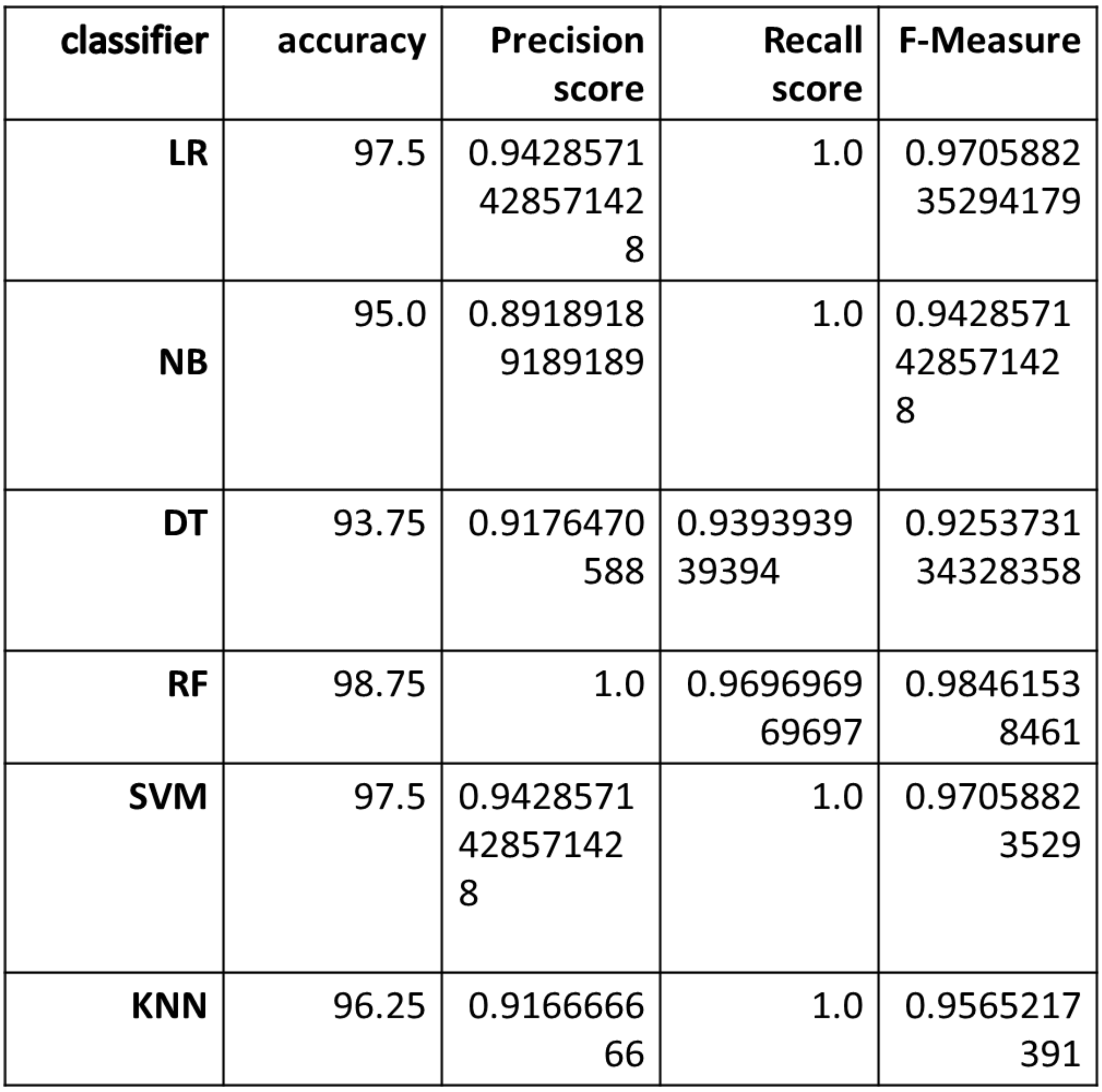


```
In [21]: # Data Modeling with the best parameters output
         models = []
         models.append(("LR",LogisticRegression(C = 0.01, multi_class = 'multinomial',
                                                penalty = 'l2', solver = 'newton-cg')))
         models.append(("NB",GaussianNB()))
         models.append(("DT",DecisionTreeClassifier(criterion = 'gini', max_features = 'sqrt',
                                                    random_state = 12, splitter = 'best')))
         models.append(("RF",RandomForestClassifier(criterion = 'entropy', max_features = 'sqrt',
                                                    n_estimators =  100, random_state = 0)))
         models.append(("SVM",SVC(C = 0.1, gamma = 'auto', kernel =  'poly')))
         models.append(("KNN", KNeighborsClassifier(algorithm = 'auto', n_jobs = -1,
                                                    n_neighbors = 7 , weights = 'uniform')))
```

```
In [28]: # Evaluating Classificatiom Algorithms
         from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score
         for model_name , model in models :
             model.fit(X_train, y_train)
             y_predicted = model.predict(X_test)
             acc_score = accuracy_score(y_test, y_predicted)
             print(model_name, " accuracy : ",acc_score*100)
             print("F-Measure : ",f1_score(y_test,y_predicted, average = 'macro'),"\n")
```

Figure 5-25: Data Modeling.

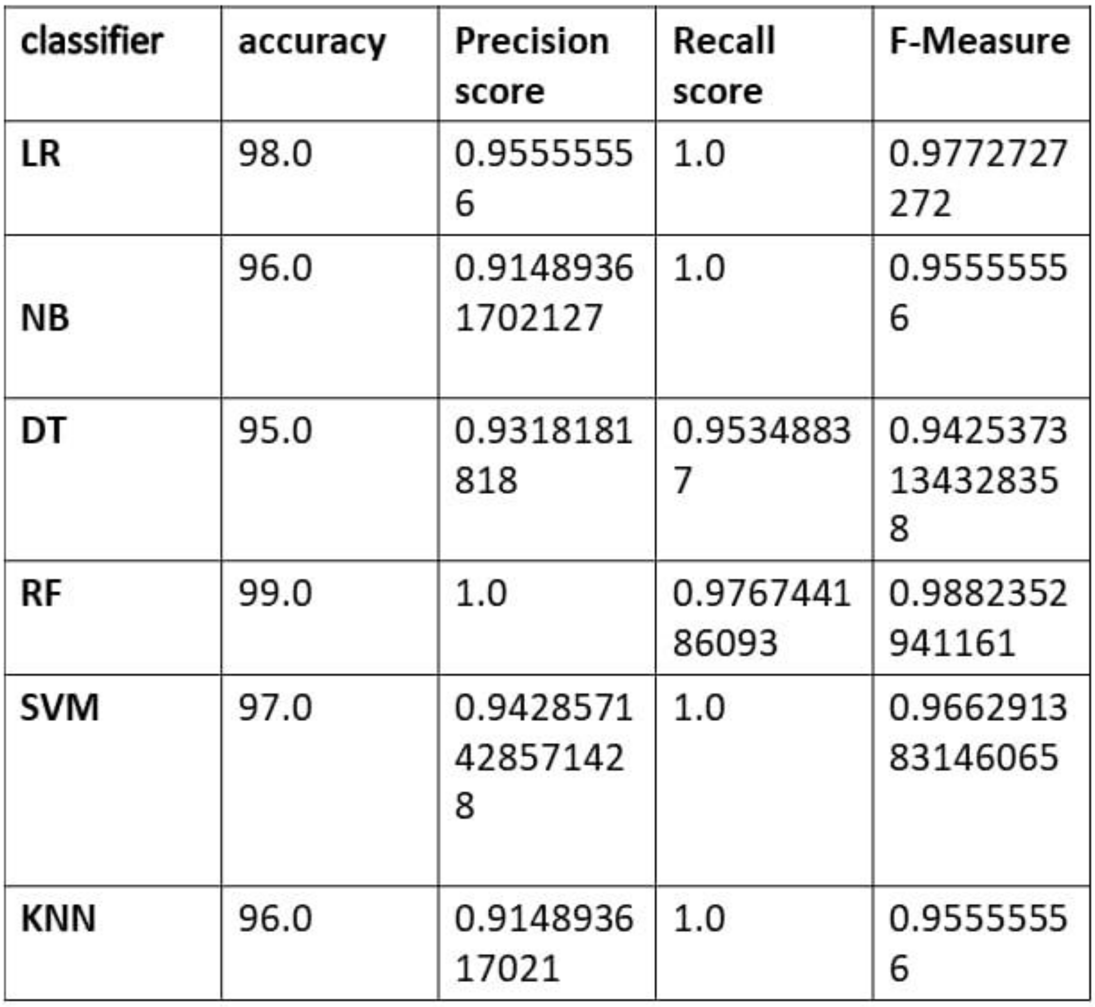
Test case 20 %



classifier	accuracy	Precision score	Recall score	F-Measure
LR	97.5	0.94285714285714 28	1.0	0.970588235294179
NB	95.0	0.89189189189189	1.0	0.94285714285714 28
DT	93.75	0.9176470588	0.939393939394	0.925373134328358
RF	98.75	1.0	0.969696969697	0.98461538461
SVM	97.5	0.94285714285714 28	1.0	0.97058823529
KNN	96.25	0.916666666	1.0	0.9565217391

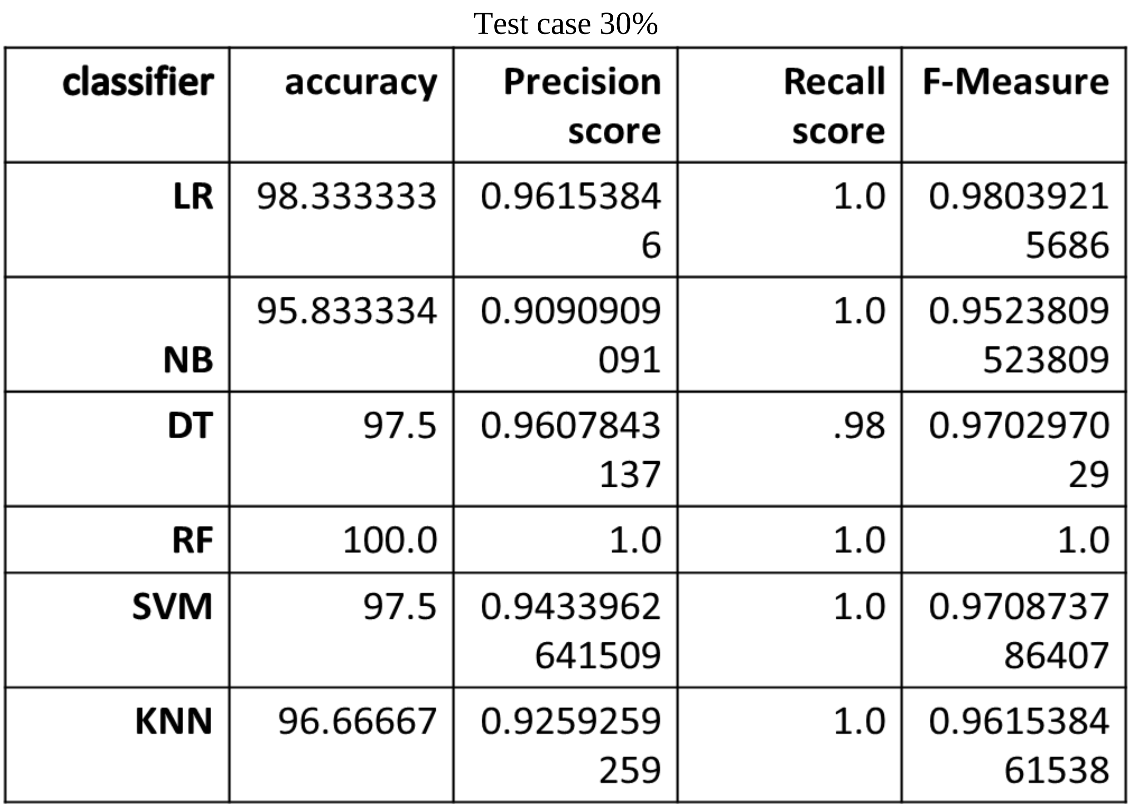
Figure 5-26: Test case 20%.

Test case 25 %



classifier	accuracy	Precision score	Recall score	F-Measure
LR	98.0	0.95555556	1.0	0.9772727272
NB	96.0	0.91489361702127	1.0	0.95555556
DT	95.0	0.9318181818	0.95348837	0.94253731343283358
RF	99.0	1.0	0.976744186093	0.9882352941161
SVM	97.0	0.942857142857142 8	1.0	0.966291383146065
KNN	96.0	0.914893617021	1.0	0.95555556

Figure 5-27:Test case 25 %.

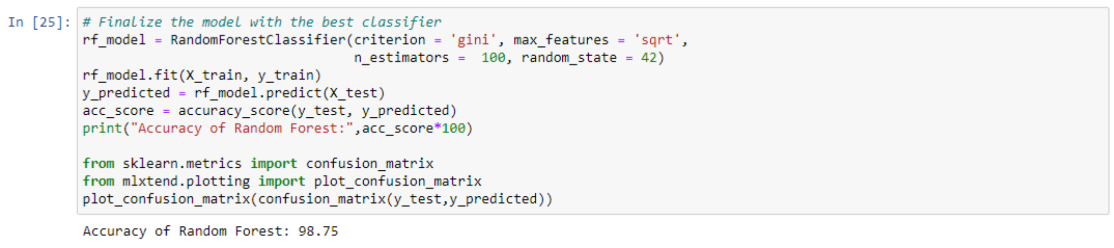


Test case 30%

classifier	accuracy	Precision score	Recall score	F-Measure
LR	98.333333	0.96153846	1.0	0.98039215686
NB	95.833334	0.9090909091	1.0	0.9523809523809
DT	97.5	0.9607843137	.98	0.970297029
RF	100.0	1.0	1.0	1.0
SVM	97.5	0.9433962641509	1.0	0.970873786407
KNN	96.66667	0.9259259259	1.0	0.961538461538

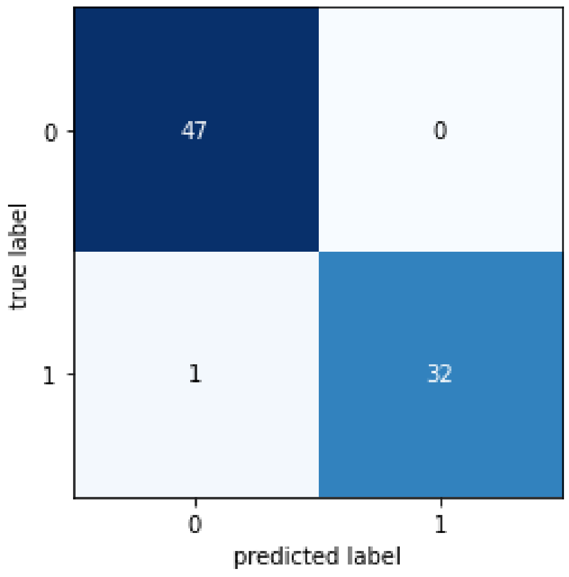
Figure 5-28: Test case 30%.

We find that random Forest Classifier is the beat classifier to our mode as like table of accuracy above.



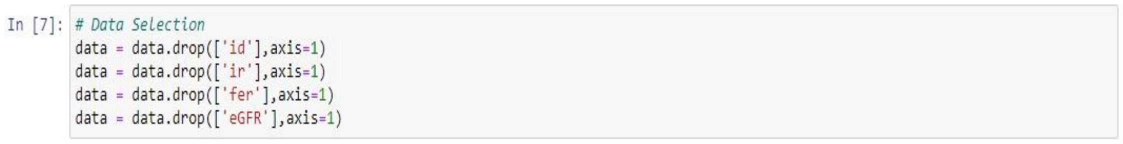
```
In [25]: # Finalize the model with the best classifier
         rf_model = RandomForestClassifier(criterion = 'gini', max_features = 'sqrt',
                                           n_estimators =  100, random_state = 42)
         rf_model.fit(X_train, y_train)
         y_predicted = rf_model.predict(X_test)
         acc_score = accuracy_score(y_test, y_predicted)
         print("Accuracy of Random Forest:",acc_score*100)

         from sklearn.metrics import confusion_matrix
         from mlxtend.plotting import plot_confusion_matrix
         plot_confusion_matrix(confusion_matrix(y_test,y_predicted))

         Accuracy of Random Forest: 98.75
```

Figure 5-29: Test case 30%



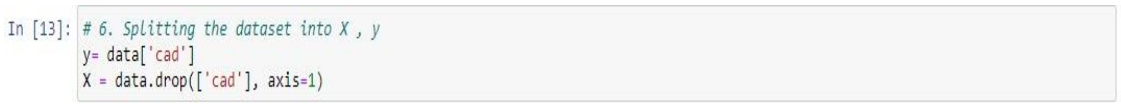
5.2.2 Model 2

Applying same steps above on model 2 to predict coronary artery disease base on the output of prediction and diagnosis of chronic kidney disease with change the data selection step to involve the output of the classification on input feature.



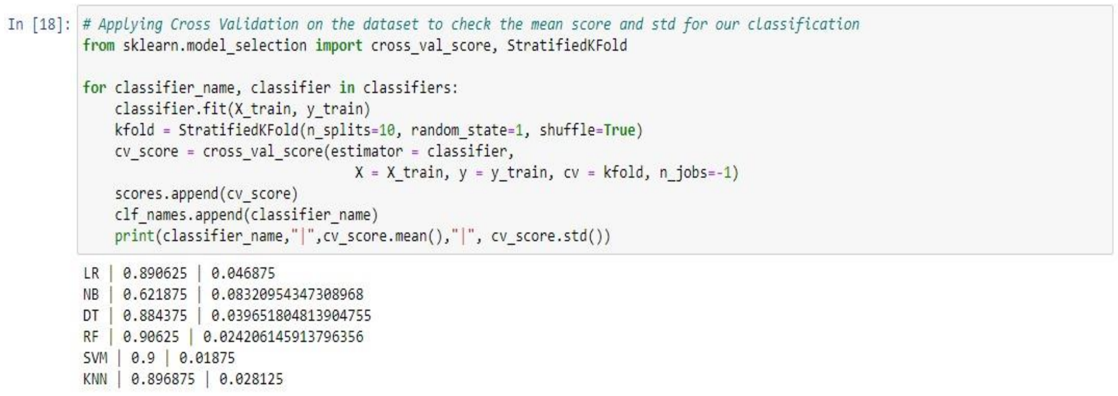
```
In [7]: # Data Selection
        data = data.drop(['id'],axis=1)
        data = data.drop(['ir'],axis=1)
        data = data.drop(['fer'],axis=1)
        data = data.drop(['eGFR'],axis=1)
```

Figure 5-30:Data selection.



```
In [13]: # 6. Splitting the dataset into X , y
         y= data['cad']
         X = data.drop(['cad'], axis=1)
```

Figure 5-31:Data splitting.



```
In [18]: # Applying Cross Validation on the dataset to check the mean score and std for our classification
         from sklearn.model_selection import cross_val_score, StratifiedKFold

         for classifier_name, classifier in classifiers:
             classifier.fit(X_train, y_train)
             kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
             cv_score = cross_val_score(estimator = classifier,
                                        X = X_train, y = y_train, cv = kfold, n_jobs=-1)
             scores.append(cv_score)
             clf_names.append(classifier_name)
             print(classifier_name,"|",cv_score.mean(),"|", cv_score.std())

         LR | 0.890625 | 0.046875
         NB | 0.621875 | 0.08320954347308968
         DT | 0.884375 | 0.039651804813904755
         RF | 0.90625 | 0.024206145913796356
         SVM | 0.9 | 0.01875
         KNN | 0.896875 | 0.028125
```

Figure 5-32: Cross validation.

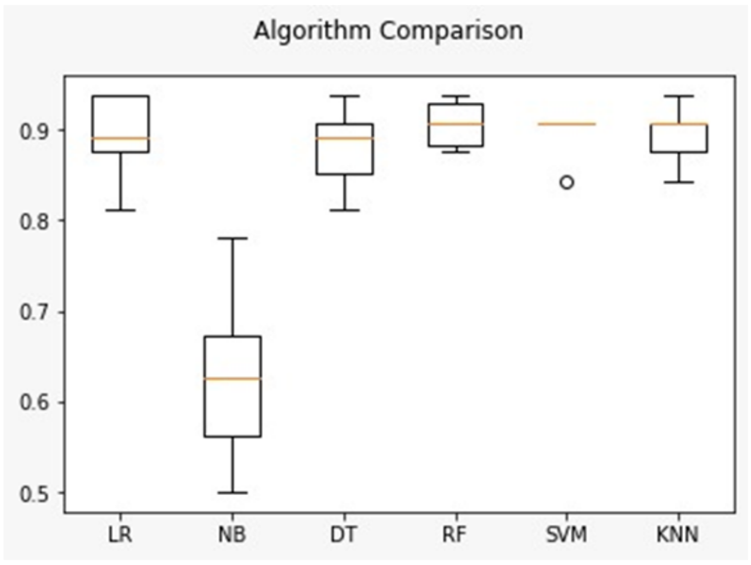
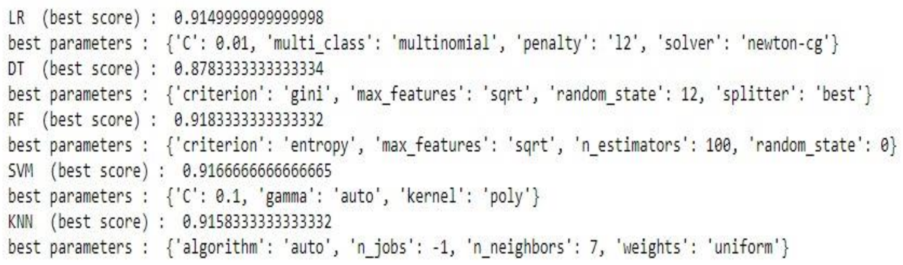
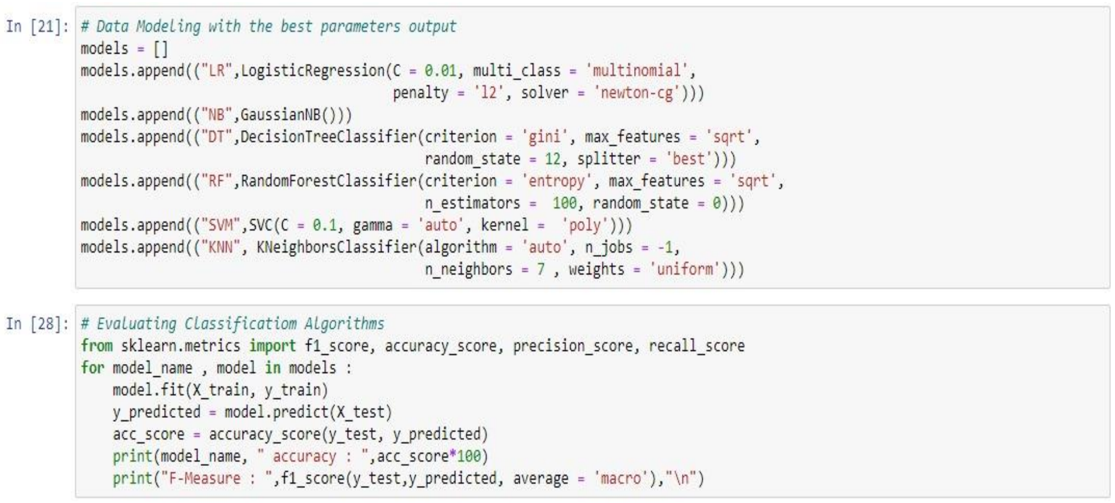


Figure 5-33: Algorithm comparison.



```
LR  (best score) :  0.9149999999999998
best parameters :  {'C': 0.01, 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'}
DT  (best score) :  0.8783333333333334
best parameters :  {'criterion': 'gini', 'max_features': 'sqrt', 'random_state': 12, 'splitter': 'best'}
RF  (best score) :  0.9183333333333332
best parameters :  {'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 100, 'random_state': 0}
SVM  (best score) :  0.9166666666666665
best parameters :  {'C': 0.1, 'gamma': 'auto', 'kernel': 'poly'}
KNN  (best score) :  0.9158333333333332
best parameters :  {'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 7, 'weights': 'uniform'}
```

Figure 5-34: Best parameters GridSearch

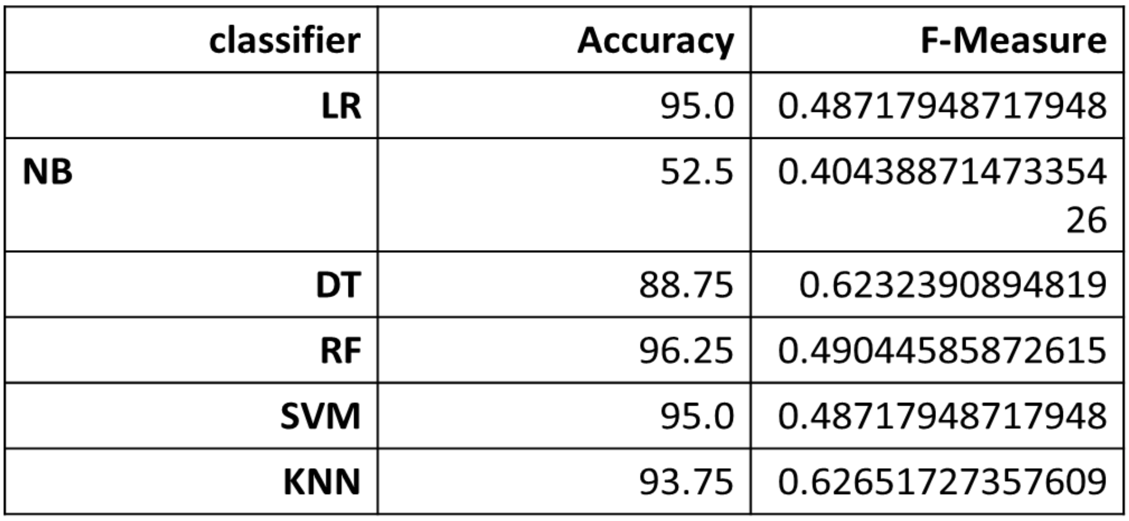


```
In [21]: # Data Modeling with the best parameters output
         models = []
         models.append(("LR",LogisticRegression(C = 0.01, multi_class = 'multinomial',
                                                penalty = 'l2', solver = 'newton-cg')))
         models.append(("NB",GaussianNB()))
         models.append(("DT",DecisionTreeClassifier(criterion = 'gini', max_features = 'sqrt',
                                                    random_state = 12, splitter = 'best')))
         models.append(("RF",RandomForestClassifier(criterion = 'entropy', max_features = 'sqrt',
                                                    n_estimators =  100, random_state = 0)))
         models.append(("SVM",SVC(C = 0.1, gamma = 'auto', kernel =  'poly')))
         models.append(("KNN", KNeighborsClassifier(algorithm = 'auto', n_jobs = -1,
                                                    n_neighbors = 7 , weights = 'uniform')))
```

```
In [28]: # Evaluating Classification Algorithms
         from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score
         for model_name , model in models :
             model.fit(X_train, y_train)
             y_predicted = model.predict(X_test)
             acc_score = accuracy_score(y_test, y_predicted)
             print(model_name, " accuracy : ",acc_score*100)
             print("F-Measure : ",f1_score(y_test,y_predicted, average = 'macro'),"\n")
```

Figure 5-35: Data modeling.

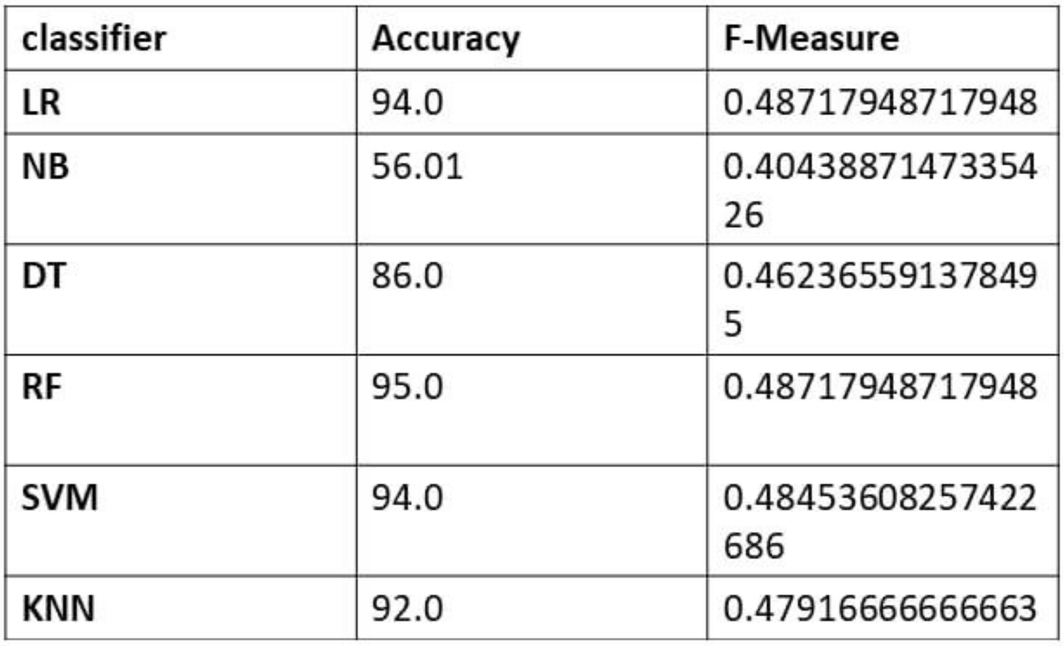
Test case 20 %



classifier	Accuracy	F-Measure
LR	95.0	0.48717948717948
NB	52.5	0.4043887147335426
DT	88.75	0.6232390894819
RF	96.25	0.49044585872615
SVM	95.0	0.48717948717948
KNN	93.75	0.62651727357609

Figure 5-36: Test case 20 %.

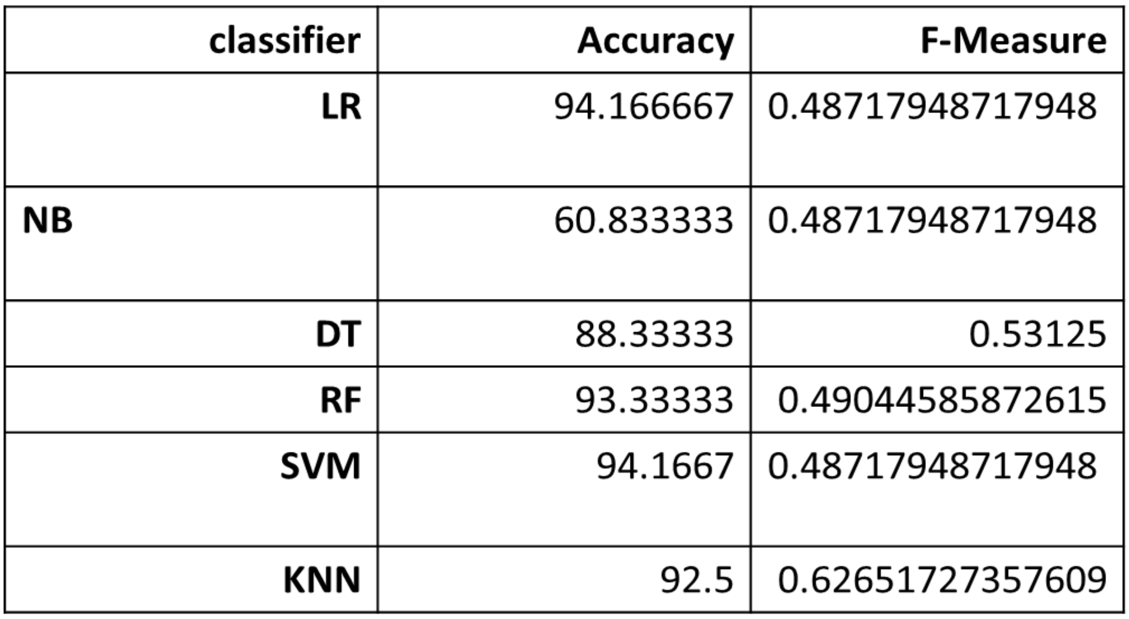
Test case 25 %



classifier	Accuracy	F-Measure
LR	94.0	0.48717948717948
NB	56.01	0.4043887147335426
DT	86.0	0.462365591378495
RF	95.0	0.48717948717948
SVM	94.0	0.48453608257422686
KNN	92.0	0.47916666666663

Figure 5-37: Test case 25 %.

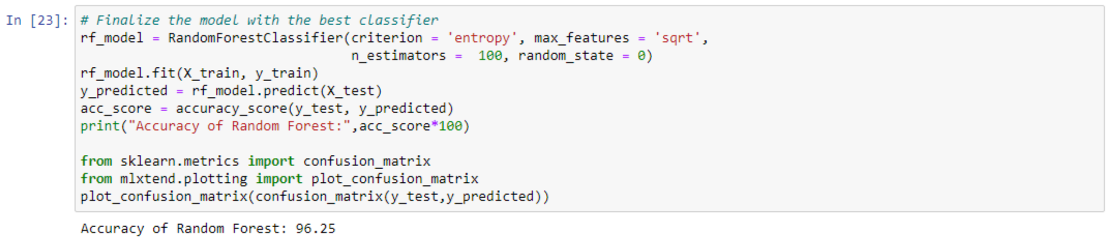
Test case 30 %



classifier	Accuracy	F-Measure
LR	94.166667	0.48717948717948
NB	60.833333	0.48717948717948
DT	88.33333	0.53125
RF	93.33333	0.49044585872615
SVM	94.1667	0.48717948717948
KNN	92.5	0.62651727357609

Figure 5-38: Test case 30 %.

We find that random Forest Classifier is the beat classifier to our mode as like table of accuracy above.



```
In [23]: # Finalize the model with the best classifier
         rf_model = RandomForestClassifier(criterion = 'entropy', max_features = 'sqrt',
                                           n_estimators =  100, random_state = 0)
         rf_model.fit(X_train, y_train)
         y_predicted = rf_model.predict(X_test)
         acc_score = accuracy_score(y_test, y_predicted)
         print("Accuracy of Random Forest:",acc_score*100)

         from sklearn.metrics import confusion_matrix
         from mlxtend.plotting import plot_confusion_matrix
         plot_confusion_matrix(confusion_matrix(y_test,y_predicted))

         Accuracy of Random Forest: 96.25
```

Figure 5-39: RF classifier.

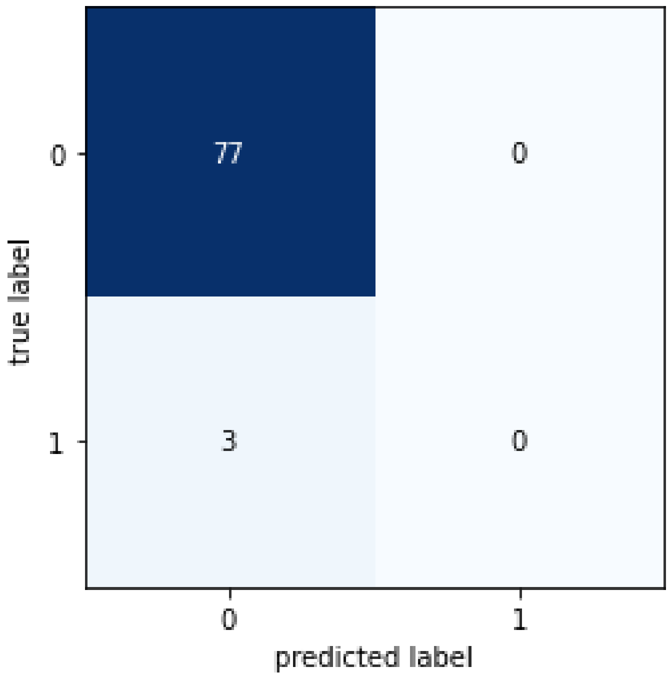
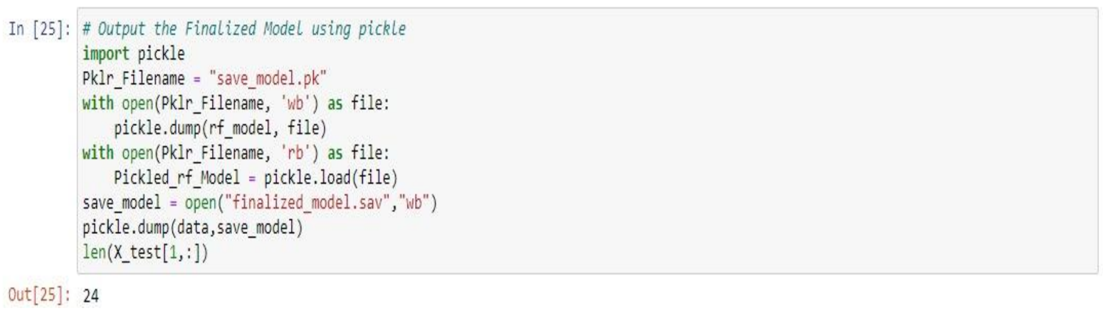


Figure 5-39: Confusion matrix.

Output of finalized model to connect website by flask method



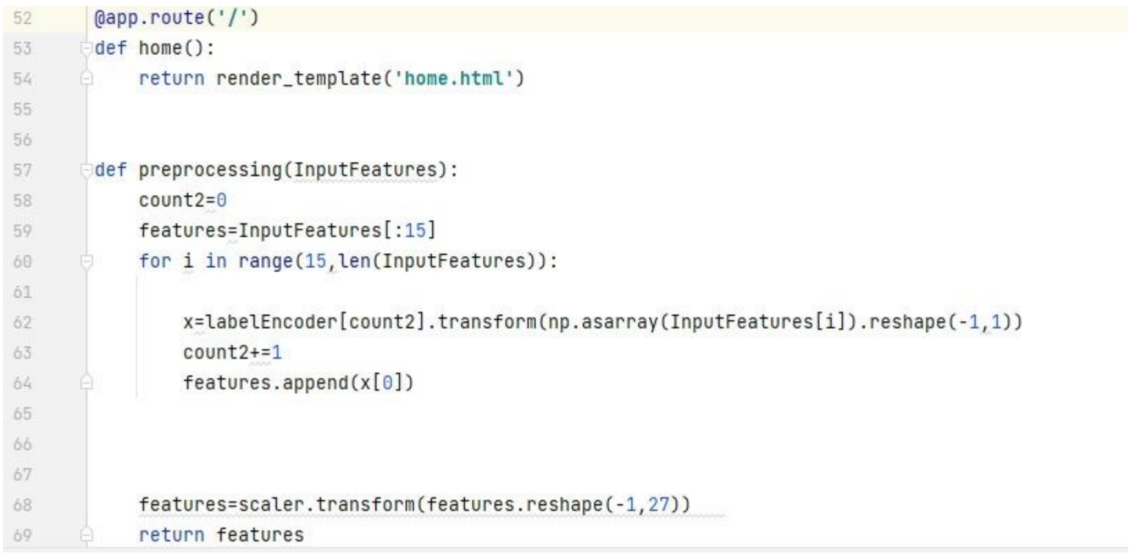
```
In [25]: # Output the Finalized Model using pickle
         import pickle
         Pklr_Filename = "save_model.pk"
         with open(Pklr_Filename, 'wb') as file:
             pickle.dump(rf_model, file)
         with open(Pklr_Filename, 'rb') as file:
             Pickled_rf_Model = pickle.load(file)
         save_model = open("finalized_model.sav","wb")
         pickle.dump(data,save_model)
         len(X_test[1,:])

Out[25]: 24
```

Figure 5-40: Finalized model.

SOURCE CODE OF FLASK MODELS

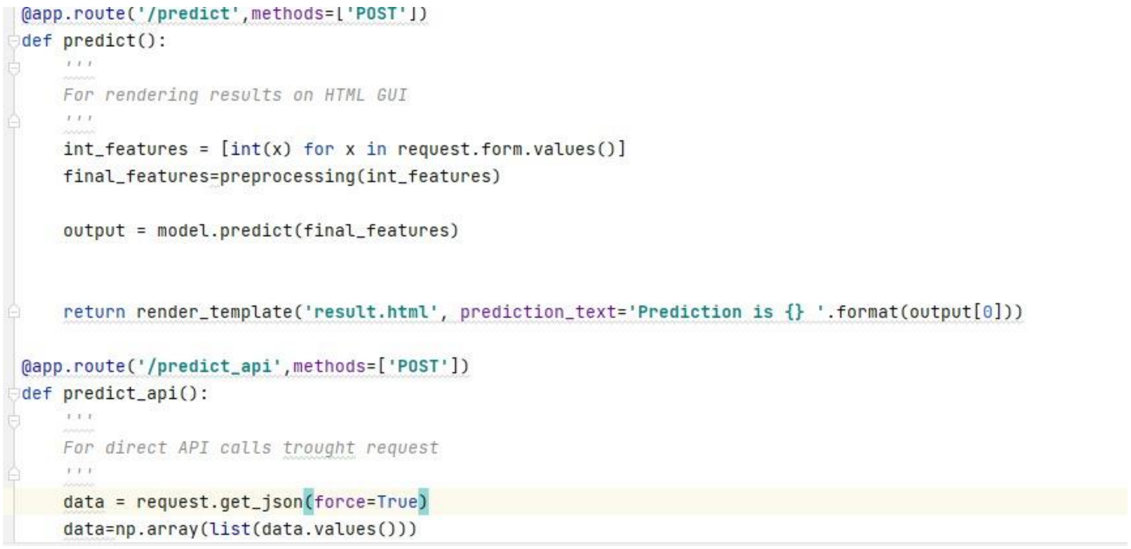
Connection between machine models and web Application by (flask).

Its function to connect between web and machine, we use flask to get data from the website and send it to machine model classifiers to get the result (if patient have CDK then predict with having HF or not) then send the result back to the website to show the result to doctor.



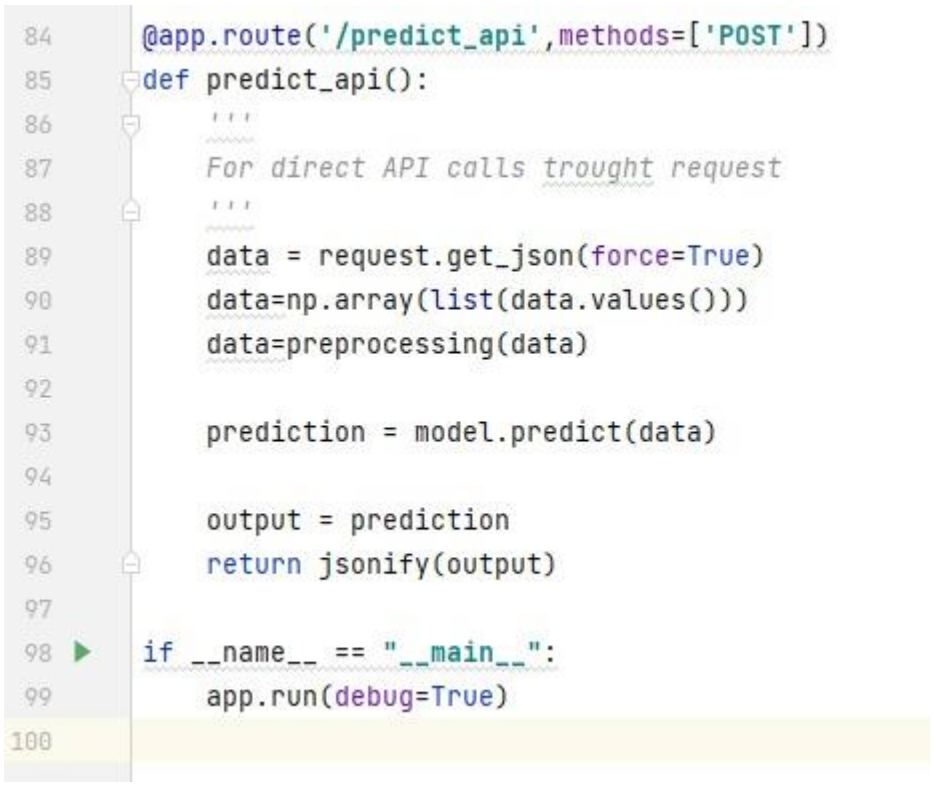
```
@app.route('/')
def home():
    return render_template('home.html')


def preprocessing(InputFeatures):
    count2=0
    features=InputFeatures[:15]
    for i in range(15,len(InputFeatures)):

        x=labelEncoder[count2].transform(np.asarray(InputFeatures[i]).reshape(-1,1))
        count2+=1
        features.append(x[0])



    features=scaler.transform(features.reshape(-1,27))
    return features
```

Figure 5-41: Flask coda part 1.



```
@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features=preprocessing(int_features)

    output = model.predict(final_features)


    return render_template('result.html', prediction_text='Prediction is {} '.format(output[0]))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    data=np.array(list(data.values()))
```

Figure 5-42: Flask coda part 2.



```
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    data=np.array(list(data.values()))
    data=preprocessing(data)

    prediction = model.predict(data)

    output = prediction
    return jsonify(output)

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

```

Figure 5-43: Flask coda part 2.

C h a p t e r   S i x

6 CONCLUSIONS

As a result, we have studied Advantages and Disadvantages of Machine Learning. Also, this blog helps an individual to understand why one needs to choose machine learning. While Machine Learning can be incredibly powerful when used in the right ways and in the right places (where massive training data sets are available) and The importance of machine learning Machine learning as a technology helps analyze large chunks of data, which facilitates the task of data scientists in an automated process and gains a lot of importance and recognition. Machine learning also benefits in high-value predictions that can guide better decisions and intelligent actions in real time without human intervention. This is the use of Amazon machine learning to predict what customers want and provide it to them, which helps it generate huge profits from behind this matter.

In this project, we meet our main purpose and provide machine learning and our data to apply our project idea to help patients and doctors by firstly we make machine model then we service and Attachment them by using flask.

For the work in the future, we will work to improve and give the project more additional features in the future.

REFERENCES

DataFlair. 2021. Advantages and Disadvantages of Machine Learning Language - DataFlair. [online] Available at: < https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/ >

IEEE DataPort. 2021. *Heart Disease Dataset (Comprehensive)*. [online] Available at: < https://ieee-dataport.org/open-access/heart-disease-dataset-comprehensive#files >

Kaggle.com. 2021. *What Causes Heart Disease? Explaining the Model*. [online] Available at: <https://www.kaggle.com/tentotheminus9/what-causes-heart-disease-explaining-the-model >

Kaggle.com. 2021. *heart_disease_prediction_SVC*. [online] Available at: <https://www.kaggle.com/savitanair/heart-disease-prediction-svc>

Kaggle.com. 2021. *Heart Disease UCI dataset Analysis*. [online] Available at: <https://www.kaggle.com/anthonycoplo/heart-disease-uci-dataset-analysis>

Kaggle.com. 2021. *notebookd143b543c1*. [online] Available at: <https://www.kaggle.com/crucifierbladex/notebookd143b543c1>

Kaggle.com. 2021. *EC524: Heart-disease classification | Kaggle*. [online] Available at: <https://www.kaggle.com/c/ec524-heart-disease/data>

Ripublication.com. 2021. [online] Available at: <https://www.ripublication.com/ijaerspl2019/ijaerv14n10spl_30.pdf>

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 10, 2019 (Special Issue) © Research India Publications. http://www.ripublication.com

Ripublication.com. 2021. [online] Available at: <https://www.ripublication.com/ijaerspl2019/ijaerv14n10spl_30.pdf>

Heart disease prediction using machine learning techniques : a survey , published: 2/8/2018 <https://www.researchgate.net/profile/V-V-Ramalingam/publication/325116774_Heart_disease_prediction_using_machine_learning_techniques_A_survey/links/5d48560a299bf1995b68266f/Heart-disease-prediction-using-machine-learning-techniques-A-survey.pdf>

Chronic Kidney Disease Prediction Using Machine Learning , published: April, 2018 <https://d1wqtxts1xzle7.cloudfront.net/56493051/41_Paper_310318109_IJCSIS_Camera_Ready_pp.308-311.pdf?1525505024=&response-content-disposition=inline%3B+filename%3DChronic_Kidney_Disease_Prediction_Using.pdf&Expires=1620169520&Signature=FI3XMUxC~8io4cZM8SVOp3sOU4K56~-95of6eDBDunnt7mdN5ky7aICDt2PDpHtdoec4CTEeKGiIuQ3NWRulu5a37hRdT9~12c0HH0TdZbkwlB5QKMUozK84Uy2kseiuC10ByJ3XPKRLSMpo6ikiiDC97sca2zPeZKRU540AoUFd1jqoA~z0Y92d~v-SdYKXXpKn5FEI5WjH6I6AtDGYtckF20-sJZ~dWySqy--mHTvTskmqpvoQiX1ri8HtmOGmf427cdsLt37FquSvLXRiS0EB6ySFlrN9OWz9B6k898trqcaKEuq3MRg6uNwcEFkabiE1EbAG9mJV8h5Lcbbs8Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>

Applying Machine Learning Techniques for Predicting the Risk of Chronic Kidney Disease , published: August, 2016 <https://sciresol.s3.us-east-2.amazonaws.com/IJST/Articles/2016/Issue-29/Article29.pdf>